DETECÇÃO DE RÉPLICAS DE SÍTIOS WEB USANDO APRENDIZADO SEMISSUPERVISIONADO BASEADO EM MAXIMIZAÇÃO DE EXPECTATIVAS

CRISTIANO RODRIGUES DE CARVALHO

DETECÇÃO DE RÉPLICAS DE SÍTIOS WEB USANDO APRENDIZADO SEMISSUPERVISIONADO BASEADO EM MAXIMIZAÇÃO DE EXPECTATIVAS

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

ORIENTADOR: NIVIO ZIVIANI
COORIENTADOR: ADRIANO VELOSO

Belo Horizonte
19 de setembro de 2014

Carvalho, Cristiano Rodrigues de

C331d Detecção de réplicas de sítios web usando aprendizado semissupervisionado baseado em maximização de expectativas / Cristiano Rodrigues de Carvalho. — Belo Horizonte, 2014
xiv, 56 p. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais. Departamento de Ciência da Computação
Orientador: Nivio Ziviani Coorientador: Adriano Veloso

1. Computação - Teses. 2. Aprendizado do computador. I. Orientador. II. Coorientador. III. Título.

CDU 519.6*82 (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 usando aprendizado semi-supervisionado baseado em maximização de expectativas

CRISTIANO RODRIGUES DE CARVALHO

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

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

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

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

PROF. RODRYGO LUIS TEODORO SANTOS
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 19 de setembro de 2014.

Agradecimentos

Este pequeno espaço de texto nunca vai ter tamanho suficiente para descrever o quanto eu sou grato a todos os responsáveis diretos e indiretos por esta conquista.

Primeiramente gostaria de agradecer aos meus pais, Lucília e Raimundo Carvalho, por sempre acreditarem e valorizarem as minhas escolhas, além de me darem o suporte necessário para atingir meus objetivos. Agradeço também aos meus irmãos Nilo César e Sandra Lúcia por sua amizade e por sempre estarem ao meu lado. Quero agradecer minha querida Maria Cristina por todo seu carinho e apoio durante esta fase tão difícil que foi minha formação como Mestre.

Ao professor Altigran Soares por ter me apresentado ao meu atual orientador prof. Nivio Ziviani. Além disso por ter me orientado durante a conclusão de minha graduação em Ciência da Computação e por me ajudar em momentos de dificuldade. Ao Roberto Oliveira por toda atenção e parceria como meu coorientador de projeto de graduação.

Gostaria de deixar evidente minha gratidão aos colegas do Laboratório para Tratamento da Informação (LATIN), muitos dos quais tenho orgulho de chamar de amigos. Agradeço a Aline Bessa, Arthur Câmara, Dilson Guimarães e Wallace Favoreto. Ao Rickson Guidolini pela amizade ainda em curto tempo de convivência e sua ajuda mesmo à distância. Ao Aécio Santos, meu parceiro de pesquisa mais próximo. Aos amigos e conselheiros Sabir Ribas e Anísio Lacerda pelo suporte imensurável durante o curso. Ao Wladmir Brandão e Adolfo Guimarães por sua amizade e parceria na manutenção do laboratório. Ao Leonardo Santos, nosso professor oficial de tênis. Ao Itamar Hata por ajudas essenciais, principalmente em trabalhos práticos, e ao recém chegado e também de muita boa vontade, Jordan Silva. Agradeço ao Thiago Sales, meu amigo do Laboratório de Banco de Dados (LBD), e ao Samuel Oliveira que também contribuiu com dicas valiosas para melhorar minha apresentação de defesa do mestrado.

Gostaria de agradecer ao meu orientador Nivio Ziviani e coorientador Adriano Veloso pelos ensinamentos que vão além do trabalho de pequisa. O tempo que passei sob a orientação de ambos foi de grande valor para minha formação como Mestre em Ciência da Computação, assim como para minha vida inteira. O nome deles estará sempre marcado em minha trajetória acadêmica e pessoal. Me sinto uma pessoa de muita sorte por ter trabalhado e aprendido tantas lições preciosas com pessoas de tamanha inteligência. Quero deixar registrado minha gratidão e também minha admiração por eles. Em especial agradeço ao professor Nivio por sua preocupação em criar as melhores condições para que pudesse realizar trabalho de pesquisa de qualidade.

Agradeço à banca examinadora composta por Nivio Ziviani, Adriano Alonso Veloso, Edleno Silva de Moura e Rodrygo Luis Teodoro Santos, por todas as sugestões que ajudaram a melhorar ainda mais este trabalho.

Por fim, gostaria de agradecer a todos os funcionários do Departamento de Ciência da Computação. Em especial à Sônia Borges, à Sheila dos Santos e ao Gabriel Teixeira, pela atenção e por estarem sempre à disposição em ajudar.

Resumo

A Web é um imenso repositório de informações. De acordo com a literatura aproximadamente 29% desse repositório contém conteúdo duplicado. A duplicação de conteúdo pode ocorrer dentro de um mesmo sítio web (intrassítios) ou entre sítios diferentes (intersítios). Esta dissertação trata do problema de detecção de réplicas intersítios. Neste trabalho, esse problema é tratado como uma tarefa de classificação, onde exemplos positivos e negativos de réplicas são utilizados no treinamento de um classificador binário.

O método proposto utiliza um algoritmo de aprendizado semissupervisionado baseado em Maximização de Expectativas (do inglês *Expectation-Maximization* - EM). O algoritmo EM é um método iterativo que permite a estimativa de parâmetros em modelos probabilísticos com dados latentes ou não observados. No caso de detecção de réplicas há uma facilidade de encontrar exemplos óbvios de réplicas e não réplicas. Nesse caso, o algoritmo EM é utilizado para encontrar exemplos não óbvios e formar um conjunto de treino para o algoritmo de classificação sem nenhum custo de uma rotulação manual.

É possível melhorar substancialmente a qualidade dos resultados obtidos com a combinação de classificadores através da exploração de um conceito da Economia, a Eficiência de Pareto. Mais especificamente, essa técnica permite a escolha de resultados que se sobressaem em pelo menos um dos classificadores utilizados. O algoritmo proposto provê ganhos significativos em relação ao estado-da-arte em detecção de réplicas de sítios.

A combinação do algoritmo proposto que elimina réplicas intersítios junto a algoritmos que eliminam réplicas de conteúdo intrassítios leva a uma solução mais completa, possibilitando uma redução mais efetiva do número de URLs duplicadas na coleção.

Palavras-chave: Réplicas de Sítios, Aprendizado de Máquina, Maximização de Expectativas, Pareto.

Abstract

The Web contains a vast repository of information. According to the literature about 29% of this repository contains duplicate content. Duplication of content may occur within a single web site (intra-site) or between different web sites (inter-site). This thesis addresses the problem of detecting inter-site replicas. In this work, this problem is treated as a classification task, where positive and negative replica examples are used to train a binary classifier.

The proposed method uses a semi-supervised learning algorithm based on the Expectation-Maximization (EM) approach. The EM algorithm is an iterative method that allows estimation of parameters in probabilistic models with latent or unobserved data. In replica detection, it is easy to find obvious replica and non-replica examples. The EM algorithm is used to find non-obvious examples and form a training set for the classification algorithm at no cost of manual labeling.

It is possible to substantially improve the quality of the results obtained with the combination of classifiers by exploring a central concept of Economics, the Pareto efficiency. More specifically, this technique allows to choose results that excel in at least one of the classifiers used. The proposed algorithm provides significant gains compared to state-of-art in detection of website replicas.

The combination of proposed algorithm that eliminates inter-site replicas with algorithms that eliminate intra-sites replica content leads to a more complete solution allowing an effective reduction in the number of duplicated URLs on the collection.

Keywords: Website replicas, Machine Learning, Expectation-Maximization, Pareto.

Sumário

Capítulo 1

Introdução

A Web é o maior repositório de informação já construído pelo homem. Esse sucesso está bastante relacionado à facilidade com que pessoas, em todo o mundo, conseguem publicar conteúdo na Web. O controle sobre as publicações, que antes estava nas mãos das editoras, graças ao surgimento da Web, hoje passa a integrar o cotidiano de pessoas comuns. Qualquer pessoa pode criar documentos eletrônicos de naturezas distintas e, em muitos casos, de maneira anônima.

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, os sistemas conhecidos como máquinas de busca tratam a necessidade de se organizar e encontrar informações em meio a esse enorme volume de dados. 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 à necessidade do usuário.

Um dos problemas enfrentados no desenvolvimento de máquinas de busca é a presença de conteúdo replicado em suas coleções de dados. Estudos na literatura estimam que cerca de 29% da informação na Web é replicada [Broder et al., 1997; Fetterly et al., 2003; Henzinger, 2006]. No caso das máquinas de busca, a replicação de informação acarreta uma série de problemas, entre eles o desperdício de tempo durante a coleta e processamento da informação coletada, o desperdício de banda de Internet, de recursos em armazenamento e processamento, entre outros. Assim sendo, vários estudos vêm sendo realizados com o intuito de minimizar os problemas relacionados à replicação de dados na Web. Além de desenvolver técnicas que minimizem os problemas enfrentamos com a duplicação de conteúdo durante a busca por informações de qualidade, uma das motivações para este trabalho é a

utilização das técnicas desenvolvidas em uma máquina de busca real, em construção dentro do Instituto Nacional de Ciência e Tecnologia para a Web[1] (InWeb).

A duplicação de conteúdo na Web pode ser dividida em duas categorias, sendo (i) intrassítios, se ela ocorre dentro de um mesmo sítio web, ou (ii) intersítios, se acontece entre sítios distintos. A forma de tratar esses dois problemas é diferente. Abordagens de detecção de conteúdo duplicado intrassítios são baseadas em algoritmos que processam URLs de um único sítio em busca de documentos duplicados internamente. Por outro lado, algoritmos para detecção de duplicatas intersítios precisam processar URLs através de sítios distintos, cujo objetivo é encontrar documentos compartilhados entre aqueles até então tratados como sítios diferentes.

Embora a maioria dos trabalhos presentes na literatura considerem a deduplicação de URLs intrassítios [Agarwal et al., 2009; Bar-Yossef et al., 2007, 2009; Dasgupta et al., 2008; Koppula et al., 2010], uma solução completa deve também considerar o tratamento de conteúdo duplicado entre sítios distintos, uma vez que URLs duplicadas entre sítios diferentes não seriam detectadas utilizando apenas a abordagem intrassítios. Dentre os poucos trabalhos existentes na literatura sobre a abordagem intersítios, podemos citar [Bharat & Broder, 1999; Bharat et al., 2000; da Costa Carvalho et al., 2007]. Esses trabalhos procuram detectar sítios replicados em coleções web.

As principais causas da replicação de sítios web são:

- Prefixo padrão: sítios que possuem uma versão com prefixo WWW e outra sem o prefixo WWW sendo indexadas.
- 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.
- Franquia: um mesmo sítio é mantido por dois parceiros diferentes com duas URLs diferentes, mas com conteúdo semelhante.
- Hospedagem: Sítios que passam por um processo de transferência para outra companhia de hospedagem.
- Objetivos maliciosos: Tentativa de aumentar as chances de um conteúdo ser listado ou ter várias listagens em máquinas de busca através da criação de múltiplas copias idênticas ou similares de seu sítio.

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

O problema estudado neste trabalho é a detecção de réplicas de sítios web em coleções de documentos de máquinas de busca. Quando um sítio é replicado, além dos problemas relacionados ao desperdício de recursos computacionais, a informação à respeito da conectividade desse sítio também é replicada. Isso interfere diretamente no funcionamento de algoritmos de *ranking* como o PageRank [Brin & Page, 1998], afetando assim a qualidade das respostas exibidas ao usuário. 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 por lhe passar a sensação de estar recebendo respostas repetidas. A remoção redundância em rankings de respostas é uma tarefa importante para melhorar a qualidade da busca por informações úteis [Bernstein & Zobel, 2005].

Detectar réplicas de sítios é uma tarefa importante, pois previne a duplicação de conteúdo intersítios em coleções de documentos de máquinas de busca. Ao reduzir a presença de conteúdo duplicado são evitados 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 favorece uma maior cobertura de sítios visitados, demandando os mesmos recursos de uma coleta sem remoção de réplicas.

Vários fatores tornam a detecção de réplicas uma tarefa difícil. Uma ideia inicial para resolver este problema seria, por exemplo, verificar se os sítios da base possuem o mesmo conjunto de páginas. No entanto, realizar essa verificação na base de dados da máquina de busca é inviável, devido à estratégia de cobertura em largura adotada por coletores web, em que é priorizada a busca por novos sítios ao invés de sítios completos. Isso dificulta o processo de se encontrar interseções entre os conjuntos de páginas de sítios replicados nas bases das maquinas de busca. Outra tentativa poderia ser a utilização do endereço IP como identificador único para cada sítio. Porém um mesmo sítio pode responder sob vários endereços IPs diferentes assim como um mesmo IP pode estar relacionado a sítios distintos.

A detecção de pares de sítios replicados em bases de máquinas de busca também se torna uma tarefa difícil devido à dimensionalidade do problema. Uma solução por força bruta, por exemplo, requer a comparação entre todos os pares de sítios da base, o que leva a uma complexidade quadrática. Portanto é preciso encontrar maneiras mais eficientes de se resolver o problema.

1.1 Objetivos

Esta dissertação tem por objetivo principal propor e avaliar abordagens baseadas em aprendizado de máquina para identificação de sítios web replicados em bases de máquinas

de busca. Mais especificamente, modelamos o problema como uma tarefa de classificação. Neste caso, um conjunto de treinamento é utilizado para produzir um classificador que relaciona características dos pares de sítios à probabilidade desse par ser ou não uma réplica. Esse classificador é então utilizado para identificar automaticamente quais pares são replicados em um conjunto de teste. Atualmente, os trabalhos presentes na literatura tratam a seleção de sítios replicados por meio de heurísticas fixas que não envolvem um processo de aprendizado.

Devido à alta complexidade do problema, este trabalho propõe a divisão da tarefa de deduplicação em duas etapas: uma seleção inicial de pares candidatos à réplica, que reduz a dimensionalidade do problema, e em seguida, uma validação dos pares candidatos extraídos na primeira etapa, verificando quais dos pares selecionados são realmente réplicas através das técnicas de classificação.

Para verificar a qualidade das técnicas de classificação desenvolvidas neste trabalho é necessário obter uma coleção de dados que seja uma representação fiel da Web. Nessa coleção, são necessários pares de sítios rotulados como positivos ou negativos, que permitam o treinamento dos métodos de classificação propostos. Porém, o grande volume de dados presente em uma coleção de sítios web torna inviável a rotulação manual dos pares de sítios coletados. Portanto, este trabalho tem o objetivo de prover uma estratégia que, a partir de exemplos óbvios e simples de se obter, possibilite a identificação de novos exemplos não óbvios, construindo assim um conjunto de treino sem a necessidade de anotação humana.

1.2 Contribuições

A principal contribuição deste trabalho é propor um algoritmo para detecção de réplicas de sítios web baseado em técnicas de aprendizado de máquina. Podemos citar como contribuições específicas as seguintes:

- *Algoritmo*: O algoritmo de classificação proposto (Capítulo 3), que utiliza aprendizado semissupervisionado a partir de exemplos óbvios de réplicas e provê ganhos significativos em relação ao estado-da-arte em detecção de réplicas de sítios [da Costa Carvalho et al., 2007].
- *Coleção de teste*: A criação de uma coleção de teste que torna possível a comparação de resultados da nossa abordagem com outras técnicas presentes na literatura e que também pode contribuir para o avanço dessa área de pesquisa. Essa coleção é descrita em detalhes na Seção 4.1
- *Treinamento sem custo de anotação humana*: Uma abordagem para aquisição automática de exemplos de treino a partir de um conjunto inicial de exemplos óbvios e sem custo de aquisição, descrita na Seção 3.3.3.

- *Escolha ótima de parâmetros*: Ao invés de utilizar parâmetros globais para todo conjunto de treino, o algoritmo proposto é capaz de aprender modelos individuais para cada instância e utiliza essa capacidade para estimar parâmetros que são ótimos localmente. Esse processo é detalhado na Seção 3.3.3.
- *Combinação de classificadores*: Mostramos como podemos melhorar substancialmente nossos resultados com a combinação de classificadores através da exploração de um conceito central da Economia, a Eficiência de Pareto. Essa estratégia é descrita na Seção 3.3.4.
- *Combinação com técnicas de detecção de réplicas intrassítios*: Avaliamos a combinação do nosso algoritmo que elimina réplicas intersítios junto a algoritmos que eliminam réplicas de conteúdo intrassítios. Especificamente, aplicar esses algoritmos em conjunto nos leva a uma solução mais completa, possibilitando uma redução de até 21% em relação ao número de URLs duplicadas na coleção (Seção 4.5).

1.3 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 e fundamentais para o bom entendimento do restante da dissertação. No Capítulo 3, caracterizamos o problema tratado neste trabalho e descrevemos a metodologia adotada para criação do método proposto. O Capítulo 4 apresenta e discute o estudo realizado, bem como seus resultados. Finalmente, no Capítulo 5, são apresentadas as conclusões e as possíveis direções para trabalhos futuros.

Capítulo 2

Referencial Teórico e Trabalhos Relacionados

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

2.1 Conteúdo Web

2.1.1 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 2.1, a sequencia 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.

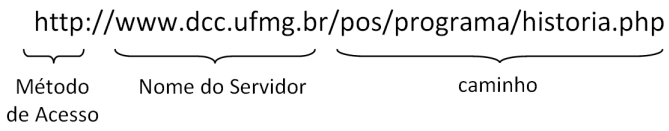

Figura 2.1: Os três campos da URL.

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 o prefixo

identifica um servidor específico sob esse domínio. No exemplo da Figura 2.1, 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. No exemplo da URL na Figura 2.1, 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 ou profundidade do caminho e pode ser definido a partir do número de "/"(barras) da URL.

2.1.2 Definição de Sítios 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] utilizam 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 sítio web, esta dissertação adota a seguinte definição para descrever um sítio:

> Um sítio web é o conjunto de páginas da Web que compartilham um mesmo nome de servidor.

2.2 Máquinas de Busca

Uma máquina de busca é criada a partir da aplicação prática de técnicas de recuperação de informação em grandes coleções de texto [Croft et al., 2009]. Apesar de as máquinas de busca para a Web serem as mais conhecidas, elas não são as únicas. Esse tipo de sistema está presente em muitas outras áreas e aplicações: computadores, *smartphones*, e-mails, aplicações empresariais, entre outros. Quando se utiliza a funcionalidade *pesquisar* do Microsoft Windows, por exemplo, se está utilizando na verdade uma máquina de busca que age sobre a coleção de documentos existente no computador em questão. O foco do restante desta dissertação são as máquinas de busca para Web, as quais serão referidas apenas como máquinas de busca.

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 apresentar os resultados de forma clara e atraente ao usuário, entre outros. Para realizar todas essas tarefas é necessário que uma série de componentes trabalhem em conjunto. Em seguida são apresentados os três componentes principais de uma máquina de busca.

2.2.1 Coletor Web

Mesmo que seja utilizada a mais nova e sofisticada tecnologia aplicada a uma máquina de busca, a informação de boa qualidade contida em seus documentos é que a torna útil [Croft et al., 2009]. Por isso, o coletor web é um dos componentes mais importantes 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 a questões como "*quando e o que coletar?*". Porém, é difícil responder a essas questões. Certamente, a melhor resposta para "*o que*" seria "*tudo o que for útil para 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, os coletores trabalham em ciclos compostos por três etapas:

i. *Fetching*: o sistema que recupera os documentos na Internet, chamado de *fetcher*, recebe uma lista de URLs a serem coletadas, dispara várias requisições HTTP em paralelo 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 à 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.

2.2.2 Indexador

Uma vez que diversos documentos foram coletados, é preciso capturar sua informação e transformá-la em um formato que o computador possa interpretar. Além disso, é necessário que seja possível pesquisar, quase que instantaneamente, o conteúdo desses documentos. 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 busca 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 grande tamanho. Em coleções enormes como a Web, o elevado número de termos pode fazer com que o processamento do índice 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 maneira 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].

2.2.3 Processador de Consultas

Uma vez que os documentos já foram coletados e indexados, a máquina de busca se torna 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 da qual ele necessita. Essa consulta é então modificada

por meio de transformações semelhantes às 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 em relação à consulta submetida pelo usuário.

Um problema encontrado durante o processamento de consultas é o de prever 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 "mp3" procurando por informações sobre onde baixar conteúdo em mp3. Outro usuário, porém, pode formular a mesma consulta procurando por informações técnicas sobre tipos de compressão de áudio, ou ainda pode haver um terceiro usuário que estaria interessado em comprar dispositivos que permitem ouvir musicas em mp3.

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]. Dificilmente uma pessoa diria simplesmente a palavra "correr" a outra, se quisesse uma resposta sobre a regência verbal desse verbo. No entanto, essa é uma situação muito comum em que as máquinas de busca enfrentam diariamente: os usuários, geralmente, não estão dispostos a digitar grandes frases para explicar o que procuram.

Existem vários modelos de recuperação de informação, os quais 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 [Frakes & Baeza-Yates, 1992] (um dos primeiros), o modelo vetorial [Salton & Lesk, 1968; Salton & Yang, 1973] (considerado o mais popular) e o modelo probabilístico [Stephen E. Robertson, 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. O livro publicado por Baeza-Yates & Ribeiro-Neto [2011] é uma boa leitura para os interessados em saber mais sobre os modelos de recuperação de informação.

2.3 Aprendizado de Máquina

Mitchell [1997] define aprendizado de máquina como o estudo de algoritmos computacionais capazes de melhorar automaticamente a execução de alguma tarefa através da experiência. Algoritmos de aprendizado de máquina se baseiam principalmente em probabilidade e estatística para aprender padrões complexos a partir de uma entrada de dados e usá-los na tomada "inteligente" de decisões sobre algum assunto. Segundo Alpaydin [2004], técnicas de aprendizado de máquina são utilizadas para realização de inferências sobre dados futuros, mesmo sem saber a priori qual processo gerou os dados de entrada. Pode ser que

não seja possível identificar completamente o processo que gerou os dados, mas é possível construir uma aproximação boa e útil. Suas áreas de aplicação são abundantes: classificação de clientes para aplicações de crédito utilizada pelos bancos, análise de sentimento, desambiguação de entidades, recomendação de rótulos (*tags*), ordenação de resultados de máquinas de busca, diagnóstico de câncer de pulmão, entre outras.

Este trabalho lida com aprendizado de máquina supervisionado aplicado à classificação de dados. De acordo com Grünwald & Langford [2007], um problema de aprendizado de máquina é definido em um domínio de entrada (ou característica) X, um domínio de saída (ou rótulo da classe) $\mathcal{Y}$ e uma distribuição de probabilidade $\mathcal{P}$ sobre $X \times \mathcal{Y}$, tal que um classificador é uma função $\mathcal{F}_S : X \rightarrow \mathcal{Y}$.

Este trabalho parte da hipótese de que algoritmos de aprendizado de máquina podem melhorar os resultados do estado-da-arte em detecção de réplicas de sítios web, a partir de uma seleção e combinação automática de características que melhor descrevem pares replicados e consequentemente maximizem a efetividade na tarefa de detecção de réplicas.

2.3.1 Classificação Associativa Sob Demanda (LAC)

O classificador associativo sob demanda (Lazy Associative Classifier - LAC) [Veloso & Meira Jr., 2011] 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. 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 $\mathcal{R}$ sobre uma instância $x_i \in \mathcal{T}$, denotadas por $\mathcal{R}^{x_i}$. Mais especificamente, sempre que uma entrada $x_i \in \mathcal{T}$ é processada, o classificador usa x_i como um filtro para remover de $\mathcal{R}$ atributos e exemplos que não são úteis para gerar o modelo de classificação para x_i. Uma vez que $\mathcal{R}^{x_i}$ contém apenas valores presentes em x_i, todas as regras de associação geradas a partir de $\mathcal{R}^{x_i}$ 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.

As características utilizadas neste trabalho foram discretizadas por uma metodologia não supervisionada [Dougherty et al., 1995]. Os valores contínuos de cada característica foram divididos em intervalos discretos. A identificação nominal de cada um desses intervalos é utilizada como valor das características, substituindo assim os valores contínuos originais.

2.3.2 Maximização de Expectativas (EM)

O método de Maximização de Expectativas (*"Expectation-Maximization"* - EM) é um método iterativo que permite a estimação de parâmetros em modelos probabilísticos com dados incompletos (dados latentes ou não observados). Em outras palavras, o EM, proposto por Dempster et al. [1977], é utilizado para estimar parâmetros de modelos probabilísticos em que existem variáveis não observadas.

O método envolve dois passos que são repetidos até que haja convergência. Os dois passos são:

- Passo E: estima-se os dados que faltam para completar a amostra de dados incompleta. Essa estimativa é feita usando os valores das variáveis que foram observadas.
- Passo M: com as novas frequências esperadas obtidas anteriormente, aplica-se um algoritmo com dados completos. Sendo assim, esses novos parâmetros substituem os parâmetros anteriores.

Neste trabalho, a utilização de um modelo baseado em EM é justificada pela hipótese de que vale a pena utilizar exemplos não rotulados no processo de construção do modelo de treino do algoritmo, uma vez que tais dados não rotulados são abundantes e obtidos com pouco esforço. O Capítulo 3 descreve as abordagens utilizadas neste trabalho e que foram baseadas no algoritmo EM.

2.3.3 Máquinas de Vetores de Suporte (SVM)

Máquinas de Vetores de Suporte (Support Vector Machines - SVM) [Boser et al., 1992] consistem de métodos de espaço vetorial para problemas de classificação binária, isto é, onde existem apenas duas classes possíveis. A idéia 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.

O conceito de melhor separação está relacionado à margem entre duas classes. A Figura 2.2 ilustra a definição de dois hiperplanos de decisão diferentes para o mesmo problema. As linhas mais escuras representam os hiperplanos de decisão. As linhas mais claras, paralelas aos hiperplanos de decisão, são chamadas hiperplanos delimitadores e representam os limites extremos de cada classe. A distância entre os hiperplanos delimitadores é chamada de margem e representa o quanto o hiperplano de decisão pode ser movido sem causar erros.

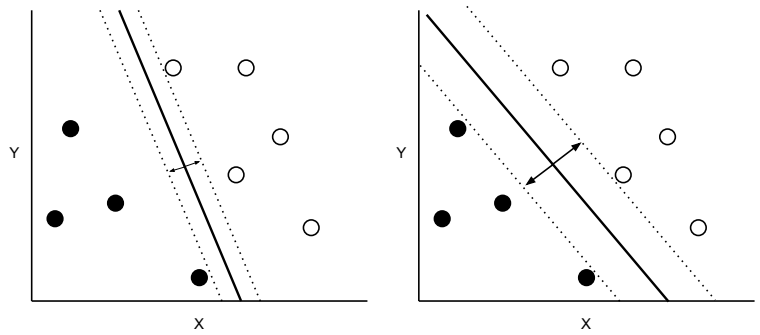

Figura 2.2: Exemplos de hiperplanos diferentes aplicados ao mesmo problema

O gráfico da esquerda apresenta um hiperplano que produz uma margem menor que a margem gerada pelo hiperplano do gráfico à direita. Nesse caso, o hiperplano à direita é o que maximiza a margem. A ideia é que quanto maior for a margem, maior será a generalização do algoritmo ao classificar instâncias novas. Dessa forma, o SVM resolve o problema de otimização que busca o hiperplano de decisão que maximiza a margem entre as instâncias das classes na coleção de treino.

Para resolver problemas não separáveis linearmente, podem ser utilizados hiperplanos que permitem algum erro de treino [Cortes & Vapnik, 1995] ou 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.

Uma propriedade importante do algoritmo SVM é a de que o hiperplano de decisão é determinado apenas por instâncias de classes que se sobrepõem aos hiperplanos delimitadores. Essas 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.

2.4 Eficiência de Pareto

O conceito de Eficiência de Pareto teve início na Economia e foi desenvolvido pelo italiano Vilfredo Pareto. Sua definição pode ser dada por: se não existe maneira de melhorar a situação de uma pessoa sem piorar a situação de outra, a solução encontrada satisfaz à Eficiência de Pareto. O subconjunto de soluções que satisfazem à condição de Eficiência de Pareto formam a Fronteira de Pareto [Börzsönyi et al., 2001]. Formalmente, a Fronteira de Pareto consiste em um subconjunto de pontos tal que nenhum desses pontos é dominado por qualquer outro. Um ponto em $\mathcal{R}^d$ consiste de um vetor de números reais de tamanho d. Um ponto domina o outro se ele é melhor ou igual em todas as dimensões e é melhor em pelo menos uma dimensão.

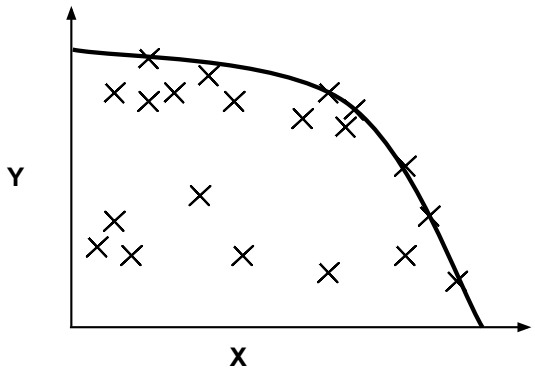

Figura 2.3: Exemplo de Fronteira de Pareto em duas dimensões

A Eficiência de Pareto tem sido utilizada em outras áreas além da Economia, como, por exemplo, otimização (Ribeiro et al. [2014]). Na área de otimização, a Eficiência de Pareto é utilizada quando se deseja maximizar uma função multiobjetivo, sendo a Fronteira de Pareto o conjunto de soluções ótimas. Neste trabalho essa técnica será utilizada na fase de detecção de pares replicados com o objetivo de agregar resultados de diferentes modelos de predição.

2.5 Detecção de Réplicas

A grande quantidade de dados replicados na Web tem motivado pesquisas com o objetivo de caracterizar e entender como tratar o problema. Nesse sentido, Ye et al. [2008] realizaram um trabalho com o objetivo de detectar duplicatas de documentos a partir da comparação do conteúdo desses documentos. Para isso, os autores utilizaram uma técnica conhecida como *shingles*. Essa técnica consiste em medir qual é a proporção de conteúdo textual igual entre os documentos comparados.

Embora essa técnica seja bastante eficiente, o uso de conteúdo textual para detectar réplicas em tempo de coleta tem custo bastante elevado. Pensando nisso, trabalhos anteriores na literatura buscaram minimizar o uso de conteúdo das páginas durante a fase de deduplicação.

Em [Bharat & Broder, 1999] os autores propuseram um método automático para a detecção de réplicas de sítios em coleções de máquinas de busca composto por duas fases, em que apenas na segunda etapa é utilizado o conteúdo textual das páginas. Primeiramente, pares de sítios candidatos à réplica são selecionados com base em características obtidas a partir das URLs das páginas. Em seguida, os pares candidatos são verificados em busca de réplicas. Esse teste consiste em coletar amostras de páginas de cada sítio para verificar a ocorrência de páginas em comum. Eles reportam uma precisão de apenas 80%, o que significa que um grande número de sítios seriam incorretamente rotulados como réplicas e removidos da base de dados da máquina de busca. Além disso, coletar diversas páginas adicionais de sítios para se verificar casos de réplica é um processo de alto custo, principalmente quando o número de candidatos é muito grande.

Uma alternativa para eliminar conteúdo duplicado é conhecida como detecção de páginas DUST (*Different URLs with Similar Text*). Nesse caso, o problema é formulado como a tarefa de detectar réplicas de conteúdo a partir apenas da análise de URLs. O primeiro algoritmo a adotar essa estratégia foi o DustBuster [Bar-Yossef et al., 2009], que detecta DUST através da criação de regras capazes de transformar determinada URL em uma outra URL que provavelmente tem o mesmo conteúdo. As regras consistem em substituições de partes do texto das URLs. Essas regras são aprendidas através de logs de coleta ou de logs da Web. O trabalho de Dasgupta et al. [2008] apresenta outra formalização para reescrita de URLs que é capaz de encontrar todas regras previamente encontradas pelo DustBuster, e também padrões mais gerais, como a presença de partes irrelevantes da URL e parâmetros de id de sessões. Experimentos mostraram uma redução de até 60% no número de URLs duplicadas.

Os autores em [Agarwal et al., 2009] estenderam o trabalho em Dasgupta et al. [2008] para produzir uma solução mais escalável. Eles propuseram um algoritmo para derivação de regras através de amostras de URLs, portanto reduzindo o custo de se inferir tais regras. Mais adiante foi utilizado um algoritmo de árvore de decisão para aprender um conjunto pequeno de regras de alta precisão, diminuindo assim o número total de regras a serem aplicadas na coleta. A avaliação do algoritmo foi feita em um conjunto de aproximadamente 8 milhões de URLs e o método atingiu uma taxa de redução de duplicatas de 42% utilizando as top 9% regras mais precisas (cujo nível de precisão é acima de 80%). Em [Koppula et al., 2010] os autores implementaram um *framework* distribuído e estenderam as representações de uma URL e das regras para incluir novos padrões. Eles avaliaram o método proposto com 3 bilhões de URLs e atingiram uma redução maior em conteúdo DUST utilizando apenas 56% das regras.

Em [Rodrigues et al., 2013], os autores apresentaram um novo método inspirado em algoritmos de alinhamentos multi-sequenciais para derivar regras usadas na eliminação de DUST. Assim como em [Agarwal et al., 2009], uma amostragem de URLs é agrupada através da comparação de seus conteúdos. Esses *clusters* de URLs com conteúdo similar são denominados *dup-clusters*. Primeiramente todas URLs são alinhadas nos *dup-clusters* obtendo um padrão em comum para cada *dupcluster*. As regras são então derivadas desses padrões levando a uma redução em número de URLs duplicadas de 54%.

Em outra direção relacionada a detecção de conteúdo duplicado levando em consideração a prevenção de desperdícios de recursos em tempo de coleta, Bharat et al. [2000] apresentam uma família de métodos para detecção de réplicas baseados em evidências derivadas de estruturas dos sítios web, tais como URLs, endereços IP e informações sobre *links*.

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. Os autores argumentam que 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, contudo, consumir recursos computacionais adicionais. Segundo os autores, o NormPaths conseguiu uma melhoria de 47% em precisão quando comparado com método proposto anteriormente. Por se tratar do estado-da-arte em detecção de réplicas de sítios web, o NormPaths será usado como *baseline* neste trabalho e é descrito em mais detalhes na Seção 4.2.

Os autores em [Cho et al., 2000] lidaram com um problema levemente diferente, uma vez que a tarefa é a detecção de coleções web duplicadas ao invés de sítios duplicados. Note que detectar coleções duplicadas inclui a detecção de sítios ou fragmentos de sítios replicados. A abordagem proposta primeiramente agrupa páginas com conteúdo similar e então expande esses *clusters* de modo que eles representem coleções inteiras.

Assim como [Bharat et al., 2000] e [da Costa Carvalho et al., 2007], este trabalho trata o problema de detecção de sítios duplicados em tempo de coleta. A maior diferença entre este e os trabalhos anteriores é o uso de classificadores na combinação de várias características. Utilizamos também, uma nova abordagem baseada em aprendizado de máquina com o intuito de treinar o algoritmo de classificação sem a necessidade de esforço manual na criação dos modelos de treino.

Capítulo 3

O 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. O algoritmo proposto 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.

3.1 Descrição do Algoritmo

A tarefa de detectar sítios replicados em base de dados de uma máquina de busca poderia ser executada por meio de um algoritmo que utiliza força bruta para comparação de todos os pares de sítios na base. Para uma base contendo s sítios, se cada sítio for comparado com todos os demais, o algoritmo terá uma complexidade $O(s^2)$. O tamanho do problema a ser resolvido torna proibitivo o uso de algoritmos de complexidade quadrática. Por exemplo, para uma coleção de aproximadamente 600 mil sítios como a que será utilizada neste trabalho, deveriam ser analisados por volta de 2×10^{11} pares, o que inviabiliza uma solução por força bruta.

O algoritmo proposto utiliza uma solução simples para reduzir o número de pares possíveis, descartando aqueles pares que obviamente não podem ser candidatos a serem réplicas. Em seguida, o conjunto menor de candidatos selecionados é então refinado através de técnicas de aprendizado de máquina em busca dos pares propriamente replicados.

A Figura 3.1 apresenta uma descrição geral do algoritmo proposto para detecção de réplicas. O algoritmo é dividido em duas fases:

1. Na fase 1, características mais gerais são utilizadas como filtro inicial para selecionar um conjunto de pares de sítios que têm algum potencial para serem réplicas, chamados de candidatos.

2. Na fase 2, características mais específicas são obtidas para o conjunto de pares gerados na fase anterior e submetidas a um algoritmo de aprendizado de máquina que retorna uma lista de pares de sítios e suas respectivas probabilidades de serem réplicas. Para isso, um conjunto de treino formado por pares de sítios previamente rotulados como réplicas ou não réplicas é utilizado por um método que avalia cada par candidato e identifica os pares replicados.

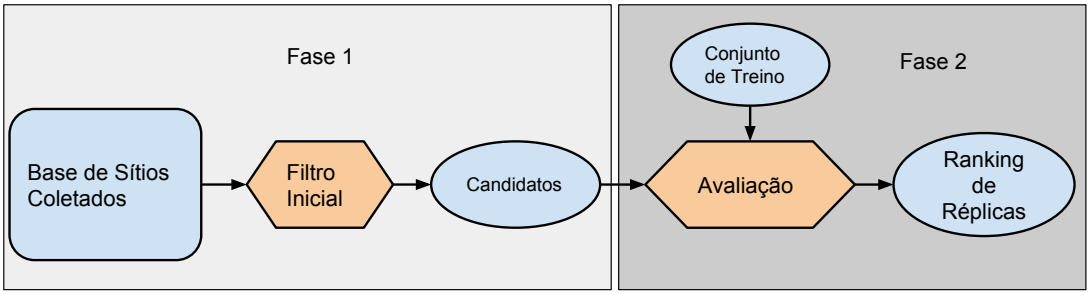

Figura 3.1: Descrição geral do algoritmo proposto

O Programa 1 apresenta um refinamento do algoritmo proposto.

Programa 1 Detecção de réplicas de sítios web.

Entrada: base de dados B de uma máquina de busca.
Saída: conjunto C de pares de sítios ordenados pela probabilidade de serem replicados.
$\rightarrow$ ***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
$\rightarrow$ ***2ª Fase:*** *Avaliação 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: com base em $\mathcal{F}_r$, obtenha, do algoritmo de aprendizado de máquina, a probabilidade de s_1 e s_2 serem réplicas
8: Ordene os pares em C pela pontuação dada pelo algoritmo de aprendizado de máquina de acordo com a chance de o par ser replicado

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 de características possui duas propriedades que tornam adequada a filtragem

inicial sobre uma grande quantidade de pares: garante uma alta revocação e possui baixo custo de processamento. 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 avaliação do conjunto de pares candidatos a serem réplicas, o algoritmo de classificação, baseado em um novo conjunto de características, determina uma probabilidade de cada par do conjunto C ser ou não uma réplica.

Uma etapa importante que precede a utilização dos algoritmos é a modelagem dos dados, onde são avaliadas e selecionadas as características dos sítios da Web que serão utilizadas na identificação de pares de sítios duplicados. Portanto a próxima seção apresenta a modelagem dos dados e em seguida é descrito o refinamento das fases do algoritmo proposto.

3.2 Modelagem dos Dados

Esta seção trada a descrição e obtenção das características utilizadas pelo algoritmo proposto. 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.

As duas primeiras características apresentadas, *Caminho da URL* e *Assinatura do Conteúdo*, correspondem ao conjunto de características mais gerais, ou conjunto básico, utilizadas em um filtro inicial que corresponde à primeira fase do algoritmo, conforme descrito na Seção 3.3.

a) *Caminho da URL*: Essa característica foi proposta por Bharat et al. [2000]. Um par de sítios A e B são considerados candidatos a réplica se A e B têm pelo menos um caminho em comum.

b) *Assinatura do Conteúdo*: Especificamente é utilizada uma função *hash* para obter a assinatura para o conteúdo das páginas dos sítios armazenadas. Na máquina de busca utilizada para obter a coleção usada nos experimentos, a assinatura de cada página é gerada durante a fase de coleta. Um par de sítios A e B são candidatos a réplica se A e B possuem pelo menos uma página com a mesma assinatura de conteúdo.

As características seguintes formam o conjunto de refinamento, que corresponde às características mais específicas utilizadas na segunda fase do algoritmo, conforme discutido na Seção 3.3.

i) *Distância de Edição (ndist)*: Essa característica proposta neste trabalho consiste em utilizar a distancia de edição [Levenshtein, 1966] entre duas URLs como atributo. Dado um par de sítios A e B, e os nomes de servidor correspondentes uA e uB, a distância de edição é dada pelo número de operações de remoção, inserção e substituição necessárias para transformar uA em uB.

ii) *Correspondência de Nomes de Servidor (nmatch)*: Essa característica foi proposta por Bharat et al. [2000]. Os segmentos de um nome de servidor são tratados como um vetor de termos. O peso de um termo t é dado pela Equação 3.1, onde $len(t)$ é o número de caracteres de t e $df(t)$ é o número de nomes de servidor que possuem o termo t:

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

Sejam dois sítios A e B e seus respectivos vetores de termos a e b, a correspondência entre seus nomes de servidor é dada pelo cosseno do ângulo entre a e b, como descrito na Equação 3.2:

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

iii) *Quatro Octetos (ip4)*:

O argumento para essa característica proposta em [Bharat et al., 2000] é que se os nomes de servidor de dois sítios são traduzidos para o mesmo endereço IP é provável que esses sítios sejam réplicas presentes no mesmo servidor. Porém, quando muitos nomes de servidor são traduzidos para o mesmo endereço IP pode ser um caso de hospedagem virtual, em que vários sítios diferentes são armazenados em um mesmo servidor web. Essas hospedagens são úteis quando um indivíduo não quer ou não pode manter um servidor web próprio.

Para calcular essa característica baseada nos quatro octetos do endereço IP é necessário o agrupamento de todos sítios da base da máquina de busca com mesmo endereço IP. Sejam dois sítios A e B, o valor do atributo é dado pela Equação 3.3, onde $\mathcal{G}$ é um agrupamento qualquer de sítios. A ideia é que quanto maior o grupo, menor a possibilidade de os sítios do grupo serem réplicas:

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

iv) *Três Octetos (ip3)*: O mesmo que a anterior, porém utilizando apenas os primeiros 3 octetos do endereço IP.

v) *Correspondência entre Caminhos Completos (fullpath)*: Esse atributo, proposto por [Bharat et al., 2000], representa os sítios como documentos e seus caminhos como termos dos documentos. Nesse caso o caminho inteiro é usado como um termo. Cada sítio (ou documento) é representado por um vetor de termos (ou caminhos). 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 termos da coleção. Todos termos com frequência acima de 100 são descartados, assim $maxdf$ é efetivamente 100:

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

Sejam dois sítios A e B e seus respectivos vetores de termos $\vec{a}$ e $\vec{b}$, o valor da característica entre A e B é dado pelo cosseno do ângulo entre seus vetores de termos, como descrito na Equação 3.2. As informações $df(t)$ e $maxdf$ devem ser obtidas antes que a característica possa ser utilizada.

3.3 Refinamento das Fases do Algoritmo

A partir da modelagem dos dados e do conhecimento das características utilizadas no algoritmo proposto, podemos passar ao refinamento de suas duas fases.

3.3.1 Seleção de Pares Candidatos

A maioria dos pares de sítios nas bases de máquinas de busca é formada por sítios que não são réplicas. O objetivo da fase de seleção de pares de 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 um filtro baseado no conjunto básico de características, que é composto pela característica *caminho da URL* (Seção 3.2 Item a) e *assinatura do conteúdo* (Seção 3.2 Item b).

Além de descartar um grande volume de pares de sítios que obviamente não são candidatos a réplicas, a metodologia desenvolvida para seleção de pares candidatos a réplicas tem custo linear em função do número de páginas na base da máquina de busca.

O Programa 2 mostra o refinamento do algoritmo em questão.

A complexidade linear é alcançada ao se fazer apenas uma varredura na base de entrada, criando listas de sítios relacionados, isto é, que possuam ao menos uma página com

Programa 2 Seleção de pares de sítios candidatos a 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 as características básicas F_b de d_i
3:     adicione o sítio de d_i às listas L_j de sítios relacionados correspondentes as características F_b
4: para todo Lista L_j faça
5:     monte todos os pares de sítios possíveis e armazene no conjunto C de candidatos a réplica
```

determinada característica do conjunto básico em comum. Para cada página da base, o primeiro passo insere seu sítio nas listas correspondentes a suas características. Para cada lista de sítios relacionados, o segundo passo monta todos os pares possíveis e, com isso, obtém 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 3.2. A estrutura é composta por uma tabela de valores para uma determinada característica. Para cada entrada da tabela existe 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 3.2 para cada característica do conjunto básico. Vale ressaltar que a varredura da base é orientada a documentos (páginas web) de modo linear e não a pares de sítios.

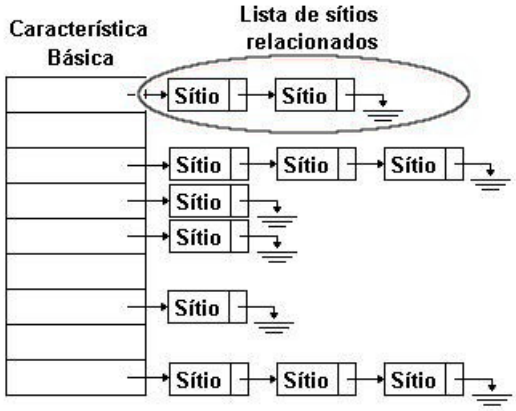

Figura 3.2: Estrutura de dados da Tabela de sítios Relacionados.

O conjunto básico de características utilizado possui duas propriedades que tornam adequada a filtragem inicial sobre uma grande quantidade de sítios: garante uma alta revocação e possui baixo custo de processamento. Durante a fase de seleção de candidatos a réplica, todo par de sítios que compartilhe alguma característica do conjunto básico em comum (pelo menos um caminho de URL ou um documento igual) é adicionado ao conjunto de candidatos a réplica.

As chances de se encontrar um caminho em comum entre sítios distintos é maior do que a de ser encontrado um conteúdo em comum. O número de caminhos conhecidos é extremamente maior do que o número de páginas propriamente coletadas. Além disso podem existir vários tipos de caminhos padronizados entre sítios diversos. De fato, a seleção de candidatos que possuem caminhos em comum é próxima de 60% dos pares encontrados como candidatos. É possível variar o grau de restrição do filtro inicial, porém mesmo que o filtro aplicado seja pouco restritivo e considere como candidatos, pares de sítios de similaridade muito baixa, o número de pares possíveis é extremamente menor que uma seleção por força bruta.

3.3.2 Avaliação do Conjunto de Pares Candidatos

O conjunto reduzido de pares candidatos obtido na primeira fase do algoritmo pode então ser refinado por um processo que utiliza um algoritmo de aprendizado de máquina para identificar a probabilidade de cada par ser um par replicado. Esse algoritmo utiliza um conjunto de novas características, chamado de conjunto de refinamento, descritas na Seção 3.2.

Neste trabalho, a tarefa de identificar a probabilidade de pares de sítios serem réplicas é modelada como um problema de classificação. Mais especificamente, um classificador é construído a partir de um conjunto de treinamento e relaciona características dos pares de sítios à probabilidade desse par ser ou não uma réplica. Esse classificador é então utilizado para identificar automaticamente quais pares são replicados em um conjunto de teste.

A principal vantagem de uma abordagem baseada em classificação é o fato de não ser preciso definir exatamente o que é uma réplica. Muitas vezes pode não ser possível identificar um par de sítios replicados apenas por uma análise da quantidade de conteúdo duplicado entre eles. Por exemplo, existem classes de sítios como os que armazenam letras musicais, que por armazenarem letras idênticas de músicas, possuem uma alta porcentagem de documentos similares, porém não podem ser caracterizados como réplicas. Por outro lado, ao ser utilizada uma abordagem de classificação, é preciso apenas fornecer uma quantidade suficiente de exemplos de treino e o classificador automaticamente aprende a diferenciar os pares que são ou não replicados, com base em diversas características e suas possíveis relações.

O método de aprendizado de máquina utilizado nesta dissertação foi o algoritmo de Classificação Associativa LAC [Veloso & Meira Jr., 2011], descrito na Seção 2.3.1. Mais especificamente, é utilizado um conjunto de treino $\mathcal{D}$ que consiste de exemplos na forma $\langle F_r, \ell \rangle$, onde F_r é um conjunto de características associadas a cada par de sítios, e $\ell \in \{\ominus, \oplus\}$ é uma variável binária que especifica se um par de sítios correspondente deve ou não ser

considerado um par replicado. O conjunto de treino é então utilizado na construção de um classificador $\mathcal{L}$ que relaciona padrões em F_r ao valor de ℓ e será responsável por calcular a probabilidade dos pares de sítios web no conjunto de candidatos $\mathcal{C}$ serem réplicas.

O classificador $\mathcal{L}_x$ é composto por um conjunto de regras de associação que são extraídas do conjunto de treino $\mathcal{D}$ conforme a definição seguinte [Veloso et al., 2006a,b; Veloso & Meira Jr., 2011; Veloso et al., 2011]:

> **Classificação baseada em regras**: Uma regra de classificação tem a forma de $\{\mathcal{F} \rightarrow \ell\}$, onde o antecedente $\mathcal{F}$ é o conjunto de características e o consequente $\ell \in \{\oplus, \ominus\}$ indica se a predição é positiva ou negativa. A cardinalidade da regra $\{\mathcal{F} \rightarrow \ell\}$ é dada pelo número de características no antecedente, ou seja $|\mathcal{F}|$. O suporte de $\mathcal{F}$ é definido como $\sigma(\mathcal{F})$ e é o número de exemplos em $\mathcal{D}$ que possuem $\mathcal{F}$ como um subconjunto. A confiança da regra $\{\mathcal{F} \rightarrow \ell\}$ é definida como $\theta(\mathcal{F} \rightarrow \ell)$ e é a probabilidade condicional de c dadas as características em $\mathcal{F}$, ou seja, $\theta(\mathcal{F} \rightarrow \ell) = \dfrac{\sigma(\mathcal{F} \cup \ell)}{\sigma(\mathcal{F})}$.

Mais especificamente, o classificador $\mathcal{L}$ é representado como um conjunto de entradas de forma $\langle$chave, propriedades$\rangle$, onde uma $chave = \{\mathcal{F}, \ell\}$ e $propriedades = \{\sigma(\mathcal{F}), \sigma(\mathcal{F} \cup \ell), \theta(F \rightarrow \ell)\}$. Cada entrada no conjunto corresponde a uma regra e a chave é usada para facilitar um acesso rápido às propriedades da regra. Uma vez que o classificador $\mathcal{L}$ é extraído de $\mathcal{D}$ as regras são coletivamente utilizadas para aproximar a probabilidade de um exemplo arbitrário ser positivo ($\oplus$) ou negativo ($\ominus$). Basicamente, $\mathcal{L}$ é interpretado como um conjunto em que cada regra $\{\mathcal{F} \rightarrow \ell\} \in \mathcal{L}$ é um voto dado por $\mathcal{F}$ para $\oplus$ ou $\ominus$. Dado um exemplo x, uma regra $\{\mathcal{F} \rightarrow \ell\}$ é considerada um voto válido apenas se for aplicável a x.

Seja $\mathcal{L}_x$ o conjunto de regras em $\mathcal{L}$ que são aplicáveis ao exemplo x, todas e apenas as regras em $\mathcal{L}_x$ são consideradas como votos válidos durante a classificação de x. Em seguida é definido $\mathcal{L}_x^\ell$ como o subconjunto de $\mathcal{L}_x$ que contém apenas regras de predição para ℓ. Os votos em $\mathcal{L}_x^\ell$ tem pesos diferentes, dependendo da confiança das regras correspondentes. Finalmente é feita a média dos pesos dos votos para ℓ, retornando uma pontuação para ℓ em relação a x (Equação 3.5). Finalmente, a probabilidade de x ser um exemplo negativo é dada pela pontuação normalizada (Equação 3.6).

$$s(x, \ell) = \sum \frac{\theta(\mathcal{F} \rightarrow \ell)}{|\mathcal{L}_x^\ell|}, \text{with } \ell \in \{\ominus, \oplus\} \tag{3.5}$$

$$\alpha(x, \ominus) = \frac{s(x, \ominus)}{s(x, \ominus) + s(x, \oplus)} \tag{3.6}$$

Para evitar o enorme espaço de busca durante o processo de extração de regras, o algoritmo LAC projeta um conjunto de treino de acordo com o exemplo a ser processado (ou

seja, pares de sítios web). Mais especificamente a extração de regras não é realizada até que um candidato x seja dado para classificação. Em seguida os valores das características de x são utilizadas como filtro que configura o conjunto de treino $\mathcal{D}$ de forma que apenas regras aplicáveis a x sejam extraídas. Esse processo, ilustrado na Tabela 3.1, produz um conjunto de treino projetado definido como $\mathcal{D}_x$, que contém apenas características que são presentes em x.

	p					ℓ
	ip4	ip3	ndist	nmatch	fullpath	
y_1	[0.1-0.3]	[0.3-0.5]	[8-10]	[0.2-0.5]	[0.3-0.5]	$\oplus$
y_2	[0.1-0.3]	[0.1-0.3]	[10-14]	[0.1-0.2]	[0.1-0.3]	$\ominus$
y_3	[0.5-0.8]	[0.1-0.3]	[8-10]	[0.1-0.2]	[0.1-0.3]	$\oplus$
y_4	[0.1-0.3]	[0.5-0.8]	[8-10]	[0.2-0.5]	[0.3-0.5]	$\oplus$
y_5	[0.3-0.5]	[0.5-0.8]	[5-8]	[0.5-0.7]	[0.3-0.5]	$\oplus$
y_6	[0.1-0.3]	[0.3-0.5]	[8-10]	[0.5-0.7]	[0.3-0.5]	$\ominus$
y_7	[0.1-0.3]	[0.1-0.3]	[10-14]	[0.1-0.2]	[0.3-0.5]	$\ominus$
y_8	[0.3-0.5]	[0.5-0.8]	[5-8]	[0.5-0.7]	[0.3-0.5]	$\oplus$
x	[0.1-0.3]	[0.1-0.3]	[8-10]	[0.2-0.5]	[0.1-0.3]	?
	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	
	ip4	ip3	ndist	nmatch	fullpath	
y_1	[0.1-0.3]	–	[8-10]	[0.2-0.5]	–	$\oplus$
y_2	[0.1-0.3]	[0.1-0.3]	–	–	[0.1-0.3]	$\ominus$
y_3	–	[0.1-0.3]	[8-10]	–	[0.1-0.3]	$\oplus$
y_4	[0.1-0.3]	–	–	[0.2-0.5]	–	$\oplus$
y_5	–	–	–	–	–	$\oplus$
y_6	[0.1-0.3]	–	–	–	–	$\ominus$
y_7	[0.1-0.3]	[0.1-0.3]	–	–	–	$\ominus$
y_8	–	–	–	–	–	$\oplus$

Tabela 3.1: Conjunto de treino $\mathcal{D} = \{y_1, y_2, \ldots, y_8\}$, e instância x. Em seguida o conjunto projetado $\mathcal{D}_x$.

O Programa 3 descreve o algoritmo para refinamento do conjunto de pares candidatos.

Programa 3 Avaliação do conjunto de pares candidatos a réplica

Entrada: conjunto C de pares de sítios candidatos a réplica.
Entrada: conjunto $\mathcal{D}$ de pares de sítios rotulados (marcados como réplicas ou não réplicas)
Saída: conjunto C de pares de sítios ordenados pela probabilidade de serem réplicas.

1: **seja** F_r o conjunto das características para refinamento
2: **para todo** par x em C **faça**
3: **seja** $\mathcal{V}_i$ o conjunto dos valores das características de x em F_r
4: projete $\mathcal{D}$ com base em V_i e x, para criar D_x
5: treine um classificador $\mathcal{L}_x$ com base no conjunto D_x
6: obtenha do classificador $\mathcal{L}_x$ a probabilidade de x ser réplica
7: Ordene os pares em C pela pontuação dada pelo classificador de acordo com a chance de o par ser replicado

Um passo crucial do método proposto é a construção do conjunto de treinamento $\mathcal{D}$ para o classificador. Devido a enorme quantidade de pares de sítios presentes em bases de

máquinas de busca, o processo de criação de uma base de treino é considerado um verdadeiro gargalo. Uma vez que seria preciso a avaliação humana de milhares de pares de sítios, principalmente em busca de exemplos positivos que são desproporcionalmente mais escassos. Mesmo a partir de uma seleção previa de sítios candidatos, o custo de uma inspeção manual em busca de exemplos positivos pode tornar inviável a seleção de grandes quantidades de exemplos. Em muitos casos, porém, certos tipos de exemplos podem ser obtidos sem esforço. Portanto, é proposta uma estratégia para criação automática do conjunto de treino $\mathcal{D}$, que será descrita na próxima seção.

3.3.3 Criação Automática do Conjunto de Treino

Nesta seção é detalhada a abordagem desenvolvida para criação automática do conjunto de treino $\mathcal{D}$ utilizado no treinamento do algoritmo de classificação de sítios replicados.

A utilização de métodos de classificação neste trabalho está sujeita ao gargalo de aquisição de dados, uma vez que a definição de exemplos de treino requer uma rotulação manual de pares de sítios web, definidos como réplicas ou não. O custo associado a este processo de anotação pode tornar inviável a aquisição de grandes quantidades de exemplos de treino.

Em muitos casos, porém, encontrar certos tipos de exemplos pode ser bastante fácil. Por exemplo, podemos atribuir um rótulo positivo para o caso de réplicas detectadas de maneira mais óbvia, pares de sítios cuja única diferença está na utilização ou não do prefixo WWW. De maneira similar, podemos considerar exemplos negativos de réplicas, aqueles pares que não são nem ao menos suspeitos de serem réplicas, ou seja, não compartilham nenhuma página com mesmo conteúdo ou caminho de URL. Entretanto, construir um conjunto de treino para um classificador diretamente desse tipo de exemplos pode levar a uma baixa qualidade na detecção de réplicas, uma vez que pode existir uma enorme quantidade de falsos-negativos nesse conjunto de treino.

Este trabalho propõe estratégias baseadas no algoritmo EM (*"Expectation-Maximization"*) [Dempster et al., 1977] para produzir um conjunto de treino de melhor qualidade.

A primeira abordagem proposta, a qual chamaremos de PU, faz uso de exemplos positivos óbvios para construção do treino conforme o seguinte cenário:

> **PU**: um conjunto de treino $\mathcal{D}$ é composto por um conjunto de exemplos *potencialmente* positivos P e um grande conjunto de exemplos não rotulados U. Os dados não rotulados contidos em U são então considerados negativos. Portanto $\mathcal{D}$ pode conter falsos negativos.

A seleção de pares que formam o conjunto P é realizada a partir de características mais óbvias e fáceis de se computar em relação ao conjunto de candidatos C obtido na fase de seleção de pares candidatos (Seção 3.3.1). São elas:

- Pares de sítios cuja diferença está apenas em ter ou não o prefixo "WWW".
  Ex: www.exemplo.com.br / exemplo.com.br

- Sítios que possuam diferenças apenas no domínio superior.
  Ex: exemplo.gov.br / exemplo.br

O conjunto U é formado pelo restante de pares candidatos, sendo $P \cap U \equiv \emptyset$. O método proposto baseado em EM realiza um processo de classificação que utiliza o algoritmo LAC para atribuir a cada par $x \in D$ uma probabilidade $\alpha(x, \ominus)$ de ser negativo. Uma vez que essas probabilidades foram calculadas, são realizadas transições de rótulos $x^{(\ominus \to \oplus)}$, fazendo com que pares previamente rotulados como negativos sejam atualizados como pares positivos. Ao final do processo EM é esperada uma convergência dos rótulos assinalados para uma combinação que seja mais próxima da realidade dos dados.

Uma questão importante que afeta o desempenho dessa abordagem é a decisão de quando realizar uma operação de transição de rótulos. Nesse caso, um limiar de transição α_{min} determina que uma operação $x^{(\ominus \to \oplus)}$ é realizada sempre quando x é um exemplo negativo e $\alpha(x, \ominus) \leq \alpha_{min}$. O valor ótimo de α_{min} não é previamente conhecido, sendo a definição desse limiar é uma peça fundamental do algoritmo.

A primeira estratégia para escolha do melhor limiar de corte consiste em alternar valores diferentes para α_{min} e avaliar quais obtêm melhores resultados. Uma segunda estratégia consiste em encontrar automaticamente o melhor limiar α^x_{min} para cada instância do modelo, ao invés de aplicar um único valor α_{min} global a todas instâncias. Para isso foi utilizado um método de minimização de entropias [Davis et al., 2012] que encontra o limiar de transição α^x_{min} e possui o seguinte cenário:

> **Escolha de Limiar**: Seja D_x o conjunto de treino induzido por um exemplo y e $\ell^y \in \{\oplus, \ominus\}$ a classe associada ao exemplo $y \in D_x$. Considere $N_\ominus(D_x)$ o número de exemplos em D_x em que $\ell^y = \ominus$. De maneira análoga, considere $N_\oplus(D_x)$ o numero de exemplos D_x em que $\ell^y = \oplus$. O método proposto busca por um limiar α^x_{min} que provê um particionamento de melhor entropia no espaço de probabilidades induzido por D_x.

A entropia de um conjunto pode ser definida como uma medida do grau de impureza do conjunto, sendo máxima quando existem tantos elementos positivos quanto negativos, e mínima quando todos os elementos são da mesma classe. É importante notar que o cálculo

da melhor entropia e, por conseguinte a escolha do melhor limiar de corte, é feito com base no conjunto de treino D_x induzido a cada instância de treino. Ou seja o limiar escolhido é local e específico para cada modelo.

Mais especificamente, dados exemplos de treino $\{y_1, y_2, \ldots, y_k, \}$ em D_x, o método primeiramente calcula $\alpha(x, \ominus)$ para todo $y_i \in D_x$. Então, os valores de $\alpha(x, \ominus)$ são ordenados em ordem crescente. De um modo ideal, existe um corte α^x_{min} tal que:

$$\ell^{y_i} = \begin{cases} \oplus & \text{Se } \alpha(y_i, \ominus) \leq \alpha^x_{min} \\ \ominus & \text{caso contrário} \end{cases}$$

Entretanto, existem casos mais difíceis, para os quais não é possível encontrar uma separação perfeita no espaço de probabilidades. Portanto, é aplicado um método mais geral para encontrar o melhor corte, ou separação do espaço. A ideia básica é a de que qualquer valor para α^x_{min} induz duas partições sobre o espaço de valores para $\alpha(x, \ominus)$, ou seja, uma partição para valores abaixo de α^x_{min} e uma para valores acima de α^x_{min}. O algoritmo atribui à α^x_{min} o valor que minimiza a entropia dessas duas partições. Uma vez que α^x_{min} é calculado, ele pode ser usado para ativação das operações de transição de rótulos das classes entre réplicas e não réplicas. Em seguida serão apresentadas as definições básicas que detalham esse algoritmo.

> **Minimização de Entropia**: Considere uma lista $\mathcal{O} = \{\ldots, <\ell^{y_i}, \alpha(y_i, \ominus)>, \ldots, <\ell^{y_j}, \alpha(y_j, \ominus)>, \ldots\}$, tal que $\alpha(y_i, \ominus) \leq \alpha(y_j, \ominus)$. Seja f um valor candidato para α^x_{min}. Nesse caso, $\mathcal{O}_f(\leq)$ é uma sub-lista de $\mathcal{O}$, ou seja, $\mathcal{O}_f(\leq) = \{\ldots, <\ell^y, \alpha(y, \ominus)>, \ldots\}$, tal que para todos elementos em $\mathcal{O}_f(\leq)$, $\alpha(y, \ominus) \leq f$. De modo similar, $\mathcal{O}_f(>) = \{\ldots, <\ell^y, \alpha(y, \ominus)>, \ldots\}$, tal que para todos elementos em $\mathcal{O}_f(>)$, $\alpha(y, \ominus) > f$. Sendo assim, $\mathcal{O}_f(\leq)$ e $\mathcal{O}_f(>)$ são partições de $\mathcal{O}$ induzidas por f.

Primeiramente o algoritmo calcula a entropia em $\mathcal{O}$ conforme a Equação 3.7. Em seguida é calculada a soma das entropias em cada partição induzida por f, de acordo com a Equação 3.8.

$$\begin{aligned} E(\mathcal{O}) = \; & - \left(\frac{\mathrm{N}_\ominus(\mathcal{O})}{|\mathcal{O}|} \times \log \frac{\mathrm{N}_\ominus(\mathcal{O})}{|\mathcal{O}|}\right) \\ & - \left(\frac{\mathrm{N}_\oplus(\mathcal{O})}{|\mathcal{O}|} \times \log \frac{\mathrm{N}_\oplus(\mathcal{O})}{|\mathcal{O}|}\right) \end{aligned} \tag{3.7}$$

$$E(\mathcal{O}_f) = \frac{|\mathcal{O}_f(\leq)|}{|\mathcal{O}|} \times E(\mathcal{O}_f(\leq)) + \frac{|\mathcal{O}_f(>)|}{|\mathcal{O}|} \times E(\mathcal{O}_f(>)) \tag{3.8}$$

Finalmente α^x_{min} recebe o valor de f que minimiza $E(\mathcal{O}) - E(\mathcal{O}_f)$, conforme a Figura 3.3.

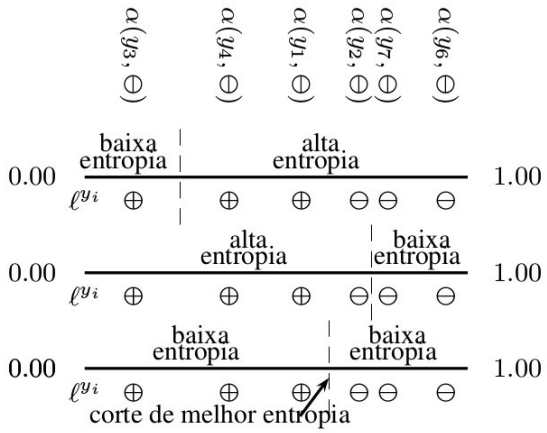

Figura 3.3: Melhor ponto de corte (entropia mínima).

Outra abordagem proposta para criação automática de conjunto de treino é referente ao uso de exemplos que sabidamente não são réplicas. Esses exemplos são fáceis de se conseguir, uma vez que milhares deles são descartados na fase inicial do algoritmo. Mais especificamente é utilizado um conjunto N de pares descartados como suspeitos na filtragem de candidatos. Nessa abordagem é proposta uma nova versão de algoritmo EM com o seguinte cenário:

> **NU**: Um conjunto de treino $\mathcal{D}$ é composto por um conjunto de exemplos negativos N e um grande conjunto de exemplos não rotulados U. Neste cenário os dados contidos em U são considerados positivos, contendo assim falsos positivos.

Similar ao processo em PU esta abordagem aplica operações $x^{(\oplus \to \ominus)}$ aos rótulos das instâncias em $\mathcal{D}$. Uma transição de rótulo é realizada sempre que x é um exemplo positivo e $\alpha(x, \oplus) \leq \alpha^x_{min}$.

Um desafio particular é que o algoritmo proposto realiza diversas transições de rótulos durante o processo EM e cada operação de transião modifica $\mathcal{D}$ e invalida parte do classificador atual $\mathcal{L}$, que precisa ser propriamente atualizado. Nesse caso, o algoritmo LAC permite

manter $\mathcal{L}$ atualizado de maneira incremental [Silva et al., 2011; Lourenco Jr. et al., 2014] de forma que o classificador atualizado seja o mesmo que seria obtido caso fosse totalmente reconstruído do zero a cada passo. Sendo assim temos o seguinte cenário:

> **Atualização Incremental:** Uma operação de transição de rótulos $x^{\ominus\to\oplus}$ (ou $x^{\oplus\to\ominus}$) não muda o valor de $\sigma(\mathcal{F})$ de nenhum conjunto de características $\mathcal{F}$. Mais especificamente a operação $x^{\ominus\to\oplus}$ muda apenas o rótulo associado a x e não muda os valores das características de x. Portanto, o número de exemplos em $\mathcal{D}$ que possuem o subconjunto $\mathcal{F}$ de características é essencialmente o mesmo em $\{(\mathcal{D} - x^{\ominus}) \cup x^{\oplus}\}$. O mesmo acontece em operações $x^{\oplus\to\ominus}$. Além disso todas as regras em $\mathcal{L}$ que devem ser atualizadas devido a $x^{\ominus\to\oplus}$ (or $x^{\oplus\to\ominus}$) são as contidas no classificador $\mathcal{L}_x$ específico para a instância x.

A atualização incremental do classificador durante o processo EM faz com que o processo seja realizado de maneira muito mais eficiente. A comparação de desempenho entre uma atualização incremental e uma atualização completa do classificador a cada iteração do EM será discutida na Seção 4.6.

O Programa 4 descreve o algoritmo para construção do treino $\mathcal{D}$.

Programa 4 Construção automática do conjunto de treino $\mathcal{D}$

Entrada: conjunto inicial $\mathcal{D}$ com exemplos de treino óbvios
Saída: conjunto $\mathcal{D}$ completo com exemplos não óbvios.

```
1: para todo par x em D faça
2:     Treine um classificador L_x com base no conjunto D
3:     Defina um limiar mínimo de réplica α^x_min
4: → Expectation:
5:     se (PU) então
6:         se (α(x, ⊖) ≤ α^x_min) então
7:             realize operações x^{⊖→⊕}
8:     senão se (NU) então
9:         se (α(x, ⊖) > α^x_min) então
10:            realize operações x^{⊕→⊖}
11:    atualize D apropriadamente
12: → Maximization:
13:    atualize L_x ⊆ L e α(x, ⊖)
```

3.3.4 Combinação de classificadores

Os métodos PU e NU são suficientes para geração de coleções de treino a serem utilizadas na classificação automática de pares de sítios replicados. Porém, é possível melhorar os resultados do processo de classificação proposto, utilizando um conceito da Economia denominado Eficiência de Pareto [Palda, 2011]. Esse método é utilizado para agregar resultados

gerados a partir das abordagens PU e NU. Mais especificamente, a agregação de resultados é dada como eficiente se a predição de determinado classificador, ao ser privilegiada, pode causar danos à predição de um outro classificador [Moreira et al., 2014]. A Eficiência de Pareto é relacionada a noção de dominância no espaço induzido pelas predições de ambos classificadores.

Esta dissertação propõe então a abordagem denominada PNU, baseada na Eficiência de Pareto, que combina resultados de PU e NU com o seguinte cenário:

> **PNU**: Seja $\alpha^P(x, \oplus)$ a probabilidade de um par x ser réplica de acordo com o classificador construído por PU. Analogamente, seja $\alpha^N(x, \oplus)$ a probabilidade de par x ser réplica de acordo com o classificador construído por NU. Cada candidato $x \in C$ é inserido em um espaço bidimensional, com coordenadas $\alpha^P(x, \oplus)$ e $\alpha^N(x, \oplus)$ (Figura 2.3). Os pares de sítios replicados são selecionados sob a curva que determina o conjunto de soluções ótimas nessas duas dimensões e caracteriza a Fronteira de Pareto.

Como demonstrado na Figura 3.4, o candidato a domina o candidato b, se e somente se, ambas condições forem mantidas:

- $\alpha^P(a, \oplus) \geq \alpha^P(b, \oplus)$ and $\alpha^N(a, \oplus) \geq \alpha^N(b, \oplus)$
- $\alpha^P(a, \oplus) > \alpha^P(b, \oplus)$ or $\alpha^N(a, \oplus) > \alpha^N(b, \oplus)$

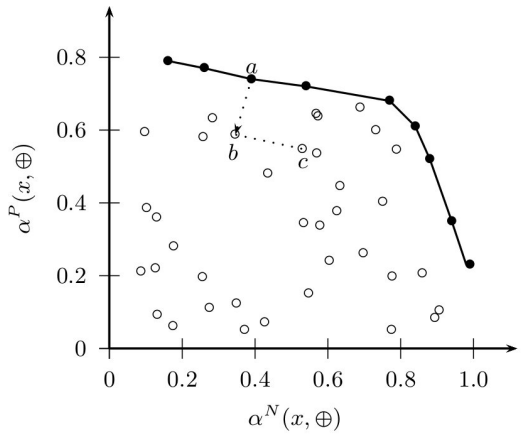

Figura 3.4: Nem b ou c são dominados entre si, porém a domina b. Pontos não dominados por nenhum outro ponto formam a Fronteira de Pareto.

O operador de dominância relaciona dois candidatos a réplica onde o resultado da operação tem duas possibilidades: (i) um candidato domina outro candidato ou (ii) dois candidatos não são dominados entre si. Portanto, a Fronteira de Pareto P é composta por

todos candidatos a réplica que não são dominados por nenhum candidato. Mais especificamente, a fronteira P é formada pela lista de k candidatos a réplica $\mathcal{P} = \{x1, x2, \ldots, k\}$ de forma que não exista nenhum par (xi, xj) onde x domina xj, como mostra a Figura 3.4.

A Fronteira de Pareto contém ou candidatos que se destacam em um classificador ou candidatos que possuem um balanceamento adequado entre os dois classificadores. O próximo passo envolve ordenar os candidatos de acordo com a probabilidade de serem réplicas.

Seja $dom(x)$ o número de candidatos que são dominados pelo candidato x. O *ranking* final é uma lista ordenada $\{x1, x2, x3\}$ de forma que não exista nenhum par (xi, xj) onde $dom(xi) > dom(xj)$, dado $i > j$. Sendo assim os pares mais dominantes aparecem no topo *ranking*

O Programa 5 descreve o algoritmo para agregação de resultados que constitui o método PNU.

Programa 5 PNU: Agregação de resultados (de PU e NU) via Fronteira de Pareto

Entrada: os espaços de predições $\alpha^P(x, \oplus)$ e $\alpha^N(x, \oplus)$ provenientes de PU e NU, respectivamente.
Saída: conjunto C de pares de sítios ordenados pela probabilidade de serem réplicas.

1: Encontre a Fronteira de Pareto $\mathcal{P} = \{x1, x2, \ldots, x_k\}$ para os pares de sítios x no espaço de predições $\alpha^P(x, \oplus)$ e $\alpha^N(x, \oplus)$
2: **para todo** par x em C **faça**
3: calcule $dom(x)$
4: Ordene os pares em C em ordem decrescente de acordo com $dom(x)$

A Figura 3.5 oferece uma visualização completa dos processos que envolvem método proposto.

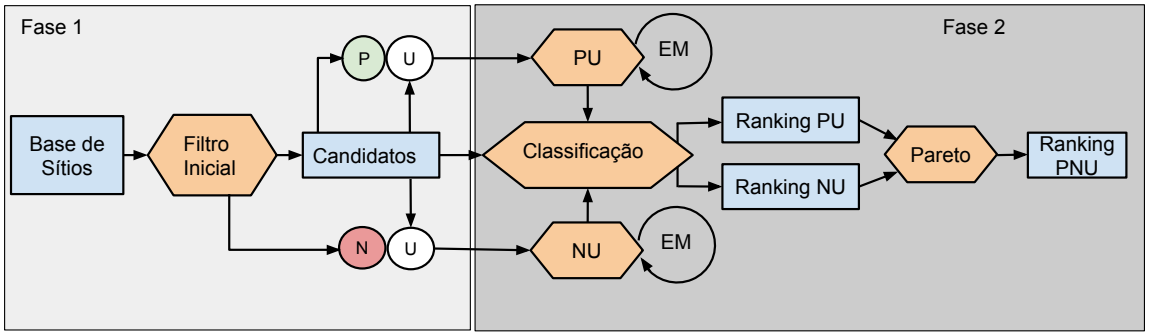

Figura 3.5: Descrição dos processos do método proposto

Capítulo 4

Resultados Experimentais

Esta seção apresenta a coleção de dados utilizada neste trabalho e os resultados experimentais para o algoritmo proposto. As estratégias propostas são comparadas a outras abordagens presentes na literatura. Para isso são descritas e utilizadas diferentes métricas de avaliação dos algoritmos, como taxas de verdadeiros e falsos positivos, precisão e revocação, taxa de redução, desempenho em detecção e tempo de execução. Em resumo serão avaliadas as seguintes hipóteses:

- *Distribuição de duplicatas intrassítios e intersítios na coleção*: Qual a correlação entre os diferentes tipos de duplicatas e o tamanho dos sítios. Réplicas geralmente aparecem em pares ou em grupos maiores de sítios idênticos? (Seção 4.1).

- *Relação entre a similaridade textual e os sítios duplicados:* A similaridade sozinha é uma boa medida para separar réplicas de não réplicas? (Seção 4.2).

- *Porcentagem de réplicas detectadas em relação ao número de não réplicas selecionadas por engano*: A análise ROC mostra uma área sob a curva de mais alta que 0.98, indicando que o algoritmo proposto detecta todas réplicas de sítios na coleção à uma baixa taxa de erro ou falsos-positivos (Seção 4.3).

- *Eficácia do algoritmo*: O algoritmo proposto é capaz de detectar casos de réplicas infiltrados em 1000 candidatos à réplicas com probabilidade maior que 99% (Seção 4.4).

- *Impacto da detecção de réplicas de sítios em relação ao número de URLs duplicadas*: O algoritmo proposto reduz em 19% o número de URLs duplicadas na coleção a uma taxa de apenas 0.005 em falsos positivos. Além disso, a combinação do algoritmo proposto com algoritmos de detecção intrassítios pode aumentar a taxa de redução em mais de 21% (Seção 4.5).

- *A viabilidade da solução em relação ao tempo de execução do algoritmo*: O tempo de execução cai em três ordens de magnitude ao atualizar os classificadores de maneira incremental durante o processo EM. O tempo necessário para avaliar um candidato a réplica não é maior que 0.1 segundos (Seção 4.6).

4.1 Coleção

A coleção de dados utilizada neste trabalho foi obtida através da coleta de páginas web, utilizando um coletor web real, que integra uma máquina de busca desenvolvida no Instituto Nacional de Ciência e Tecnologia para a Web (InWeb). O conjunto utilizado corresponde a mais de 30 milhões de páginas web que se restringem a sítios brasileiros. Essa coleta foi realizada entre Setembro e Outubro de 2010, e não houve nenhuma restrição ou filtro quanto à qualidade ou conteúdo duplicado. Portanto, essa coleção é um retrato fiel da Web. O conjunto de dados final corresponde a 583,411 sítios web, sendo assim aproximadamente 2×10^{11} pares possíveis, sem nenhuma atribuição de classes manual ou semi-automática.

Uma vez que a proposta deste trabalho consiste em identificar sítios replicados, pares de sítios que obviamente não formam candidatos a réplicas são descartados, como será discutido em seguida.

Dup-clusters A seleção de pares candidatos a réplica, conforme a Seção 3.3.1, é feita a partir do descarte de pares de sítios que não possuem nenhuma característica básica em comum. Para que seja possível caracterizar a coleção de sítios presente na base de dados e estudar as distribuições de conteúdo duplicado entre sítios, são criados os chamados dup-clusters [Agarwal et al., 2009; Yang & Callan, 2006]. Páginas web que possuem conteúdo idêntico são agrupadas em um mesmo (dup-)cluster, conforme mostra a Figura 4.1. Sítios diferentes que possuem URLs em um mesmo dup-cluster são tratados como candidatos a réplicas.

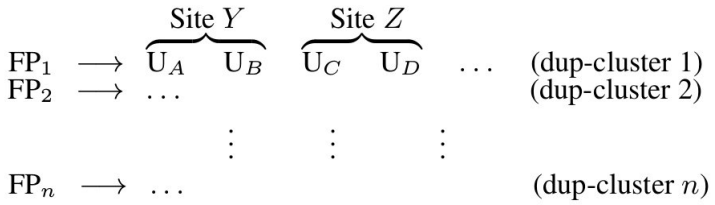

Figura 4.1: Descrição de um dup-cluster

Cada FP_i é uma assinatura de conteúdo e cada U_j é uma URL. URLs duplicadas são aquelas contidas em um mesmo dup-cluster. URLs em um dup-cluster que pertencem a um

mesmo sítio web são consideradas duplicatas intrassítios (U_A e U_B), enquanto aquelas que pertencem a sítios diferentes são consideradas duplicatas intersítios (U_A e U_C).

Especificamente foram obtidos 172.004 sítios web que compartilham pelo menos uma assinatura com outro sítio web, resultando em 111 milhões de candidatos a réplica. Desse conjunto foram separados 50 mil casos óbvios de serem réplicas utilizados como exemplos positivos e 800 mil pares de sítios que não compartilham nenhum conteúdo em comum, utilizados como exemplos negativos.

O tamanho de um dup-cluster é dado pelo número de URLs no dup-cluster. A Figura 4.2 (esquerda) mostra a distribuição de tamanhos dos dupclusters formados. Claramente, muitos dup-clusters contêm poucas URLs e poucos dup-clusters contêm muitos milhares de URLs. A Figura 4.2 (direita) mostra o número de dup-clusters em que um sítio aparece. Mais uma vez, poucos sítios estão presentes em muitos dup-clusters enquanto a maior parte dos sítios estão contidos em poucos dup-clusters.

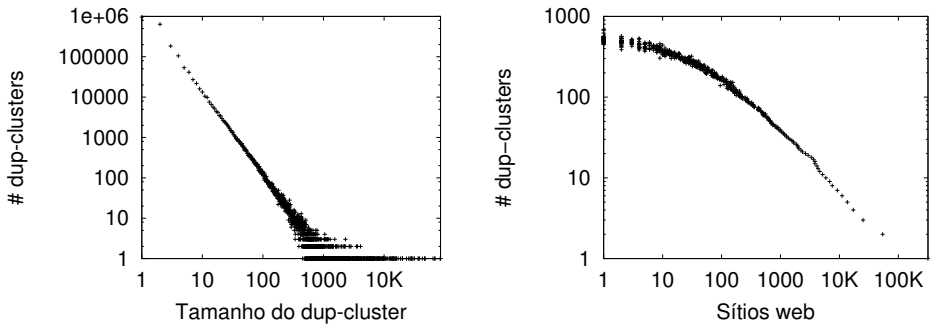

Figura 4.2: Esquerda: Distribuição de tamanho dos dup-clusters. Direita: Distribuição de sítios por dup-cluster

Distribuição de réplicas intersítios e intrassítios A Figura 4.3 mostra a relação entre o número de réplicas intrassítios e intersítios em cada sítio web. A curva de correlação mostra que um crescimento de três vezes para dez vezes em relação ao número de URLs duplicadas intersítios e URLs duplicadas intrassítios, respectivamente.

Construção do gabarito A tarefa de definir se um par de sítios é ou não uma réplica é muito difícil devido a (i) muitos sítios duplicados possuírem conteúdo dinâmico, onde mesmo que determinada URL seja coletada duas vezes em seguida, o conteúdo coletado irá diferir, e (ii) muitos sítios web podem conter páginas de conteúdo idêntico ou similar e ainda assim não serem réplicas.

Portanto, para possibilitar a avaliação dos classificadores propostos, foram selecionamos aleatoriamente um conjunto de 1.600.000 pares de sítios do conjunto de 111 milhões

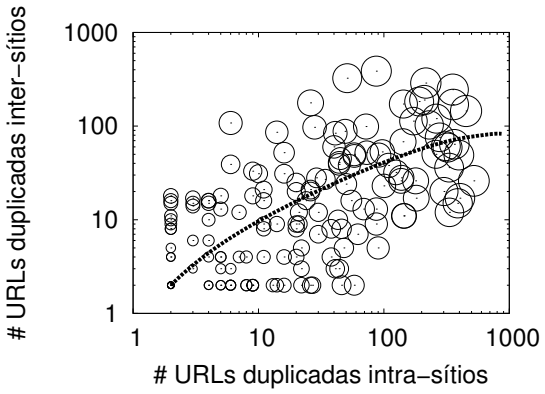

Figura 4.3: Correlação entre o número de URLs duplicadas intrassítios e intersítios por sítio web. Cada ponto corresponde a um sítio web e seu tamanho indica o número de URLs dentro do sítio. Esta figura é uma amostragem de 200 sítios

de pares possíveis de candidatos a serem réplicas. A partir desses pares, foram classificados manualmente 6.823 pares como sítios replicados. Do grupo de sítios selecionados como réplicas, 354 são casos óbvios que consistem em versões com ou sem prefixo WWW ou pequenas variações de domínio (".com", ".gov", ".net", etc.). Os 6.469, pares restantes, são casos não óbvios, envolvendo variações maiores quanto ao nome dos servidores. No conjunto de 1.600.000 pares de sítios existem 49.636 sítios, sendo 3.765 sítios que aparecem em pelo menos um dos 6.823 pares de réplica.

A Figura 4.4 mostra a distribuição do tamanho de sítios replicados e não replicados. Sítios replicados têm em média 224 URLs, enquanto sítios não replicados possuem uma média de 135 URLs.

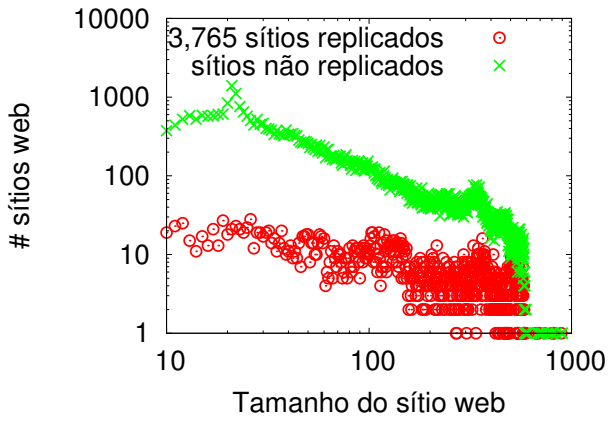

Figura 4.4: Distribuição de tamanho de sítios replicados e não replicados

A Figura 4.5 mostra a fração de sítios que aparecem em pelo menos x casos de réplicas. Como pode ser notado, a maioria dos sítios replicados aparece em poucos casos de réplicas. A eliminação de todos 3.765 sítios replicados pode levar a uma redução de 12.1%

de conteúdo duplicado (843.526 URLs seriam removidas de um total de 6.948.501 URLs duplicadas). Vale notar que todas URLs pertencentes a um mesmo dup-cluster foram consideradas URLs duplicadas. Outro fator importante é que devido ao enorme número de pares candidatos, a inexistência de falsos falsos-negativos não é garantida.

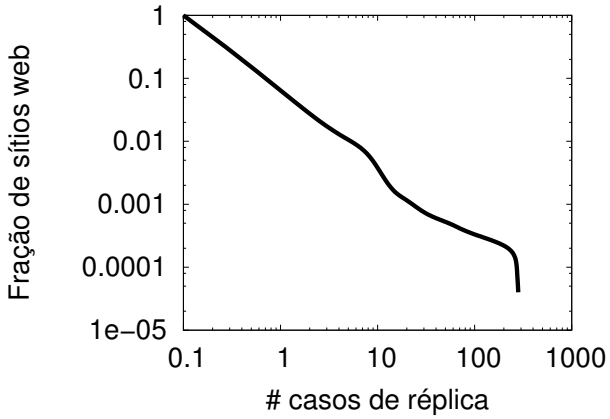

Figura 4.5: Fração de sítios que aparecem em pelo menos x casos de réplicas

Distribuição de similaridade entre sítios A similaridade entre sítios A e B é dada em termos do numero de dup-clusters que contêm ambos os sítios. Especificamente, seja C_A o conjunto de dup-clusters contendo o sítio A, a similaridade entre A e B é dada pelo Coeficiente de Jaccard $\frac{C_A \cap C_B}{C_A \cup C_B}$. A Figura 4.6 mostra a distribuição de similaridade, que representa a fração de pares com similaridade acima de x. Também é apresentada a concentração de réplicas, que representa a fração de pares replicados cuja similaridade é acima de x. É possível notar que, ainda que sítios altamente similares tendem a ser de fato um par replicado, a similaridade sozinha não é suficiente para separar completamente replicas de não réplicas. Alguns casos de sítios duplicados envolvem sítios que não são altamente similares.

4.2 Metodologia de Avaliação

Os experimentos foram realizados a partir da técnica de validação cruzada. A base de dados de 1.600.000 pares candidatos foi dividida aleatoriamente em 2-folds, onde a cada rodada de testes, um fold foi utilizado como treino e outro como teste. Cada conjunto utilizado como treinamento foi individualmente dividido em 5-folds para realização de aprendizado automático via Maximização de Expectativas. Os resultados reportados são a média das execuções, sendo utilizado o teste estatístico Wilcoxon Signed-rank Test [Wilcoxon, 1945] para determinar se a diferença em desempenho foi estatisticamente significante. Em todos os casos foram reportadas apenas as conclusões a partir de resultados que foram significantes a

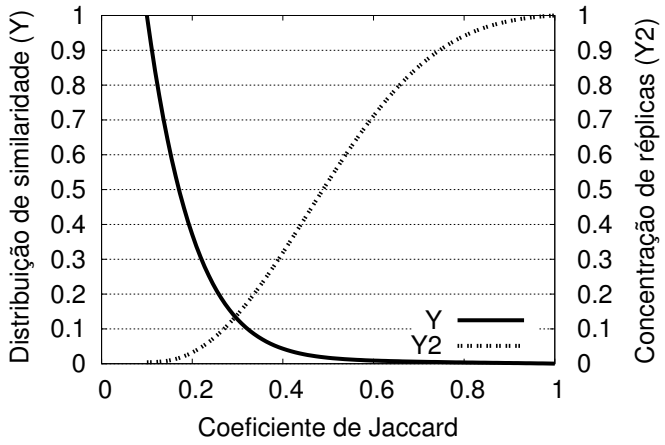

Figura 4.6: Distribuição de similaridade e concentração de réplicas

um nível de pelo menos 5%. Em cada rodada de treino e teste, foi adicionado o mesmo conjunto de 50 mil casos óbvios de réplicas ao fold de treino com dados PU e os mesmos 800 mil casos óbvios de não réplicas para cada fold treino com dados NU, conforme a Seção 3.3.3.

Métricas de avaliação As métricas de avaliação utilizadas neste trabalho foram as seguintes:

i) *Precisão*: Fração de réplicas que são corretamente detectadas.

ii) *Revocação ou Taxa de Verdadeiros Positivos (TPR)*: Corresponde à fração de replicas que são removidas da coleção.

iii) *Taxa de Falsos Positivos (FPR)*: Corresponde à fração de não replicas que são removidas da coleção.

iv) *Taxa de Redução (RR)*: Sendo U o conjunto de URLs duplicadas na coleção original e U^* o subconjunto de U que é obtido depois da remoção de sítios replicados, temos que a taxa de redução é dada por $\frac{|U| - |U^*|}{|U|}$.

v) *Taxa de Detecção de Réplicas (RDR)*: Seja I a lista de $k+1$ candidatos a réplica, onde exatamente um par é conhecido como réplica e os k pares restantes são aleatoriamente selecionados. Seja i a posição do par replicado depois de ordenarmos todos $k+1$ pares em ordem decrescente de $\alpha(x, \ominus)$. Seja n o número de pares replicados de teste, o processo é repetido n vezes e a taxa de detecção de réplicas é dada por $\frac{1}{n}\sum^{n}\frac{1}{i}$. Nesse caso foram selecionados do gabarito aleatoriamente 100 pares replicados.

Baselines

- NormPaths: O algoritmo *NormPaths* [da Costa Carvalho et al., 2007] foi utilizado como baseline neste trabalho. Esse método obtém uma pontuação entre um par de sítios, o qual indica possibilidade de os sítios do par serem réplicas. Essa pontuação é chamada de *rsim*. Quanto maior o valor dessa característica, maiores as chances de replicação. Para a computação desse atributo são extraídos 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.

  Dados dois sítios A e B da base da máquina de busca, o *rsim* entre eles é dado pela Equação 4.1, onde SL é o conjunto de todas as listas L criadas e $|L|$ é número de sítios na lista L

$$\text{rsim}(A, B) = \sum_{\forall L \in SL | A \in L, B \in L} \frac{1}{|L|}. \tag{4.1}$$

  Candidatos a réplica são ordenados de forma decrescente de acordo com $rsim(A, B)$, e os primeiros n candidatos são considerados réplicas.

- B-SVM: O algoritmo B-SVM (*Biased SVM*) [Liu et al., 2003] é considerado estado-da-arte em aprendizado semissupervisionado sob dados PU. O B-SVM utiliza uma margem SVM como classificador. Esse classificador é inteiramente reconstruído a cada iteração EM. Além disso é aplicado um único limiar de transição α_{min} para o conjunto inteiro de dados.

- Limite Superior (Upper bound): Foi considerado como limite superior para o desempenho em detecção de réplicas o resultado obtido por um classificador treinado com um modelo correto de treino, ou seja, um treinamento feito com instâncias previamente rotuladas. Esses resultados são comparados aos resultados obtidos por um classificador que utiliza o modelo de treino construído pelos algoritmos baseados em EM.

4.3 Melhor Limiar para Definição de Réplicas

Nesta seção, são apresentados dois grupos de resultados. A diferença entre eles está na escolha do parâmetro utilizado pelo algoritmo no momento de escolher um limiar mínimo para transição de rótulos entre réplicas e não réplicas. Os próximos resultados avaliam a utilização de diversos valores diferentes em busca de um limiar único que obtenha bons resultados.

Limiar Fixo para Todas Instâncias Este conjunto de experimentos consiste em avaliar o desempenho do método proposto para diferentes valores de corte, ou seja, um limiar mínimo que define uma réplica durante as iterações do modelo EM, conforme a Seção 3.3.3. O desempenho do método proposto é avaliado através da curva ROC (*Receiver Operating Characteristic*). Essa é uma plotagem gráfica para a taxa de verdadeiros positivos e falsos positivos obtidos pelo classificador quando o grau de certeza é variado. Para realizar uma síntese estatística da curva ROC, será utilizada a área sob a curva ROC ou AUC (*Area Under the ROC Curve*). A curva AUC representa a probabilidade em que um classificador constrói um *ranking* onde uma instância positiva aleatoriamente escolhida é inserida acima de uma instância negativa. A Tabela 4.1 mostra os resultados em termos de AUC, associada à cada limiar avaliado entre 0.1 e 0.9.

α_{min}	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
PU	0.7559	0.7594	0.7689	0.7643	0.9455	**0.9683**	0.9260	0.9351	0.9050
NU	0.3642	0.3632	0.3689	0.9281	**0.9482**	0.9413	0.3643	0.3521	0.3776

Tabela 4.1: Desempenho de diferentes valores para α_{min}

A tabela mostra que os limiares que proporcionaram o melhor desempenho do algoritmo foram 0.6 para o método PU e 0.5 para NU. Portanto, os melhores resultados foram obtidos com o uso de pontos de corte medianos. Uma possível justificativa é a de que limiares menores por serem muito tolerantes aumentam as chances de serem definidos falsos-positivos. De maneira análoga, limiares maiores e mais restritivos favorecem o aparecimento de falsos negativos.

O próximo conjunto de experimentos consiste em avaliar as características propostas na Seção 3.2. Uma vez que seria inviável realizar um estudo envolvendo todas combinações de características possíveis, foram considerados dois cenários: (i) cada característica é utilizada isoladamente para representar os candidatos a réplica, e (ii) todas características são utilizadas para representar candidatos a réplica, exceto aquela em que se pretende avaliar.

Conforme a Tabela 4.2, em se tratando de características utilizadas individualmente, as melhores foram *ndist* e *fullpath*, utilizando a abordagem PU, e *nmatch* e *fullpath* para abordagem NU. As piores características foram *ip3* e *ip4* quando utilizadas isoladamente em PU, enquanto *ndist* foi a pior em NU.

Na maioria dos casos, quando apenas uma característica é descartada a abordagem PU obteve resultados próximos aos alcançados com o uso de todas as características. Já a abordagem NU obteve quedas de desempenho mais significativas quando qualquer característica é eliminada. Quando a característica *ndist* foi descartada os resultados demonstraram a maior queda em desempenho para PU. O mesmo acontece quando a característica *fullpath* é des-

	AUC (Area Under the Curve)			
	Individualmente		Todas exceto	
Características	PU	NU	PU	NU
ip3	0.5033	0.5114	0.9455	0.8166
ip4	0.5244	0.5218	0.9451	0.7036
nmatch	0.5426	0.5729	0.9301	0.6178
ndist	0.7450	0.5031	0.9183	0.7163
fullpath	0.6311	0.6304	0.9300	0.5845
$Todas$	0.9683	0.9482	0.9683	0.9482

Tabela 4.2: Desempenho de diferentes combinações de características

cartada em NU. É possível notar uma equivalência entre a melhor característica individual e a queda de desempenho quando esta é retirada do conjunto completo de características.

A Figura 4.7 mostra a análise ROC para a avaliação dos métodos propostos e baselines. O algoritmo *normpaths* obteve a pior relação entre as taxas de verdadeiros positivos (TPR) e falsos positivos (FPR), alcançando um número de 0.9415 em termos de AUC. Os classificadores construídos através da abordagem PU e NU tiveram um desempenho de 0.9683 e 0.9482, respectivamente. A combinação de ambos classificadores utilizando a agregação proposta via Eficiência de Pareto, resultou em uma melhoria que chegou a 0.9763, o que ainda está distante do valor de 0.9956 atingido pelo limite superior.

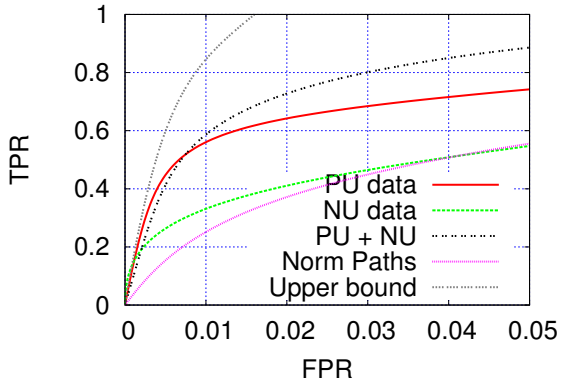

Figura 4.7: Análise da curva ROC

A Figura 4.8 (esquerda) mostra a relação entre a taxa de redução e a taxa de falsos positivos. Claramente todos algoritmos mostram um desempenho similar se um grande numero em FPR é permitido. No entanto, é importante considerar a redução obtida ao custo de uma taxa bem pequena de FPR. Portanto, a Figura 4.8 (direita) mostra a mesma relação considerando um valor máximo de 0.005 em FPR. Nesta escala a qualidade dos resultados dos diferentes algoritmos é bem diferente. Especificamente, o *normpaths* atinge os piores resultados, fornecendo uma redução de 14% com uma taxa de FPR de 0.005. Nesse mesmo

nível de precisão, classificadores aprendidos a partir dos modelos de PU e NU conseguiram uma redução de 17% e $15,5\%$, respectivamente. A combinação das predições de ambos classificadores não alcançou resultados superiores ao modelo PU nesta escala, obtendo uma redução de $16,5\%$. Os ganhos da melhor abordagem sobre o *baseline* são mais expressivos se considerarmos baixos valores de FPR. Por exemplo, o melhor método (PU) alcançou uma redução de 11% com uma taxa de FPR de 0.001, enquanto o baseline *normpaths* atingiu uma redução de apenas 7% nessa escala de precisão. Ainda que superem o baseline *normpaths* as abordagens obtiveram uma redução abaixo da alcançada pelo limite superior que foi próxima de 19%.

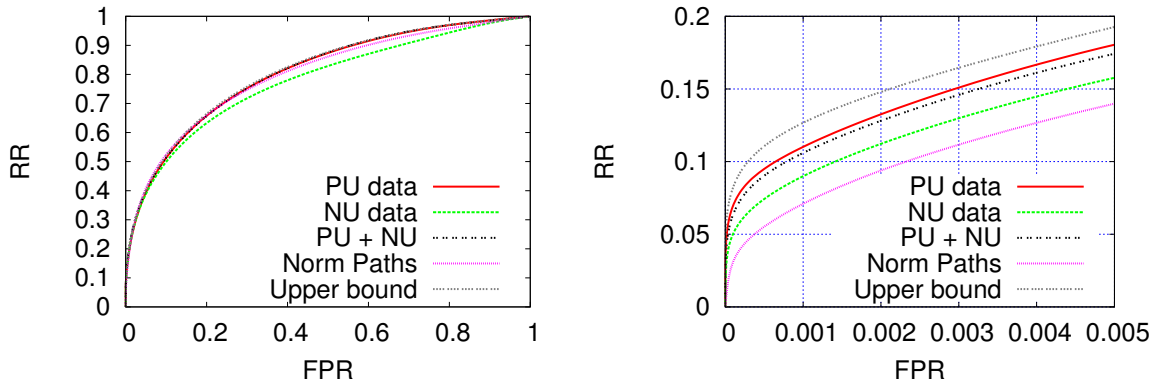

Figura 4.8: Esquerda e Direita: Taxa de redução e Falsos-positivos

Em seguida são apresentados os resultados e melhorias alcançadas ao utilizar o método (Seção 3.3.3) que encontra o melhor limiar para cada par de réplicas individualmente. Além disso são demonstrados os resultados utilizando o algoritmo B-SVM.

Uso de Limiar Individual por Instância Este segundo conjunto de experimentos utiliza uma abordagem baseada em entropia mínima para escolha do melhor limiar de corte, conforme a Seção 3.3.3.

Conforme a Tabela 4.3, em se tratando de características utilizadas individualmente, as melhores foram *ndist* e *fullpath*. Utilizando a abordagem PU, as piores características foram *ip3* e *ip4*, enquanto *namematch* foi a pior na abordagem NU.

Na maioria dos casos, quando descartamos apenas uma característica obtivemos um resultado muito próximo aos alcançados com o uso de todas as características. Isso sugere que algumas das características são redundantes. De fato tal redundância é clara em alguns conjuntos de características, como nos casos de *ip3* e *ip4*, referentes aos endereços IP dos sítios, assim como *nmatch* e *ndist* os quais se referem às strings das URLs. Em contrapartida ao descartamos a característica *fullpath* os resultados demonstraram a maior queda em desempenho.

	AUC (Area Under the Curve)			
	Individualmente		Todas exceto	
Features	PU	NU	PU	NU
ip3	0.5132	0.5116	0.9706	0.9411
ip4	0.5288	0.5283	0.9701	0.9431
nmatch	0.6565	0.2312	0.9646	0.9391
ndist	0.7716	0.6528	0.9631	0.9272
fullpath	0.7187	0.6274	0.9550	0.9206
$Todas$	0.9908	0.9688	0.9908	0.9688

Tabela 4.3: Desempenho de diferentes combinações de características

A Figura 4.9 (esquerda) mostra a análise ROC para a avaliação dos métodos. O algoritmo *normpaths* obteve a pior relação entre as taxas de verdadeiros positivos (TPR) e falsos positivos (FPR), alcançando um número de 0.9415 em termos de AUC. Os classificadores construídos através da abordagem PU e NU tiveram um desempenho de 0.9908 e 0.9688, respectivamente. A combinação de ambos classificadores utilizando a agregação proposta via Eficiência de Pareto, resultou em uma melhoria que chegou a 0.9950, o que é extremamente próximo ao valor de 0.9956 atingido pelo limite superior. O algoritmo B-SVM demonstra um desempenho competitivo alcançando valores de até 0.9824 em termos de AUC.

As curvas de precisão e revocação podem ser visualizadas na Figura 4.9 (direita). Mais uma vez o *NormPaths* obtém os piores resultados. Além disso, os classificadores construídos através de dados PU superam bastante os resultados obtidos com o uso de NU, assim como o B-SVM. A combinação dos classificadores construídos com PU e NU, utilizando a técnica de agregação por Fronteira de Pareto proposta, mostra um desempenho levemente superior.

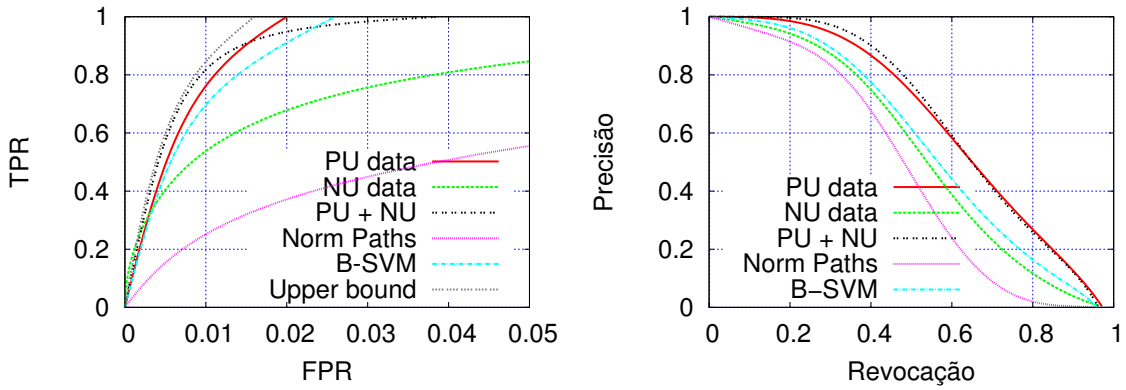

Figura 4.9: Esquerda: Análise da curva ROC. Direita: Precisão x Revocação

A Figura 4.10 mostra a relação entre a taxa de redução e a taxa de falsos positivos. Uma vez que todos os algoritmos mostram um desempenho similar quando permitido um grande numero em FPR, é mostrada a relação considerando um valor máximo de 0.005 em

FPR. Nessa escala, o *normpaths* atinge os piores resultados, fornecendo uma redução de 14% com uma taxa de FPR de 0.005. Nesse mesmo nível de precisão, classificadores aprendidos a partir dos modelos de PU e NU conseguiram uma redução de 18% e 17%, respectivamente. A combinação das predições de ambos classificadores possibilitou uma redução extremamente próxima dos 19% alcançados pelo limite superior. Os ganhos da melhor abordagem sobre o *baseline* são mais expressivos quando considerados baixos valores de FPR. O melhor método (PU + NU) alcançou uma redução de 12% com uma taxa de FPR de 0.001, enquanto o baseline *normpaths* atingiu uma redução de apenas 7% nessa escala de precisão.

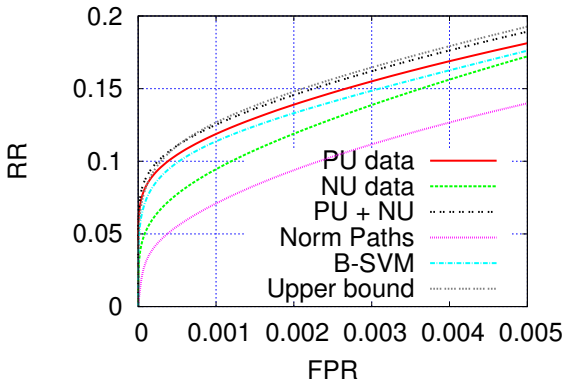

Figura 4.10: Taxa de redução e Falsos-positivos

Os resultados alcançados utilizando a metodologia de entropia mínima para definição do corte mínimo de réplicas foram superiores aos obtidos quando se é utilizado um limiar fixo de réplica. A próxima seção mostra a combinação entre as taxas de detecção de réplicas dos métodos discutidos.

4.4 Taxa de Detecção de Sítios Replicados

A Tabela 4.4 mostra os valores em taxa de detecção para os diferentes algoritmos. O classificador construído através da estratégia de PU é bastante efetivo se considerarmos menos de 1.000 candidatos a réplica, obtendo resultados próximos ao limite superior. Por outro lado, o classificador construído por NU tem uma qualidade inferior se consideramos menos de 10 candidatos a réplicas. Ainda assim, seu desempenho parece se manter quase constante quando o número de candidatos cresce, indicando que as réplicas encontradas receberam uma pontuação bem alta. O *NormPaths* mostrou uma característica similar ao NU, alcançando resultados bem parecidos.

A próxima seção mostra a comparação entre o melhor método (PU+NU) de detecção de réplicas intersítios apresentado neste trabalho e uma abordagem de detecção de replicas intrassítios.

	Número de Candidatos			
Algorithms	$10+1$	$100+1$	$1000+1$	$10000+1$
PU data	0.9900	0.9891	0.9615	0.7656
NU data	0.6590	0.4429	0.4300	0.4287
PU + NU	1.0000	1.0000	0.9905	0.8272
NormPaths	0.6619	0.4192	0.4105	0.4088
B-SVM	0.9805	0.9622	0.9349	0.7034
Upper bound	1.0000	0.9901	0.9586	0.7555

Tabela 4.4: Taxa de detecção de réplicas

4.5 Combinação de algoritmos intersítios e intrassítios

A Tabela 4.5 mostra a taxa de redução obtida pelo algoritmo de detecção de réplicas intersítios proposto, com diferentes taxas de FPR. A tabela também mostra uma taxa de redução obtida por um algoritmo que detecta réplicas intrassítios [Dasgupta et al., 2008].

A parte superior da tabela mostra o número de URLs duplicadas na coleção de teste e a distribuição desse valor em duplicatas intrassítios e intersítios. Em seguida é possível observar a interseção entre os diferentes tipos de duplicatas, ou seja, quantas URLs são tanto intrassítios quanto intersítios. Além disso é demonstrada a quantidade de duplicatas obtidas ao variar o grau de erro dos algoritmos de detecção intersítios (FPR de 0.000 a 0.005) e intrassítios (ϵ de 0.0 a 0.1). O valor de FPR representa a taxa de sítios falsos positivos recuperados durante a detecção de réplicas e ϵ representa o número de regras erradas admitidas pelo algoritmo de DUST ao representar URLs de conteúdo similar.

A parte inferior da tabela mostra a taxa de redução obtida ao aplicar os algoritmos e o quanto é possível reduzir ao combinar as técnicas distintas. É possível também avaliar a taxa de redução obtida com a variação da taxa de erro permitida em cada técnica.

Existem 6.948.501 URLs duplicadas na coleção: 6.514.746 duplicatas intrassítios e 843.526 duplicatas intersítios. Além disso, 409,771 URLs ocorrem tanto como intersítios quanto como intrassítios simultaneamente. Foram considerados duas configurações de algoritmos intrassítios: (i) sem nenhuma taxa de erro para as regras geradas ($\epsilon = 0.0$) que elimina cerca de 4% das duplicatas, e (ii) com uma alta taxa de erros ($\epsilon = 0.1$) que elimina cerca de 23% do conteúdo duplicado. A redução obtida pelo algoritmo proposto depende da taxa de FPR permitida, e varia de 7.9%, sem nenhuma taxa de falsos positivos, para 19% com uma FPR de 0.005. Finalmente, a combinação de ambas técnicas possibilita uma redução maior que 21% com $\epsilon = 0.0$. A taxa de redução sobe para 37% para $\epsilon = 0.1$.

		FPR		
		0.000	0.001	0.005
# URLs duplicadas	6948501	–	–	–
# URLs intrassítios	6514746	–	–	–
$\rightarrow\ \epsilon = 0.0$	293374	–	–	–
$\rightarrow\ \epsilon = 0.1$	1628685	–	–	–
# URLs intersítios	843526	555880	865384	1331215
# inter $\cap$ intra	409771	286485	446182	758927
$\rightarrow \epsilon = 0.0$	64947	38835	70232	110284
$\rightarrow \epsilon = 0.1$	151683	98377	170827	388022
RR intra	0.9376	–	–	–
$\rightarrow\epsilon = 0.0$	0.0422	–	–	–
$\rightarrow\epsilon = 0.1$	0.2344	–	–	–
RR inter	0.1213	0.0791	0.1245	0.1916
RR inter + intra	1.0000	–	–	–
$\rightarrow \epsilon = 0.0$	0.1541	0.1166	0.1567	0.2179
$\rightarrow \epsilon = 0.1$	0.3340	0.3002	0.3343	0.3701

Tabela 4.5: Taxa de redução em URLs duplicadas

4.6 Tempo de Execução

O processo EM é executado inteiramente *offline*, construindo assim um conjunto de treinamento final para o classificador. A Tabela 4.6 mostra o tempo gasto durante o processo EM. A criação do conjunto de treino através de PU gasta mais de 2.5 horas se um classificador parcial é construído do zero após cada transição de rótulos. Esse tempo cai para 112 segundos se os modelos parciais são construídos e atualizados de maneira incremental [Lourenco Jr. et al., 2014]. A criação de um conjunto de treino através de NU exige mais iterações, e portanto mais tempo é necessário. Finalmente são gastos menos de 0.1 segundos para construir um classificador para uma instância arbitraria de candidato a réplica.

Offline		Online	
EM process	Time	Learning classifiers	Time
−PU (from scratch)	8838.98	−Rule extraction	0.0575
−PU (incremental)	112.26	−Prediction	0.0086
−NU (from scratch)	9172.37	−Aggregation	0.0232
−NU (incremental)	131.11		

Tabela 4.6: Tempo de execução em segundos

4.7 Comparação entre Métodos

Esta seção apresenta uma discussão sobre o comportamento e resultados obtidos pelos algoritmos discutidos neste trabalho.

NormPaths Um fator que prejudica a identificação de réplicas com o uso de heurísticas baseadas no conteúdo textual das páginas é a estratégia de cobertura em largura adotada por coletores web. Nesse caso, é priorizada a busca por novos sítios ao invés de sítios completos, o que dificulta o processo de encontrar interseções entre os conjuntos de páginas de sítios replicados nas bases das maquinas de busca. Sendo assim, mesmo que um sítio A seja uma réplica de um sítio B, é possível que o conjunto de páginas de A, conhecido até o momento pelo coletor web, seja muito distinto do conjunto de páginas coletado de B, o que prejudica a heurística de predição. Além disso sítios duplicados podem conter conteúdo dinâmico, onde mesmo que determinada URL seja coletada duas vezes em seguida, o conteúdo coletado irá diferir. Também podem existir páginas de conteúdo idêntico ou similar e ainda assim não serem réplicas. Um exemplo de sítios de conteúdo similar que não constituem uma réplica são os sítios que armazenam cifras musicais. Nesse tipo de sítio o conteúdo é muito similar, uma vez que ambos podem ter as mesmas letras musicais.

Por combinar várias características de naturezas diferentes e utilizar um processo de classificação para aprender padrões que caracterizam a Web e por conseguinte sítios replicados, o algoritmo de aprendizado de máquina foi capaz de superar problemas enfrentados pelo NormPaths alcançando assim resultados superiores.

B-SVM O algoritmo semissupervisionado B-SVM foi competitivo na criação de conjuntos de treino mais diversos assim como na avaliação de pares candidatos a réplica. Porém, o custo computacional exigido para treinamento do algoritmo durante a coleta de páginas web é uma grande desvantagem. Outro fator importante é que as características inerentes do algoritmo não permitem a utilização de múltiplos limiares de transição de rótulos durante o processo EM. Mais especificamente, esse algoritmo não realiza uma indução de treino individual para cada instância de teste, sendo necessário um limiar global para todas as instâncias.

Réplicas intrassítios Versus Réplicas intersítios Foi observado um número expressivamente maior de URLs duplicadas intrassítios do que réplicas intersítios. A primeira explicação possível também está relacionada à característica de cobertura da Web realizada pelos coletores, uma vez que podem ser encontradas interseções pequenas de páginas conhecidas, mesmo em sítios completamente duplicados.

Outro fator importante é que, no momento em que um par de sítios duplicados é detectado, é possível parar a coleta desses sítios, fazendo com que novas URLs não sejam coletadas (inclusive URLs internamente duplicadas). Desse modo, a taxa de redução de conteúdo duplicado pode ser ainda maior. Porém, este trabalho não propôs uma metodologia para escolha correta de qual sítio deve ser removido em um par replicado, ou ainda, quando sítios replicados devem ser reavaliados.

Capítulo 5

Conclusões e Trabalhos Futuros

A presença de conteúdo duplicado na Web tem um impacto negativo em sistemas de recuperação de informação. As máquinas de busca da web sofrem o custo de armazenar e processar conteúdo desnecessário e até mesmo por prover resultados de busca que não oferecem valor real aos usuários.

É possível dividir os tipos de URLs duplicadas em conjuntos intrassítios, se elas ocorrem dentro de um mesmo sítio web, e intersítios, se ocorrem em sítios distintos. Enquanto a maioria dos trabalhos na literatura lida com réplicas intrassítios, uma solução completa exige que o conteúdo duplicado intersítios também seja tratado.

Esta dissertação propôs um algoritmo para detecção de réplicas intersítios e avaliou o impacto da remoção de sítios duplicados sobre uma coleção real de páginas web. Foi proposta uma abordagem baseada em Maximização de Expectativas na criação de conjuntos de treino para um classificador binário. Mais especificamente, essa abordagem permite a identificação de exemplos não óbvios a partir de exemplos óbvios e fáceis de se conseguir. Além disso, as características do algoritmo de classificação utilizado (LAC) permitiram a definição de valores ótimos para os parâmetros que definem os rótulos dos exemplos desconhecidos. Assim foi possível criar um treinamento efetivo para o classificador proposto sem o alto custo de anotação humana do conjunto de treino.

Os resultados de um classificador construído a partir de exemplos positivos de réplicas foi combinado aos resultados de um classificador construído a partir de exemplos negativos de réplicas. Essa estratégia faz com que os erros associados a um classificador possam ser compensados por outro classificador e assim melhorar o desempenho da tarefa de detecção de réplicas. Os experimentos realizados mostraram uma redução de quase 8% no número de URLs duplicadas. Se for permitida uma taxa de falsos positivos de 0.005, a taxa de redução sobe para 19%. Finalmente a combinação do algoritmo com técnicas de eliminação de réplicas intrassítios possibilitou uma redução de até 21% no número de duplicadas.

No futuro, pretendemos estudar novas características que ajudem a melhorar a qualidade do algoritmo de detecção de réplicas proposto. Uma possibilidade é avaliar a conectividade entre sítios, ou seja qual a relação entre sítios duplicados, como possíveis apontamentos compartilhados entre si. Além disso pretendemos investigar novas estratégias de combinação de rankings oriundos de treinamento com exemplos de réplicas e não réplicas. Também pretendemos realizar um estudo sobre a melhor estratégia para escolha de quais sítios replicados devem ser propriamente removidos das bases de máquinas de busca e quais devem permanecer. Também é importante investigar o impacto das técnicas propostas em tempo de coleta e estudar a viabilidade da adaptação do algoritmo proposto em um coletor real.

Referências Bibliográficas

Agarwal, A.; Koppula, H. S.; Leela, K. P.; Chitrapura, K. P.; Garg, S.; GM, P. K.; Haty, C.; Roy, A. & Sasturkar, A. (2009). Url normalization for de-duplication of web pages. Em *18th ACM Conference on Information and Knowledge Management*, pp. 1987–1990.

Aizerman, A.; Braverman, E. M. & Rozoner, L. I. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, pp. 821--837.

Alpaydin, E. (2004). *Introduction to Machine Learning*. The MIT Press.

Baeza-Yates, R. A. & Ribeiro-Neto, B. A. (2011). *Modern Information Retrieval - the concepts and technology behind search, Second edition*. Pearson Education Ltd., Harlow, England.

Bar-Yossef, Z.; Keidar, I. & Schonfeld, U. (2007). Do not crawl in the dust: different urls with similar text. Em *16th International World Wide Web Conference*, pp. 111–120.

Bar-Yossef, Z.; Keidar, I. & Schonfeld, U. (2009). Do not crawl in the dust: Different urls with similar text. *ACM Transactions on the Web*, p. 3.

Bernstein, Y. & Zobel, J. (2005). Redundant documents and search effectiveness. Em *14th International Conference on Information and Knowledge Management*, pp. 736--743.

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

Bharat, K.; Broder, A. Z.; Dean, J. & Henzinger, M. R. (2000). A comparison of techniques to find mirrored hosts on the www. *Journal of the American Society for Information Science*, 51(12):1114–1122.

Börzsönyi, S.; Kossmann, D. & Stocker, K. (2001). The skyline operator. Em *17th International Conference on Data Engineering*, pp. 421–430.

Boser, B. E.; Guyon, I. M. & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. Em *5th Computational Learning Theory*.

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

Broder, A. Z.; Glassman, S. C.; Manasse, M. S. & Zweig, G. (1997). Syntactic clustering of the web. *Computer Networks*, pp. 1157–1166.

Cho, J.; Shivakumar, N. & Garcia-Molina, H. (2000). Finding replicated web collections. Em *SIGMOD Record*, pp. 355–366.

Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273--297.

Croft, B.; Metzler, D. & Strohman, T. (2009). *Search Engines: Information Retrieval in Practice*. Addison-Wesley Publishing Company.

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

Dasgupta, A.; Kumar, R. & Sasturkar, A. (2008). De-duping urls via rewrite rules. Em *14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 186–194.

Davis, A.; Veloso, A.; da Silva, A. S.; Laender, A. H. F. & Meira Jr., W. (2012). Named entity disambiguation in streaming data. Em *50th Annual Meeting of the Association for Computer Linguistics*, pp. 815–824.

Dempster, A. P.; Laird, N. M. & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em em algorithm. pp. 1--21.

Dougherty, J.; Kohavi, R. & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. Em *Twelfth International Conference on Machine Learning*, pp. 194–202.

Fetterly, D.; Manasse, M. & Najork, M. (2003). On the evolution of clusters of near-duplicate web pages. Em *1st Latin American Web Congress*, pp. 37–45.

Frakes, W. B. & Baeza-Yates, R. A., editores (1992). *Information Retrieval: Data Structures & Algorithms*. Prentice-Hall, Inc.

Grünwald, P. & Langford, J. (2007). Suboptimal behavior of bayes and mdl in classification under misspecification. *Machine Learning*, 66(2-3):119–149.

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

Koppula, H. S.; Leela, K. P.; Agarwal, A.; Chitrapura, K. P.; Garg, S. & Sasturkar, A. (2010). Learning url patterns for webpage de-duplication. Em *3rd International Conference on Web Search and Web Data Mining*, pp. 381–390.

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

Liu, B.; Dai, Y.; Li, X.; Lee, W. & Yu, P. (2003). Building text classifiers using positive and unlabeled examples. Em *3rd IEEE International Conference on Data Mining - ICDM*, pp. 179–188.

Lourenco Jr., R.; Veloso, A.; Pereira, A. M.; Jr., W. M.; Ferreira, R. & Parthasarathy, S. (2014). Economically-efficient sentiment stream analysis. Em *37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 637–646.

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

Moreira, M.; dos Santos, J. A. & Veloso, A. (2014). Learning to rank similar apparel styles with economically-efficient rule-based active learning. Em *International Conference on Multimedia Retrieval*, p. 361.

Palda, F. (2011). *Pareto's Republic and the new Science of Peace*. Cooper-Wolfling.

Ribeiro, M. T.; Lacerda, A.; Moura, E.; Hata, I.; Veloso, A. & Ziviani, N. (2014). Multi-objective pareto-efficient approaches for recommender systems. *Transactions on Intelligent Systems and Technology*, 5(1).

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

Rodrigues, K. W. L.; Cristo, M.; de Moura, E. S. & da Silva, A. S. (2013). Learning url normalization rules using multiple alignment of sequences. Em *20th String Processing and Information Retrieval*, pp. 197–205.

Salton, G. & Lesk, M. (1968). Computer evaluation of indexing and text processing. *Journal of the ACM*, 15(1):8–36.

Salton, G. & Yang, C. S. (1973). On the specification of term values in automatic indexing. pp. 351–372.

Silva, I. S.; Gomide, J.; Veloso, A.; Jr., W. M. & Ferreira, R. (2011). Effective sentiment stream analysis with self-augmenting training and demand-driven projection. Em *34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 475--484.

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

Veloso, A.; Jr., W. M.; Cristo, M.; Gonçalves, M. A. & Zaki, M. J. (2006a). Multi-evidence, multi-criteria, lazy associative document classification. Em *International Conference on Information and Knowledge Management*, pp. 218--227.

Veloso, A.; Jr., W. M.; Gonçalves, M. A.; de Almeida, H. M. & Zaki, M. J. (2011). Calibrated lazy associative classification. *Information Sciences*, 181(13):2656--2670.

Veloso, A.; Jr., W. M. & Zaki, M. J. (2006b). Lazy associative classification. Em *f6th International Conference on Data Mining*, pp. 645--654.

Veloso, A. & Meira Jr., W. (2011). *Demand-Driven Associative Classification*. Springer Briefs in Computer Science. Springer.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83.

Witten, I. H.; Moffat, A. & Bell, T. C. (1999). *Managing Gigabytes: Compressing and Indexing Documents and Images, Second Edition*. Morgan Kaufmann Publishers Inc.

Yang, H. & Callan, J. P. (2006). Near-duplicate detection by instance-level constrained clustering. Em *29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 421–428.

Ye, S.; Wen, J.-R. & Ma, W.-Y. (2008). A systematic study on parameter correlations in large-scale duplicate document detection. *Knowledge and Information Systems*, 14(2):217–232.

Ziviani, N.; de Moura, E. S.; Navarro, G. & Baeza-Yates, R. A. (2000). Compression: A key for next-generation text retrieval systems. *IEEE Computer*, 33(11):37–44.