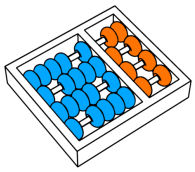
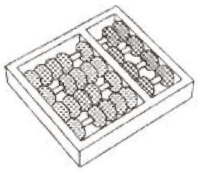


Thiago Anzolin de Godoi

"Uma Abordagem baseada em Realimentação de Relevância para o Problema da Desambiguação de Nome de Autores"

CAMPINAS
2013



Universidade Estadual de Campinas
Instituto de Computação

Thiago Anzolin de Godoi

"Uma Abordagem baseada em Realimentação de Relevância para o Problema da Desambiguação de Nome de Autores"

Orientador(a): **Profa. Dra. Ariadne Maria Brito Rizzoni Carvalho**
Co-Orientador(a): **Prof. Dr. Ricardo da Silva Torres**

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Computação da Universidade Estadual de Campinas para obtenção do título de Mestre em Ciência da Computação.

Este exemplar corresponde à versão final da Dissertação defendida por Thiago Anzolin de Godoi, sob orientação de Profa. Dra. Ariadne Maria Brito Rizzoni Carvalho.

Ariadne M. B. R. Carvalho

Assinatura do Orientador(a)

CAMPINAS
2013

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Maria Fabiana Bezerra Muller - CRB 8/6162

G547a

Godoi, Thiago Anzolin de, 1989-
Uma abordagem baseada em realimentação de relevância para o problema da desambiguação de nome de autores / Thiago Anzolin de Godoi. – Campinas, SP : [s.n.], 2013.

Orientador: Ariadne Maria Brito Rizzoni Carvalho.
Coorientador: Ricardo da Silva Torres.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Programação genética (Computação). 2. Reconhecimento de padrões. 3. Sistemas de recuperação da informação. I. Carvalho, Ariadne Maria Brito Rizzoni,1958-. II. Torres, Ricardo da Silva,1977-. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: A relevance feedback approach for the author name disambiguation problem
Palavras-chave em inglês:
Genetic programming (Computer science)
Pattern recognition
Information storage and retrieval systems
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Ariadne Maria Brito Rizzoni Carvalho [Orientador]
Eduardo Valle
Renata Galante
Data de defesa: 06-12-2013
Programa de Pós-Graduação: Ciência da Computação

TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 06 de dezembro de 2013, pela Banca examinadora composta pelos Professores Doutores:

Profª. Drª. Ariadne Maria Brito Rizzoni Carvalho
IC / UNICAMP

Profª. Drª. Renata de Matos Galante
II / UFRGS

Prof. Dr. Eduardo Alves do Valle Júnior
FEEC / UNICAMP

Instituto de Computação
Universidade Estadual de Campinas

Uma Abordagem baseada em Realimentação de Relevância para o Problema da Desambiguação de Nome de Autores

Thiago Anzolin de Godoi[1]

06 de dezembro de 2013

Banca Examinadora:

- Profa. Dra. Ariadne Maria Brito Rizzoni Carvalho (Orientadora)
- Prof. Dr. Eduardo Valle
  Faculdade de Engenharia Elétrica e de Computação - UNICAMP
- Profa. Dra. Renata Galante
  Instituto de Informática - UFRGS
- Prof. Dr. Alexandre Xavier Falcão
  Instituto de Computação - UNICAMP (Suplente)
- Prof. Dr. André Tavares da Silva
  Centro de Ciências Tecnológicas - UDESC (Suplente)

[1]Apoio financeiro: Bolsa de mestrado CNPq de Mar. a Jul. de 2012, Bolsa de mobilidade internacional Santander de Ago. a Out. de 2012, Auxílio viagem FAEPEX (Jul/2013).

Resumo

Este trabalho apresenta um novo método semiautomático para desambiguação de nomes que explora a utilização de iterações com realimentação de relevância. Uma etapa não supervisionada é utilizada para definir exemplos puros para o treinamento, e uma etapa híbrida supervisionada é empregada para aprender a função de classificação que irá atribuir autores a referências. O modelo combina um classificador por floresta de caminhos ótimos (OPF - *Optimum–Path Forest*) com uma função de similaridade complexa gerada por um algoritmo de Programação Genética (PG).

As principais contribuições deste trabalho são: (i) proposta de um novo método para desambiguação de nomes de autores; (ii) avaliação em uma nova aplicação, da combinação entre os algoritmos OPF e PG, também conhecida como GOPF (*Genetic Programming* e *Optimum–Path Forest*), incrementada por uma etapa de realimentação de relevância; (iii) avaliação do algoritmo do GOPF em um problema de classificação multiclasse; e (iv) adaptação do algoritmo do GOPF para lidar com problemas de classificação de conjunto aberto, isto é, que não possuem todas as classes definidas previamente.

O método proposto foi validado em duas coleções tradicionais muito utilizadas para avaliação de métodos de desambiguação de nomes de autores. A primeira é a coleção extraída da DBLP e que possui 4.287 referências associadas a 220 autores distintos; a segunda é chamada de KISTI, gerada pelo *Korea Institute of Science Technology Information*, e que contém os primeiros 1000 autores mais frequentes na versão do banco de dados da DBLP no final de 2007. Após 5 iterações de realimentação do usuário, nossa abordagem atingiu os melhores resultados para a desambiguação de nomes de autores quando comparado com os outros métodos existentes que utilizam somente as informações básicas da referência.

Abstract

This work presents a new name disambiguation method that exploits user feedback on ambiguous references across iterations. An unsupervised step is used to define pure training samples, and a hybrid supervised step is employed to learn a classification model for assigning references to authors. Our disambiguation method combines the Optimum–Path Forest (OPF) classifier with complex reference similarity functions generated by a Genetic Programming (GP) framework.

The main contributions of this work are: (i) proposal of a novel author name desambiguation method; (ii) evaluation in a new application of the combination between GP and OPF algorithms, also known as GOPF, in interaction learning systems; (iii) evaluation of the GOPF algorithm in a multi-class classification problem; and (iv) extension of the GOPF algorithm to handle open-set classification problems, i.e., classification problems in which class samples are not known in advance.

The proposed method was validated with two traditional databases largely used for the evaluation of author name disambiguation methods: one is a collection extracted from DBLP which sums up 4,287 references associated with 220 distinct authors; the other is called KISTI and was built by the Korea Institute of Science and Technology Information; it contains the top 1000 most frequent author names from the late-2007 DBLP database. After 5 iterations of relevance feedback, our approach yielded the best results for author name disambiguation when compared with the state-of-the-art methods that just consider basic reference information, such as author names, publication title, and venue title.

Agradecimentos

Eu gostaria de agradecer ...

A Deus, pela proteção diária e também por todos os dons que me concedeu.

Aos meus pais, Marcos e Catarina, que no decorrer de toda a minha vida me deram educação, carinho, amor, caráter e me incentivaram na busca de meus objetivos.

A minha noiva Fabiana Thomé, que está sempre ao meu lado nos momentos de alegria e de dificuldade.

Ao CNPq, ao banco Santander e à FAEPEX pelo apoio financeiro para execução deste trabalho.

Aos professores Dr. Edward Fox e Dr. Patrick Fan, da *Virginia Polytechnic Institute and State University*, que me receberam entre agosto e novembro de 2012 e me deram valiosos conselhos e orientações para realização deste trabalho.

Aos professores Dr. Anderson Almeida Ferreira, da Universidade Federal de Ouro Preto e Dr. Marcos André Gonçalves, da Universidade Federal de Minas Gerais, que colaboraram com este trabalho cedendo as bases de dados para realização dos experimentos e participando como coautores do artigo publicado.

Aos meus orientadores Profa. Dra. Ariadne Maria Brito Rizzoni Carvalho e Prof. Dr. Ricardo da Silva Torres, que foram partes fundamentais para realização deste trabalho e sempre estiveram dispostos a me orientar, corrigir e ensinar.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Bibliotecas Digitais (BDs) podem ser definidas como repositórios de uma ou mais coleções, que provêm acesso aos dados enriquecidos com uma variedade de serviços baseados em informações para seus usuários, que podem ser produtores, consumidores ou gerenciadores deste conteúdo [46]. Alguns serviços adicionais típicos oferecidos por BDs são suporte à colaboração entre usuários; preservação de coleções; classificação, disseminação e recuperação de informação; e identificação e classificação de pesquisadores [31]. DBLP[1], Citeseer[2], BDBComp[3] e MEDLINE[4] são exemplos de bibliotecas digitais tradicionais.

Um dos mais importantes usuários das bibliotecas digitais são as agências de financiamento, que se baseiam em informações obtidas de BDs, como quantidade de publicações e de citações, para calcular métricas de avaliação da qualidade da produção científica de pesquisadores, grupos de pesquisa e instituições [24]. Esses valores são utilizados pelas agências para tomada de decisões, como por exemplo, conceder ou não apoio financeiro a um projeto. Entretanto, ao realizar esse tipo de análise, os usuários pressupõem que o conteúdo seja de alta qualidade [43]. Essa necessidade faz com que a imperfeição dos dados nas BDs seja considerada um problema crítico, que afeta além da *eficácia*, a *eficiência* dos serviços prestados por estes portais.

A tarefa de melhorar e manter a qualidade da informação presente nas BDs é um caso particular de uma linha de pesquisa chamada *data cleaning* (limpeza de dados). As pesquisas voltadas a *data cleaning* definem esse processo como detecção e remoção de erros e inconsistências de um conjunto de dados [58]. No caso das BDs, um dos principais e mais complexos desafios para melhorar a qualidade das informações é a *desambiguação de nomes de autores*. Esse problema ocorre quando um mesmo autor publica utilizando nomes similares, porém distintos (sinônimos), ou quando autores distintos compartilham

[1] `http://www.informatik.uni-trier.de/~ley/db/` (Acessado em Jul. de 2013).
[2] `http://citeseerx.ist.psu.edu/index` (Acessado em Jul. de 2013).
[3] `http://www.lbd.dcc.ufmg.br/bdbcomp/` (Acessado em Jul. de 2013).
[4] `http://www.ncbi.nlm.nih.gov/pubmed/` (Acessado em Jul. de 2013).

o mesmo nome, ou variações muito parecidas do mesmo nome (homônimos).

Homônimos surgem naturalmente, pois nomes são sequências de letras que não possuem nenhuma restrição quanto a serem únicas e, em casos nos quais abreviações são utilizadas, mesmo nomes diferentes podem ser reduzidos à mesma forma. Já os sinônimos são originados por diversas causas, como erros de digitação, erros de OCR[5], abreviações, diferentes fontes de informação e a falta de um padrão para a forma de representar um nome [45].

A Tabela 1.1 apresenta um exemplo real de ambiguidade envolvendo nomes de autores, que foi extraído da base de dados da DBLP. Os nomes destacados nas citações c_1, c_2 e c_3 são sinônimos, pois os três são diferentes formas de representar o nome do autor *Jan M. Smith*, assim como os nomes destacados nas citações c_4 e c_5 são duas formas diferentes de representar o nome do autor *John Miles Smith*. Os nomes destacados nas citações c_3 e c_5 são homônimos, pois apesar de serem iguais, representam dois autores diferentes.

Tabela 1.1: Exemplos de nomes de autores pertencentes a DBLP.

Id	Referência Bibliográfica
c_1	Thierry Coquand, **Jan M. Smith**. An application of constructive completeness. TPP, 1996.
c_2	Bengt Nordström, Kent Petersson, **Jan Smith**. Programming in Martin-Löf's type theory. 1990.
c_3	T. Coquand, S. Sadocco, G. Sambin, **J. Smith**. Formal topologies on the set of first-order formulae. JSL, 2000.
c_4	Chyuan Shiun Lin, Diane Smith, **John Miles Smith** The design of a rotating associative array memory for a relational database management application. TODS, 1976.
c_5	D. Smith, **J. Smith**. Relational data base machines. Computer, 1979.

A tarefa de desambiguar nomes de autores pode ser formulada da seguinte forma: Considere uma coleção de *registros de citações*, cada um contendo uma lista de *atributos*. Essa lista necessariamente inclui nomes de autores, título do trabalho e título da publicação. Cada atributo associa um valor específico à citação. Um atributo pode ser composto por diversos *elementos*. No caso do atributo "nome de autores", um elemento corresponde ao nome de um único autor. Cada elemento nome de autor é uma *referência* para um autor. O objetivo de uma tarefa de desambiguação é produzir uma função de desambiguação que é usada para particionar o conjunto de referências em autores, em que cada bloco da partição contenha idealmente todas as referências para um mesmo autor e nenhuma referência para outros autores.

[5]Do inglês *Optical Character Recognition*, o reconhecimento óptico de caracteres é uma tecnologia para digitalizar documentos impressos, escritos ou datilografados em forma de texto.

Assumimos que cada referência de autor extraída de um registro de citação contém pelo menos os seguintes atributos: nome do autor, nome dos coautores, título do trabalho e nome da publicação. Por exemplo, a Tabela 1.2 apresenta uma referência ao autor Jan M. Smith, encontrada na citação c_1 da Tabela 1.1, e seus atributos.

Tabela 1.2: Exemplo de referência para um autor e seus atributos.

Referência	Jan M. Smith.
Nomes dos coautores	Thierry Coquand
Título do Trabalho	An application of constructive completeness
Título da Publicação	Types for Proofs and Programs

Antes da execução de um método de desambiguação de nomes de autores, é necessário realizar uma etapa de pré-processamento da coleção, conhecida na literatura como blocagem, para tratar problemas relacionados à escalabilidade, evitando a necessidade de comparações entre todas as referências [54]. Nessa etapa, o conjunto de referências para autores é particionado em grupos de referências chamados *grupos ambíguos*. Em cada grupo ambíguo, os valores dos nomes dos autores são similares, de acordo com uma determinada regra. No método desenvolvido neste trabalho, assim como na maioria dos outros métodos que obtiveram bons resultados descritos na literatura, os autores são agrupados com aqueles que possuem o mesmo sobrenome e a primeira letra do primeiro nome em comum.

Uma grande quantidade de métodos foram propostos para lidar com o problema da desambiguação dos nomes de autores [26]. Entretanto, ainda há espaço para melhoras substanciais, principalmente quando consideramos o caso em que há somente o mínimo de informação disponível nas citações, por exemplo, quando cada citação possui apenas os nomes dos autores, o título do trabalho e o nome da publicação.

Além disso, os melhores métodos encontrados na literatura são normalmente baseados em técnicas de aprendizado supervisionado [36, 70], que exigem uma etapa custosa de geração de dados para o treinamento do algoritmo. A construção dos exemplos para uso na fase de aprendizado do algoritmo é muitas vezes inviável para os administradores de BDs, pois exige um alto número de referências rotuladas. Todo esse processo é feito sem nenhuma orientação por parte do sistema, ou seja, a escolha das referências a serem rotuladas como exemplos é totalmente manual e arbitrária.

Nesta dissertação, propomos um novo método de desambiguação que somente requer uma pequena intervenção manual dos administradores das BD. A realimentação provida se dá em poucas iterações e é usualmente baseada na rotulação de casos em que o algoritmo de desambiguação identifica um maior potencial de aprendizado, ou seja, as referências rotuladas pelo usuário para o autor correto e inseridas no conjunto de treinamento, que irão promover o maior ganho de eficácia para o algoritmo de classificação. O método

detecta esses casos explorando etapas intermediárias de um método de classificação e uma combinação de funções de similaridade geradas automaticamente.

Para encontrar essas funções utilizamos *Programação Genética* (PG) [3], que é uma técnica de aprendizado supervisionado baseada na ideia de evolução genética para encontrar as melhores soluções. De acordo com a teoria da evolução, os melhores indivíduos possuem maior aptidão para se reproduzir e continuar evoluindo para se adaptar às mudanças do ambiente (em nosso caso aos dados do treinamento). Nós escolhemos utilizar PG por uma série de razões, incluindo (i) grande eficiência em estudos prévios para encontrar funções de similaridade [1, 7, 17, 21, 23, 30, 67], e (ii) habilidade de encontrar soluções próximas da ótima em grandes espaços de busca. Essa técnica é utilizada em diversas outras aplicações, como por exemplo, na combinação de descritores de imagem em recuperação de imagens por conteúdo [67].

O classificador que utilizamos, chamado Floresta de Caminhos Ótimos - do inglês *Optimum–Path Forest* (OPF) [55, 56], é considerado o estado da arte. O OPF é uma técnica baseada em grafos, que possui algumas vantagens em relação às técnicas tradicionais de classificação: (i) ela não depende de nenhum parâmetro pré-definido; (ii) ela, *a priori*, não assume nenhuma separação entre os objetos no espaço; e (iii) ela trata nativamente o problema da classificação multiclasse. O algoritmo representa cada classe como uma ou mais árvores, e os exemplos a serem classificados são nós que serão disputados por essas árvores, ganhando a árvore em que o caminho do nó até a raiz seja mínimo. O classificador utiliza a função de similaridade encontrada pelo algoritmo de PG para decidir a qual autor a citação pertence, ou se ela pertence a um novo autor não representado no conjunto de treinamento.

A combinação entre PG e o classificador OPF também é conhecida como GOPF (*Genetic Programming - Optimum–Path Forest*).

As principais contribuições deste trabalho são:

- Estudo comparativo de técnicas de desambiguação de nomes de autores.
- Avanço no estado da arte em algoritmos de desambiguação de nomes de autores.
- Implementação de um sistema de aprendizado baseado em programação genética para encontrar funções de similaridade textuais que melhor se aplicam a um determinado conjunto de treinamento.
- Avaliação do algoritmo do GOPF em problemas de classificação multiclasse.
- Extensão do algoritmo do GOPF para problemas de classificação de conjunto aberto, ou seja, quando na etapa de aprendizado ainda não são conhecidas todas as classes, e representantes de novas classes podem surgir na etapa de classificação.

O resto desta dissertação está organizado como descrito a seguir. O Capítulo 2 discute trabalhos relacionados; o Capítulo 3 descreve a abordagem proposta; o Capítulo 4 apresenta os resultados de nossa avaliação experimental; e finalmente, o Capítulo 5 apresenta nossas conclusões e direções para futuros trabalhos.

Capítulo 2

Conceitos e Trabalhos Relacionados

Este capítulo apresenta uma visão geral dos trabalhos relacionados a métodos de desambiguação de nomes de autores e realimentação de relevância. Além disso, são introduzidos conceitos e técnicas que integram o método proposto. A Seção 2.1 apresenta uma revisão dos métodos de desambiguação de nomes de autores e os conceitos de realimentação de relevância, e de que forma essa técnica pode agregar valor ao método proposto. A Seção 2.2 introduz algumas funções de similaridade e de que forma são utilizadas nos métodos de desambiguação. A Seção 2.3 descreve o funcionamento do algoritmo de Programação Genética e as vantagens de sua utilização. E finalmente, a Seção 2.4 apresenta o algoritmo Floresta de Caminhos Ótimos e discute seu diferencial em relação a outros classificadores.

2.1 Desambiguação de nome de autores

A maioria dos métodos automáticos para desambiguação de nomes propostos na literatura adotam soluções que podem ser classificadas de acordo com o tipo de abordagem explorada. Em [26], os autores propuseram uma divisão em duas categorias de abordagem: as que realizam *agrupamento de autores* e as que realizam *atribuição de autores*. Nesta seção iremos discutir os métodos de desambiguação de nomes de autores conhecidos utilizando esta classificação.

2.1.1 Métodos de agrupamento de autores

Métodos que realizam agrupamento de referências por autor utilizam alguns tipos de similaridade entre os atributos das referências [5, 8, 13, 14, 22, 28, 38, 39, 41, 52, 53, 57, 63, 64, 68, 69]. Esses métodos usam técnicas de clusterização junto com funções de similaridade. A função de similaridade é aplicada aos atributos da referência (ou grupo de referências)

para decidir quando colocar referências no mesmo grupo ou não, utilizando a técnica de clusterização. A função de similaridade pode ser: (i) pré-definida, baseada em funções existentes e dependendo do tipo de atributo [5, 8, 13, 38, 53, 63]; (ii) aprendida, utilizando uma técnica supervisionada de aprendizado de máquina [14,39,49,68,69]; ou (iii) extraída dos relacionamentos entre os autores e os coautores, normalmente representados como um grafo [22, 48, 52].

Em [38], é apresentado um método que utiliza uma técnica de clusterização espectral de k-vias, em que as referências são representadas como vetores de características baseadas em TF-IDF [2]. Nesse método, é montada uma matriz quadrada M de tamanho n, em que n é a quantidade total de referências, e cada linha i e coluna j representa uma referência. A matriz é então preenchida utilizando a função de Distância do Cosseno entre as referências: $M[i][j] = d_{cos}(r_i,r_j)$. Após uma série de transformações lineares, obtem-se uma matriz n x k onde cada linha i representa uma citação e cada coluna k representa um cluster, e a posição em determinada linha i que possui maior valor absoluto corresponde ao *cluster* a que a referência será atribuída.

Em [13], os autores propõem uma técnica baseada em heurísticas para combinar funções de similaridade para desambiguar nomes de autores em referências. O algoritmo é dividido em duas etapas: (i) gerar *clusters* puros e (ii) unir *clusters*. Na primeira etapa, o algoritmo utiliza uma técnica que é o agrupamento de referências a autores ambíguos que compartilham pelo menos um coautor. Na segunda etapa, o algoritmo compara cada par de referências (r_i,r_j) de cada par de *clusters* (C_i,C_j), e caso a similaridade entre os títulos dos trabalhos, ou entre os nomes das publicações, ou ainda entre o primeiro nome dos autores (quando presente nas citações) seja maior que um limiar previamente definido, une os *clusters* C_i e C_j em um novo *cluster*.

Em [8], é proposto um sistema chamado INDi (*Incremental Name Disambiguation)*, não supervisionado e incremental para atribuir referências de novas citações inseridas em uma biblioteca digital a seus autores corretos. Para isso, quando uma nova referência é inserida, é realizada uma busca em toda a base de citações por autores com nomes similares, com pelo menos um coautor em comum e títulos de trabalho e publicações similares. Quando não é encontrada nenhuma publicação que compartilha um coautor, a pesquisa é refeita, porém utilizando limiares maiores para as similaridades no título do trabalho e publicação.

Em [39], os autores propõem um método no qual uma função de similaridade é dinamicamente treinada por um algoritmo chamado LASVM [6], que é uma modificação do famoso SVM (*Support Vector Machines*) [11], para tornar o método mais eficiente e consumir menos memória para ser executado, mantendo a acurácia muito próxima do algoritmo original. Utilizando essa função, que foi treinada pelo LASVM, o método aplica um algoritmo de clusterização conhecido como DBSCAN [19], que por sua vez utiliza

conceitos de densidade e transitividade para criar os *clusters*.

Em [14], os autores apresentam uma técnica baseada na aglomeração de *clusters* gulosa e que é orientada a erros, ou seja, novos exemplos de treinamento são gerados a partir de previsões incorretas no conjunto de treinamento.

Em [52], os autores apresentam um método baseado em técnicas de partição em grafos para a solução do problema da desambiguação de nomes de autores. É proposta uma métrica de similaridade entre referências, para identificar se duas citações de autores em duas referências pertencem ao mesmo autor ou não, que é chamada de distância *quasi-clique*. Para calcular essa distância, são criados dois grafos, um para cada referência, e em cada grafo, cada nó n_i representa um autor dessas duas referências, e cada aresta a_{ij} existente no grafo indica que os autores representados pelos nós n_i e n_j possuem pelo menos mais um trabalho em que eles são coautores dentro da coleção. Dados esses dois grafos, a distância entre as referências é baseada no tamanho do maior *quasi-clique* em comum entre os dois grafos gerados.

2.1.2 Métodos de atribuição de autores

Métodos de atribuição de autores visam atribuir as referências diretamente a seus respectivos autores [4, 25, 28, 36, 37, 66, 70]. Esses métodos utilizam técnicas de aprendizado supervisionado [25,28,36,70], ou técnicas de clusterização baseadas em modelo [4,37,66]. Eles inferem o modelo que representa os autores, como por exemplo, as probabilidades de um autor publicar um artigo com outros coautores em uma dada publicação, utilizando uma lista de termos específicos no título do trabalho.

Em [36], são apresentadas duas abordagens para a desambiguação de nomes que utilizam aprendizado supervisionado, e ambas consideram que as informações sobre as citações se limitam ao nome dos autores, título da publicação e título do trabalho. A primeira utiliza um classificador Naïve Bayes (NB) [42], que assume independência entre os atributos que fazem parte das referências no conjunto de treinamento, para gerar um modelo de probabilidades que infere, dados os atributos de uma nova referência, qual é o autor mais provável. O segundo classificador utiliza o SVM (*Support Vector Machines*) [11] para mapear uma citação em um vetor de características e atribuí-la ao autor que possuir a representação, vinda do conjunto de treinamento, mais próxima a essa referência no espaço vetorial.

Em [66], é proposto um método baseado em Hidden Markov Random Field (HMRF) [35] para definir o peso que será atribuído a cada atributo da citação ou relacionamento existente entre um par de referências. Esse peso é então utilizado em conjunto com as características das referências para atribuí-las aos autores. Para estimar o número de autores na coleção, é utilizado um modelo probabilístico baseado em Naïve Bayes. Essa aborda-

gem, apesar de atingir bons resultados, exige uma grande quantidade de informação sobre as referências e que usualmente não está disponível para todas as citações, como resumo e referências citadas pelo trabalho.

2.1.3 Realimentação de relevância

Realimentação de relevância (RF - *Relevance Feedback*) tem sido utilizada principalmente em tarefas de buscas em Recuperação de Informação [2]. Nessas tarefas, os resultados obtidos para uma dada busca são avaliados como relevantes ou não por um usuário que realiza a busca. Esta informação é dada como realimentação para o sistema, de forma que possa melhorar o entendimento do sistema de recuperação em relação ao que o usuário deseja encontrar. A nova busca modificada é então utilizada para recuperar novos documentos, e este processo continua até o usuário estar satisfeito ou sair do sistema.

Alguns tipos de realimentação do usuário aplicados ao problema da desambiguação de nomes de autores são propostos em [22, 25, 50, 71]. Em [22], é proposto um sistema chamado GHOST (*GrapHical framewOrk for name diSambiguaTion*). O algoritmo utiliza uma abordagem baseada em grafos e foca somente em resolver o problema dos sinônimos. Esse método é dividido em cinco etapas: (i) construção do grafo; (ii) seleção de caminhos válidos; (iii) cálculo da similaridade; (iv) clusterização; e (v) realimentação de relevância. Na etapa (i), cada referência a um autor é representada como um vértice em um grafo, e cada aresta representa uma publicação em coautoria entre os autores; na etapa (ii), é selecionado no grafo um subconjunto dos possíveis caminhos entre cada par de vértices, eliminando os que são duplicados pelo fato de algumas publicações possuírem mais de dois autores; na etapa (iii), é proposta uma função de similaridade nesse grafo, que é baseada na lei de Ohm [34] para o cálculo da resistência em um circuito paralelo; na etapa (iv), os autores são clusterizados utilizando uma técnica de propagação por afinidade [32], em que não é necessário especificar a quantidade de *clusters* existentes antes do processamento; finalmente, na etapa (v), são realizados alguns experimentos de interação com o usuário, porém a realimentação esperada do usuário é custosa e pressupõe que o usuário entende o funcionamento do algoritmo.

Em [71], os autores propõem o ADANA (Active Name Disambiguation), que trabalha em um modo iterativo. O sistema solicita ao usuário correções, ao invés de esperar por ações do usuário. Para isso, o sistema escolhe alguns resultados de desambiguação potencialmente errados, depois do processo de desambiguação ter sido executado. A seleção ativa visa minimizar o número de iterações necessárias para maximizar a eficácia. Percebe-se que para obter bons resultados, os autores utilizam grandes quantidades de informação adicional, como afiliação e referências bibliográficas, que na maioria dos casos não está disponível.

Em [50], os autores propõem a utilização de um percéptron como classificador. Eles incorporaram a realimentação do usuário de uma rede de cooperação científica com atributos e restrições, para melhorar o desempenho do método de desambiguação. Inicialmente, utilizam somente atributos extraídos da publicação e da web, como a página pessoal. Depois da primeira desambiguação, os usuários procuram por possíveis problemas. Os atributos e restrições providos pela realimentação, juntamente com os outros atributos, são utilizados para revisar o modelo de classificação. Ele também utiliza outros atributos, além dos que são encontrados normalmente em todas as publicações, como por exemplo a afiliação dos autores. Além disso, ele tenta resolver somente os problemas de homônimos e não sugere possíveis erros, por exemplo, referências ambíguas para guiar a realimentação do usuário.

Em [25], os autores estendem um desambiguador de nomes de autores auto-treinado [28, 70], e que incorpora realimentação do usuário. O desambiguador proposto trabalha em duas etapas. Na primeira, as referências são agrupadas em *clusters* utilizando o grafo de coautoria. Alguns desses *clusters* são selecionados automaticamente e suas referências são utilizadas como exemplos de treinamento inicial para a próxima etapa; as outras referências irão compor o conjunto de treinamento. Na segunda etapa, é utilizado um classificador associativo, que é capaz de detectar novos autores. Esse classificador trabalha em duas fases. Na primeira fase, ele seleciona uma porcentagem das decisões que ele precisa tomar e que foram consideradas mais difíceis pelo algoritmo, e pede ao administrador para atribuí-las para os autores corretos (realimentação do usuário). Na segunda fase, as decisões difíceis que foram rotuladas pelo usuário, são adicionadas ao conjunto de treinamento, e os autores das referências no conjunto de teste são escolhidos.

A Tabela 2.1 sumariza a comparação entre os métodos descritos nas últimas seções. Podemos observar que os métodos que utilizam realimentação de relevância surgiram apenas a partir de 2012.

Tabela 2.1: Tabela comparativa entre métodos de desambiguação de nomes de autores.

Método	**Ano**	**Tipo**	**Realim.**	**Coleções**	**Métricas de avaliação**[1]
Han et al. [36]	2004	Atribuição de autores	Não	DBLP	Precisão
Han et al. [38]	2005	Agrupamento de autores	Não	DBLP	Precisão
Huang et al. [39]	2006	Agrupamento de autores	Não	CiteSeer	Métrica F1
On et al. [52]	2006	Agrupamento de autores	Não	ACM,BioMed, IMDB	Precisão, Revocação
Culotta et al. [14]	2007	Agrupamento de autores	Não	Penn, Rexa, DBLP	Métrica F1, MUC, B-Cubed
Cota et al. [13]	2010	Agrupamento de autores	Não	DBLP, BDBComp	Métrica F1, Métrica K
Ana Paula et al. [8]	2011	Agrupamento de autores	Não	SyGAR, BDBComp	Métrica K
Fan et al. [22]	2011	Agrupamento de autores	Não	DBLP, MEDLINE	Métrica F1
Ferreira et al. [25, 28]	2012	Atribuição de autores	Sim	DBLP, KISTI	Métrica F1, Métrica K
Wang et al. [71]	2012	Agrupamento de autores	Sim	Publication,CALO,New Stories	Métrica F1
Li et al. [50]	2012	Atribuição de autores	Sim	Personal dataset	Métrica F1
Tang et al. [66]	2012	Atribuição de autores	Não	ArnetMiner	Métrica F1

[1] As métricas de avaliação serão descritas em detalhes no Capítulo 4.

Abordagens com realimentação de relevância têm sido propostas baseadas no classificador OPF [15,16,18]. Essas iniciativas, entretanto, focam em recuperação de imagem por conteúdo [15, 16] e tarefas de classificação [18]. Nas abordagens apresentadas em [15, 18], PG também é utilizada para determinar o peso das arestas no grafo OPF. Esses métodos de realimentação de relevância, entretanto, adotam um modelo de classificação binário em que cada imagem é classificada como relevante ou não relevante. Em nosso caso, temos um problema multi-classe, porque referências podem ser atribuídas para mais do que dois autores. Classificação multi-classe é intrinsecamente um problema muito mais complexo. Primeiro, a definição de conjuntos de treinamento adequados, em termos de cobertura e tamanho, é muito mais difícil. Além disso, muitos classificadores que são bons em problemas de classificação binária falham quando utilizados ou estendidos para lidar com múltiplas classes. Em situações práticas, a estratégia normalmente empregada é cara, pois é baseada na combinação de múltiplos classificadores binários.

2.2 Funções de similaridade

Funções de similaridade desempenham um papel fundamental nos algoritmos de desambiguação de nomes de autores. Excluindo-se os métodos que não buscam informações externas, como em Pereira et al. [57], em que os autores exploram informações encontradas na Web para auxiliar na tarefa de desambiguação, todas as outras técnicas propostas na literatura baseiam-se nas características textuais dos atributos de uma referência, como título do trabalho, nome dos autores, afiliação dos autores, título da publicação, ano da publicação, etc. Para comparar os dados das referências é necessário buscar relações e encontrar formas de representá-las. Para isso empregam diversos tipos de funções de similaridade entre os atributos das referências [5,8,13,14,22,28,38,39,41,52,53,57,63,64,68,69].

A função de similaridade é aplicada aos atributos da referência (ou grupo de referências) para decidir quando duas referências pertencem ao mesmo autor ou não. A função de similaridade pode ser pré-definida (baseada em funções existentes e dependendo do tipo de atributo) [5,8,9,13,38,53,63], aprendida utilizando uma técnica supervisionada de aprendizado de máquina [14, 39, 49, 68, 69], ou extraída dos relacionamentos entre os autores e os coautores, normalmente representados como um grafo [22, 48, 52].

A seguir, apresentamos a definição de algumas funções de similaridade textual que utilizamos em nosso trabalho:

Distância de Levenshtein

A Distância de Levenshtein [47], também conhecida como Distância de edição, contabiliza a quantidade de operações necessárias para transformar uma cadeia de caracteres em outra. As operações são: inserção, deleção ou substituição de um caractere. Essa função é muito útil para identificar erros de digitação e de OCR, situação em que com a realização de poucas operações é possível transformar uma cadeia com erros em outra, correta. Ela é definida pela Equação 2.1

$$D_{ij} = i + d + s, \tag{2.1}$$

onde i é a quantidade de inserções, d a quantidade de deleções e s a quantidade de substituições necessárias para a transformação da cadeia de caracteres i na cadeia j.

Distância de Jaro-Winkler

A Distância de Jaro-Winkler [72] é baseada no número e na ordem dos caracteres iguais entre as duas cadeias de caracteres. Para casos em que se deseja identificar similaridade entre duas cadeias em que uma é abreviação de outra, essa função funciona muito bem, pois ela considera a ordem dos caracteres. Ela é definida pela Equação 2.2

$$D_{ij} = \frac{m}{3a} + \frac{m}{3b} + \frac{m-t}{3m}, \tag{2.2}$$

onde m é o tamanho da sequência máxima de caracteres que aparecem em i e j na mesma ordem; a e b a quantidade de caracteres das cadeias i e j, respectivamente; e t é o número de caracteres em comum não necessariamente na mesma posição em ambas.

Distância do Cosseno

A Distância do Cosseno [2] é uma função para calcular distâncias entre vetores. Para aplicar essa função à similaridade entre cadeias de caracteres é necessário mapeá-las utilizando a contagem das palavras presentes no texto. Por exemplo, a cadeia "João da Silva da Costa" poderia gerar um vetor [1,2,1,1], onde cada dimensão do vetor representa respectivamente [João,da,Silva,Costa], e o número em cada posição é a quantidade de aparições da palavra. Feito esse mapeamento, a Equação 2.3 determina como calcular o valor da Distância do Cosseno entre dois nomes representados por estes vetores

$$D_{ij} = \frac{A \cdot B}{||A|| \times ||B||}, \tag{2.3}$$

onde A e B são os vetores que representam as cadeias de caracteres i e j, respectivamente.

Para identificar similaridade entre nomes que são representados utilizando padrões diferentes, como por exemplo [primeiro nome último nome] e [último nome, primeiro nome]; essa função é eficaz pois considera apenas a quantidade de caracteres nas cadeias e não a posição em que eles aparecem.

Distância de Jaccard

Assim como a Distância do Cosseno, a Distância de Jaccard [65] também mapeia a cadeia de caracteres para um vetor de frequência de palavras, porém faz apenas um mapeamento binário. Por exemplo, dada a cadeia de caracteres "João da Silva da Costa", poderia ser gerado um vetor [1,1,1,1,0], em que cada dimensão do vetor representa respectivamente [João,da,Silva,Costa,Melo]. A Equação 2.4 apresenta essa função

$$D_{ij} = \frac{q_{11}}{q_{01} + q_{10} + q_{00}}, \tag{2.4}$$

onde q_{11} é a quantidade de dimensões em que ambos os vetores possuem 1; q_{00} é a quantidade de dimensões em que ambos os vetores possuem 0; q_{10} é a quantidade de dimensões em que o vetor que representa a cadeia i possui 1, e o vetor que representa a cadeia j possui 0; e q_{01} é a quantidade de dimensões onde o vetor que representa a cadeia i possui 0, e o vetor que representa a cadeia j possui 1.

Essa função, assim como a Distância do Cosseno, é eficaz para identificar similaridade entre nomes que são representados utilizando padrões diferentes. Uma pequena diferença entre ela e a Distância do Cosseno é que como sua contagem é binária, o efeito de palavras repetidas (de, da, e) é amortizado em casos que aparecem mais de uma vez na cadeia.

2.3 Programação Genética (PG)

Encontrar uma função de similaridade ideal para um determinado problema é uma tarefa nem sempre trivial, pois alguns algoritmos podem obter bons resultados em coleções específicas e não serem tão efetivos em outros conjuntos. A maioria dos métodos para desambiguação de nomes de autores utiliza alguma forma automática de seleção, ou uma combinação de funções de similaridade, como por exemplo, algoritmos de aprendizado supervisionado.

Aprendizado supervisionado pode ser definido como inferir uma função baseada em um conjunto de exemplos de entradas e saídas esperadas, conhecido como conjunto de treinamento [51]. Programação Genética (PG) [3] é um algoritmo de aprendizado supervisionado baseado na ideia de evolução genética. A literatura mostra que esse método é eficaz em encontrar soluções próximas da ótima, em problemas de domínio esparço. Isso

se deve ao fato do algoritmo utilizar um componente probabilístico em seu treinamento, não limitando assim sua busca a pontos ótimos locais.

Sob a ótica da PG, uma solução de um dado problema é chamada de indivíduo, e esse indivíduo representa um programa de computador que pode ser ilustrado em forma de árvore, como na Figura 2.1.

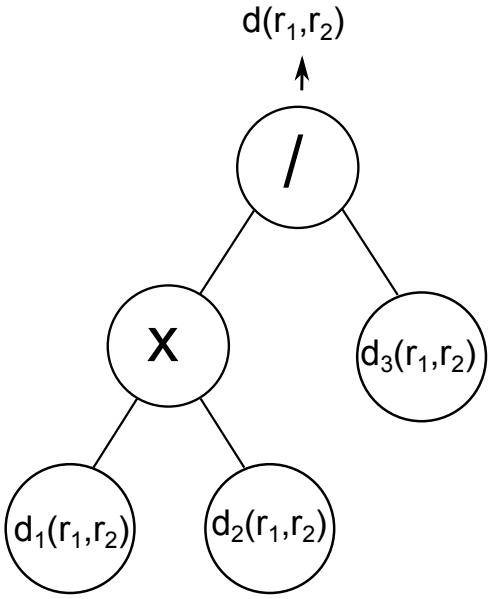


Figura 2.1: Exemplo de indivíduo em programação genética que combina três distâncias d_1, d_2, d_3 entre referências r_1 e r_2 através da função $(\frac{d_1 \times d_2}{d_3})$.

Um conjunto de soluções de um determinado problema é chamado de população, e essa população sofre alterações ao longo de diversas iterações, definidas como gerações no contexto de PG, na tentativa de encontrar indivíduos cada vez melhores.

A cada nova geração, os melhores indivíduos são selecionados e expostos às seguintes operações: reprodução, em que o mesmo indivíduo é reproduzido na próxima geração; mutação, em que um indivíduo sofre uma alteração aleatória em um ou mais genes; e cruzamento, em que dois indivíduos são combinados para formar um novo indivíduo. As três operações são ilustradas na Figura 2.2. Essas operações visam gerar novas e distintas soluções para o problema e são repetidas por um número pré-definido de iterações, ou até que um critério de qualidade da solução seja obtido.

A forma com que é definida a qualidade de um indivíduo faz parte da modelagem do problema, e é nessa etapa que é utilizado o conjunto de treinamento que contém os exemplos pré-rotulados em que os indivíduos serão avaliados. Outros parâmetros do algoritmo de PG que fazem parte da modelagem do problema são: o tamanho da população inicial, a quantidade de indivíduos a serem levados para a próxima iteração, a quantidade de

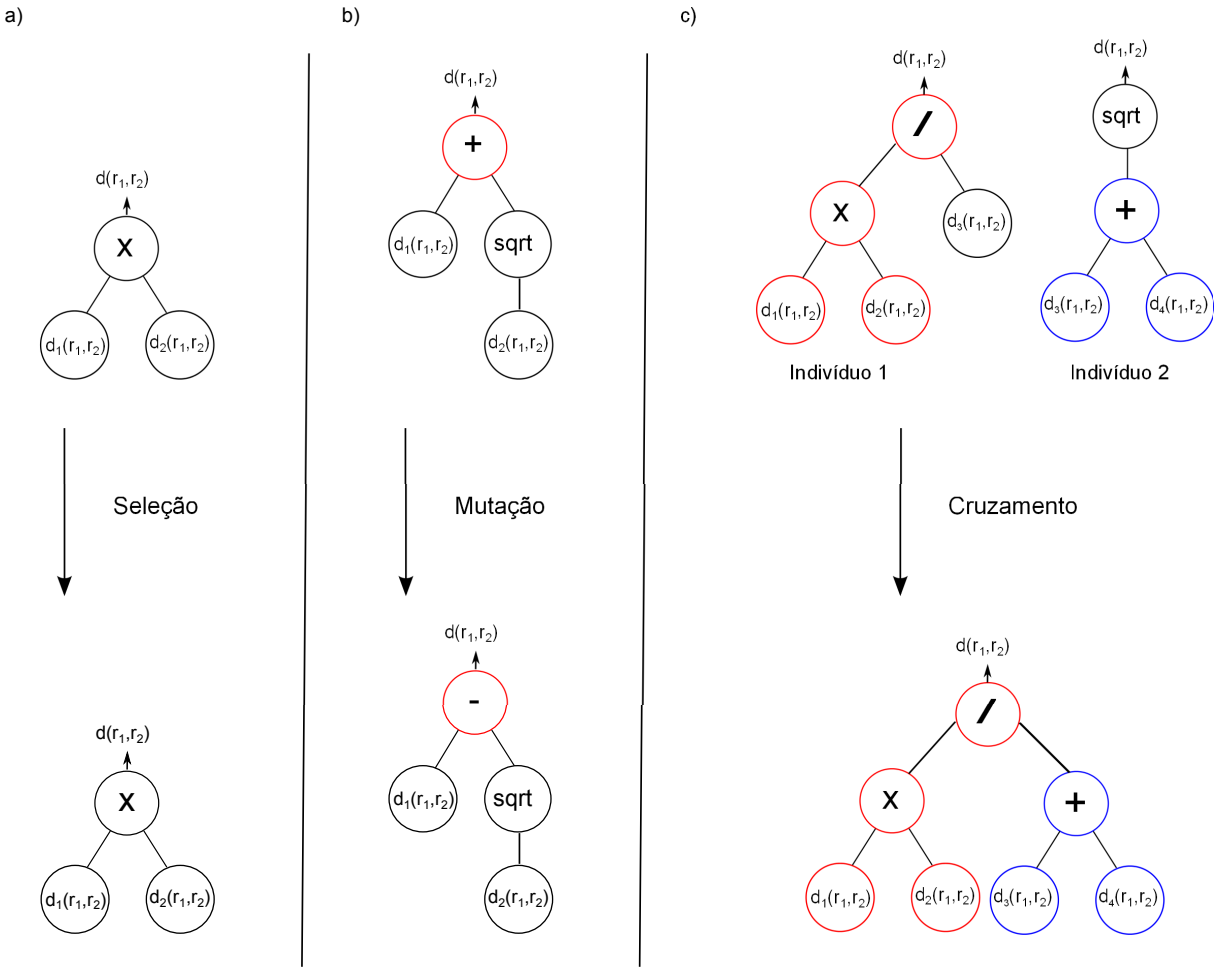


Figura 2.2: Operações genéticas: a) seleção; b) mutação; c) cruzamento.

gerações e os operadores e funções que constituem os indivíduos.

O Algoritmo 1 mostra o funcionamento típico de uma máquina de aprendizado que utiliza PG. Começando com um conjunto de dados de treinamento contendo exemplos, o motor de PG começa sua execução gerando uma grande população de indivíduos aleatoriamente combinados. Essas funções são então avaliadas. Se o critério de parada não for encontrado, as transformações genéticas serão executadas para criar e evoluir até a próxima geração iterativamente. Finalmente, depois de um número pré-definido de gerações, o indivíduo com a melhor nota é selecionado.

Em [7], Calumby et al. utilizam um modelo de programação genética para agregar descritores de imagens, que é baseado em diferentes modalidades e tipos de evidências, e que obtêm um novo descritor composto que é utilizado na tarefa de recuperação de imagens por conteúdo. Andrade et al. [1] aplicam programação genética ao problema de

Algoritmo 1 Arcabouço de PG

1: Gerar uma população inicial de "árvores de similaridade"
2: **Enquanto** número de gerações $\leq N_{gen}$ **Faça**
3: Calcular a nota de cada árvore de similaridade
4: Guardar as $N_{melhores}$ árvores de similaridade
5: **Para** todos os $N_{melhores}$ indivíduos **Faça**
6: Reprodução
7: Cruzamento
8: Mutação
9: **Fim Para**
10: **Fim Enquanto**
11: Seleciona a "melhor árvore de similaridade" (a melhor árvore de similaridade da última geração)

recuperação de imagens e vídeo por conteúdo, e fazem uma análise de como a combinação de diferentes tipos de descritores, como locais e globais, afetam a solução encontrada.

Em [29], o autor apresenta dois arcabouços de Programação Genética que utilizam realimentação do usuário para aprender uma função para recuperação de imagem por conteúdo. O primeiro utiliza apenas os exemplos de imagens rotuladas pelo usuário como relevantes (positivos) na próxima etapa do treinamento do algoritmo; e o segundo utiliza os casos positivos (revelantes) e negativos (não relevantes). Esses arcabouços combinam descritores de cor, forma e textura para encontrar funções de similaridade que melhor se adequam à pesquisa realizada pelo usuário e aos seus critérios do que é relevante ou não. Como resultado do treinamento, os autores utilizam um conjunto de boas funções encontradas, ao contrário da maioria das abordagens, que só utilizam a melhor, e realizam uma votação para enfim definir se uma imagem é relevante ou não.

Em [62], o método apresentado por Ferreira et al. [29] é adaptado para o reconhecimento de regiões em imagens de sensoriamento remoto. Os autores utilizam descritores de imagem, que por sua vez utilizam características espectrais e de textura das imagens para o aprendizado. Após definir as regiões na imagem, o sistema então segmenta a imagem e a vetoriza.

Já em [17], Mossri et al. geram funções de ranqueamento de documento adaptadas à coleção utilizando programação genética. O sistema usa diversas métricas para medir a importância de um documento, dada uma consulta, como a frequência do termo no documento e na coleção, combinando-as utilizando PG. Experimentos mostram que os resultados encontrados superam as métricas largamente difundidas na literatura.

2.4 Floresta de Caminhos Ótimos (OPF)

Encontrar um bom equilíbrio entre eficiência e eficácia ao utilizar algoritmos de aprendizado em grande quantidade de dados é sempre um desafio. Nos últimos tempos uma solução que vem obtendo bons resultados é a Floresta de Caminhos Ótimos (OPF - *Optimum–Path Forest*) [20], e que é muito mais eficiente que algoritmos tradicionais, como SVM e Naïve Bayes. Esse algoritmo mapeia os exemplos de treinamento em um grafo e pode ser utilizado como uma técnica de aprendizado supervisionado [56] ou não supervisionado [60].

Essa técnica, em sua forma supervisionada, interpreta os exemplos de treinamento como nós em um grafo, e as arestas como uma relação entre os exemplos. Toda aresta nesse grafo possui um valor dado por uma função de conectividade, e com base nesses custos, pode ser encontrada uma floresta de caminhos ótimos, em que cada árvore dessa floresta irá representar uma das classes. Essa metodologia supõe que a função de conectividade seja transitiva; assim, quaisquer dois exemplos pertencentes à mesma classe serão ligados nessa floresta por uma cadeia de exemplos próximos [55]. A atribuição de novos exemplos às classes do conjunto de treinamento é eficientemente realizada utilizando a distância do novo exemplo e as florestas existentes.

O Algoritimo 2 descreve o processo de treinamento do OPF. O primeiro passo se refere à criação de uma representação em grafos para os exemplos de treinamento (linha 1). Seja T o conjunto de treinamento; é gerado um grafo completo $G = (V, A)$ com pesos nas arestas, em que cada exemplo $t \in T$ representa um vértice v no grafo, e cada arco $(v_i, v_j) \in A$ que conecta dois nós v_i e v_j, possui um custo $d(v_i, v_j)$ determinado por uma função que representa a distância entre os nós v_i e v_j, ou seja, quanto menor o valor dessa função, mais parecidos são os exemplos. Esse valor pode ser o inverso da similaridade entre os nós.

Um exemplo de mapeamento de um conjunto de treinamento em grafo é ilustrado na Figura 2.3(a), em que foram considerados cinco exemplos de treinamento, sendo dois pertencentes à classe A e um pertencente à classe B.

Algoritmo 2 Classificador OPF: Fase de treinamento

1: Gera um grafo completo $G = (V, A)$ utilizando o conjunto de treinamento T
2: Computa o conjunto de protótipos $S \subseteq V$ utilizando uma MST
3: Particiona o grafo em árvores (cujas raízes são os protótipos) computando uma floresta de custo mínimo

No próximo passo, um conjunto S de bons representantes das classes (protótipos) é obtido computando uma árvore geradora mínima (MST - *Minimum Spanning Tree*) [10] no grafo completo (linha 2). Seja $\lambda(v_i)$ uma função que retorna o rótulo da classe do exemplo

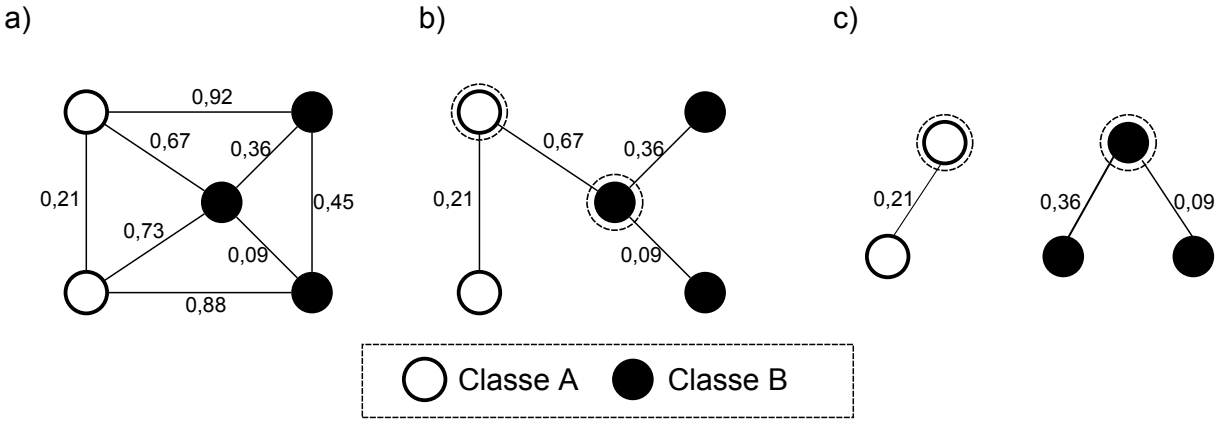


Figura 2.3: Etapa de treinamento do OPF: (a) Mapeamento de um conjunto de treinamento em um grafo completo, com duas classes representadas por nós pretos e brancos delimitados por linhas grossas, (b) MST no grafo, com destaque para os protótipos (limitados por círculos pontilhados), (c) floresta de caminhos ótimos gerada a partir dos protótipos.

representado pelo vértice v_i. Para cada arco (v_i, v_j) na MST, se $\lambda(v_i) \neq \lambda(v_j)$, então v_i e v_j são selecionados como protótipos e adicionados ao conjunto S. Na Figura 2.3(b), é ilustrada a MST encontrada no grafo da Figura 2.3(a), e são destacados os protótipos encontrados nessa MST.

Para cada nó $v_i \in T$, desejamos computar um caminho de custo mínimo com nó terminal v_i e origem no conjunto de protótipos S (linha 3). A ideia é obter uma partição ótima T, em que as diferentes classes representadas pelos protótipos em S irão competir entre si. O uso de protótipos, portanto, evita erros na classificação causados por um caminho ótimo de uma classe diferente. Todo nó de treinamento $v_i \in T$ será atribuído para a classe do protótipo mais fortemente conectado. Esta partição é computada como uma floresta de caminho ótimo P, um grafo acíclico que armazena todos os caminhos ótimos em um mapa de predecessores P. Mais precisamente, para cada nó $v_i \in T \setminus S$, $P(v_i)$ denota o nó predecessor no caminho ótimo de S, onde $P(v_i) = \text{nil}$ para cada nó $v_i \in S$. Na Figura 2.3(c), são apresentadas as árvores geradas pelos protótipos que representam a Classe A e a Classe B de nosso exemplo.

O Algoritimo 3 mostra a fase de classificação do OPF. Seja $D = \{v_1, v_2, \ldots, v_k\}$ um conjunto de k novos elementos a serem rotulados. Na fase de classificação, para todo $v_i \in D$, é necessário calcular todos os caminhos ótimos em cada árvore, com origem no conjunto de protótipos S e com terminal v_i (linha 2). Na Figura 2.4(a), um novo elemento é adicionado ao grafo, e seu caminho de custo mínimo em cada árvore gerada pela etapa de treinamento é encontrado.

De todos os caminhos encontrados, é então identificado o nó de treinamento $v\star \in T$

Algoritmo 3 Classificador OPF: Fase de classificação

1: **Para** cada $v_i \in D$ **Faça**
2: Calcular o caminho ótimo de v_i para cada protótipo em S
3: Coloque v_i na árvore cuja raiz (protótipo $v_s \in S$) esteja conectado a v_i pelo caminho ótimo
4: Atribua v_i para a mesma classe que contém v_s
5: **Fim Para**

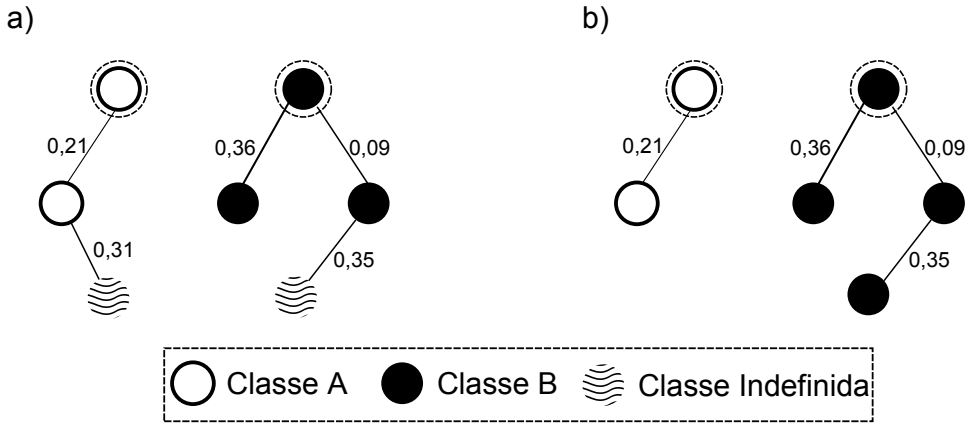


Figura 2.4: Etapa de classificação do OPF: (a) Caminho ótimo do novo elemento em cada árvore da floresta, (b) Caminho ótimo na floresta é encontrado, e novo elemento classificado como Classe B.

que fornece um caminho de custo mínimo de v_i para qualquer nó em S. O nó $v\star$ é o predecessor $P(v_i)$ no caminho ótimo com origem em S e terminal v_i (linha 3). O exemplo v_i é então classificado como $\lambda(v\star)$ (linha 4). Na Figura 2.4(b), o algoritmo verifica que o custo do caminho, entre a raiz e o novo elemento a ser classificado, na árvore que representa a classe A é de 0,52 (0,31+0,21), e na árvore B é de 0,44 (0,35+0,09); então o novo elemento é classificado como pertencente à classe B.

No próximo capítulo, apresentamos em detalhes como modelamos o problema da desambiguação de nomes de autores usando Programação Genética, Floresta de Caminhos Ótimos e realimentação de relevância.

Capítulo 3

Método Proposto

A abordagem proposta explora algumas das estratégias utilizadas por métodos de agrupamento de autores e atribuição de autores. Por exemplo, nós utilizamos uma máquina de aprendizado para aprender funções de similaridade para referências, e um sistema de classificação supervisionado para atribuir referências para os autores. A novidade está nos métodos de aprendizado de máquina empregados para implementar cada estratégia, e na combinação dessas estratégias com exemplos de treinamento gerados por realimentação do usuário.

Utilizamos Programação Genética (PG) para aprender funções de similaridade para as referências, e um classificador Floresta de Caminhos Ótimos (OPF) para rotular as referências. A combinação do OPF com PG, também conhecida como GOPF (do inglês, *Genetic Programming* e *Optimum–Path Forest*), tem sido muito efetiva em abordagens com realimentação de relevância [15], atingindo melhores resultados que os métodos que utilizam apenas funções de similaridade baseadas em PG.

O método de desambiguação consiste em particionar o conjunto de referências de autores $R = \{r_1, r_2, \ldots, r_m\}$ em um conjunto $S = \{s_1, s_2, \ldots, s_n\}$, em que cada partição s_i possui todas as referências para o mesmo autor.

Apresentamos uma abordagem semi-automática e iterativa para o problema da desambiguação de nomes de autores. Nossa estratégia se baseia na utilização de realimentação de relevância, que permite a atribuição correta de referências ambíguas para os autores correspondentes. Nós propusemos uma abordagem com duas etapas para criar essa partição, como ilustrado na Figura 3.1. O primeiro passo é não supervisionado e possui como objetivo automatizar a criação dos exemplos de treinamento para serem utilizados nos passos seguintes (o módulo rotulado como A na Figura 3.1). A segunda etapa segue a realimentação de relevância, e é supervisionada e interativa (caixa tracejada B). O usuário interage com o sistema de desambiguação provendo realimentação em referências ambíguas (seta rotulada como D). O sistema de desambiguação, por sua vez, treina um

classificador baseado na realimentação do usuário e nos exemplos de treinamentos obtidos na primeira etapa. O sistema então utiliza o modelo de classificação gerado para relacionar cada referência da entrada para um autor. As fases de treinamento e classificação são realizadas no módulo C. Esta segunda etapa é executada iterativamente até que um critério de parada seja atingido, por exemplo, com um número pré-definido de iterações. As próximas seções apresentam ambas as etapas com mais detalhes.

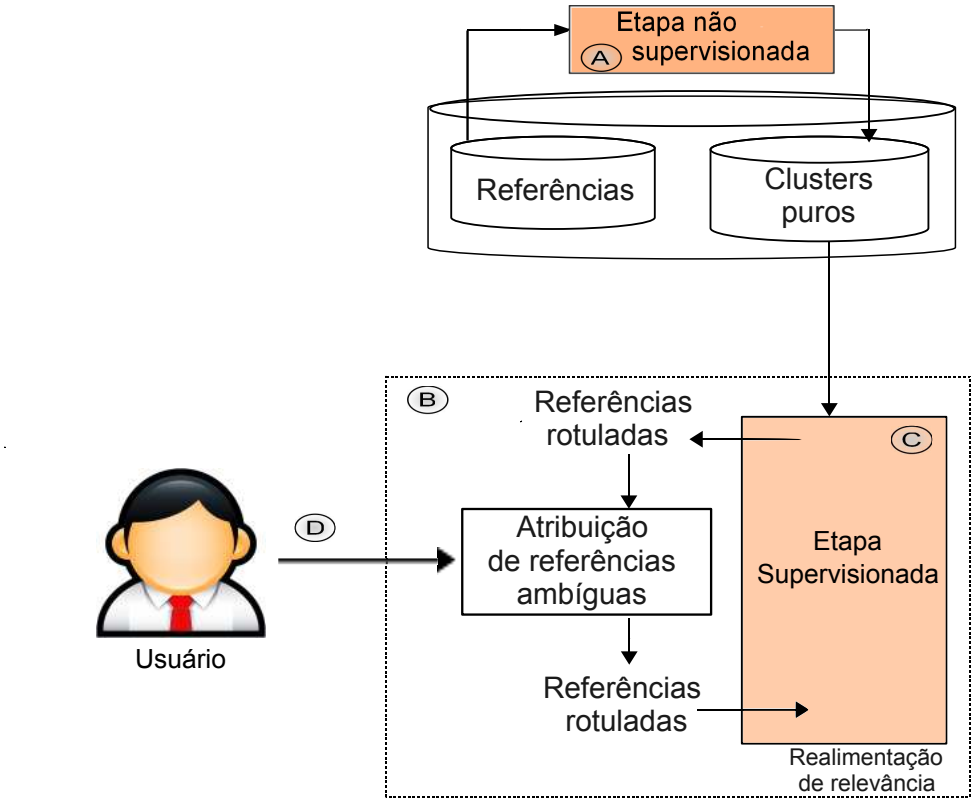


Figura 3.1: Método para desambiguação de nomes em duas etapas.

3.1 Etapa não-supervisionada

Esta etapa é inspirada no método proposto em [25,28] para gerar automaticamente *clusters* de referências "puros" para serem utilizados como conjunto de treinamento. Dada uma referência, ela deve ser muito similar às outras do mesmo *cluster*, e muito diferente das referências de outros *clusters*. Um *cluster* é *puro* se ele contém principalmente e idealmente só referências para o mesmo autor.

Ferreira et al. [25,28] propuseram a utilização de grafos de coautoria para criar *clusters*

puros: duas referências pertencem ao mesmo *cluster* se elas possuem pelo menos um coautor em comum. Os autores se basearam na observação de que raramente dois autores com nomes ambíguos compartilham o mesmo coautor, ou possuem dois coautores diferentes que também possuem nomes ambíguos. Essa estratégia, entretanto, pode gerar *clusters* fragmentados, ou seja, referências distintas para o mesmo autor podem ser colocadas em diferentes *clusters*. Os autores propuseram uma estratégia, descrita no Algoritmo 4, que é baseada no tamanho do *cluster* e em suas similaridades, para descartar *clusters* fragmentados [25, 28] do conjunto de treinamento que será utilizado no próximo passo, pois isso também irá causar fragmentação na próxima etapa do método, gerando um efeito indesejado.

Algoritmo 4 Gerando conjunto de treinamento automaticamente

Entrada: Lista de referências ambíguas L
Saída: Conjunto de treinamento T, Conjunto de teste D
1: $C = HeuristicaCoautores(L)$
2: T,$D = SelecionaClustersSignificativos(C)$
3: **Retorne** T,D

A heurística baseada em coautores, que é aplicada para a geração automática do conjunto de treinamento, é descrita no Algoritmo 5. O método começa percorrendo todas as referências da lista ambígua recebida como entrada (linha 2), e vai comparando o nome dos coautores das referências (linha 3) e agrupando-as em *clusters* (linha 4).

Algoritmo 5 Heurística de coautores

Entrada: Lista de referências ambíguas L
Saída: Conjunto de *clusters* C
1: $C = \emptyset$
2: **Para** toda referência $r_i \in L$ **Faça**
3: **Se** existe um *cluster* $c_n \in C$ que contém uma referência r_j que compartilha um coautor com r_i **então**
4: Adiciona r_i ao *cluster* c_n
5: **Senão**
6: Cria um novo *cluster* c_i contendo a referência r_i
7: Adiciona c_i em C
8: **Fim Se**
9: **Fim Para**
10: **Retorne** C

Ao comparar os coautores, estamos descartando o nome que as referências possuem em comum para estarem na lista ambígua; dessa forma, duas referências somente serão

agrupadas no mesmo *cluster* quando possuírem dois nomes de autores similares. Quando nenhum *cluster* existente possui uma referência com nome de coautor similar, um novo *cluster* é criado (linha 6) e a referência que está sendo processada é adicionada a ele (linha 7). Essa heurística não garante que os clusters serão 100% puros, porém essas imperfeições podem ser corrigidas com a interação do usuário em outra etapa de nosso método.

Com essa heurística, serão construídos *clusters* puros, porém não gostaríamos de incluir no conjunto de treinamento mais de um *cluster* representando o mesmo autor. Para reduzir as chances disso acontecer é aplicada a estratégia definida no Algoritmo 6, em que selecionamos os maiores *clusters* que não possuem distância abaixo de um determinado limiar t, definido por Ferreira et al. [28].

Algoritmo 6 Selecionando *clusters* mais significativos

Entrada: Conjunto de *clusters* C
Saída: Conjunto de treinamento T, Conjunto de teste D

1: $T = \emptyset$
2: $D = \emptyset$
3: Ordena C em ordem decrescente de quantidade de referências.
4: **Para** todo *cluster* $c_i \in C$, seguindo a ordenação feita no passo anterior **Faça**
5: **Se** existe um *cluster* $c_n \in T$, onde $distanciaCluster(c_i,c_n) \leq t$ **então**
6: Adiciona c_i ao conjunto D
7: **Senão**
8: Adiciona c_i ao conjunto T
9: **Fim Se**
10: **Fim Para**
11: **Retorne** T,D

O algoritmo começa com duas listas vazias que irão representar o conjunto de treinamento (linha 1) e o conjunto de testes (linha 2). Em seguida, o conjunto de *clusters* de entrada é ordenado em ordem crescente de tamanho (linha 3), e um a um, os *clusters* são processados seguindo esta ordem (linha 4). Cada *cluster* selecionado será adicionado ao conjunto de treinamento (linha 8) se e somente se ainda não existir um *cluster* com representação similar à dele dentro do conjunto de treinamento da iteração anterior (linha 5); caso exista, ele é então adicionado ao conjunto de testes (linha 6). Desta forma, o conjunto de treinamento gerado será representado por *clusters* puros e disjuntos.

A similaridade entre os *clusters* é calculada utilizando a Distância do Cosseno [2], uma medida de similaridade calculada sobre vetores e descrita na Seção 2.2. Os vetores que representam os *clusters* são definidos utilizando uma medida bem conhecida na área de recuperação de informação, baseada na frequência dos termos no documento (referência) específico e na coleção como um todo, o TF-IDF [2], considerando os termos presentes no título do trabalho e no título da publicação.

Ao final desta etapa, teremos dois conjuntos de *clusters*: um que será chamado daqui em diante de conjunto de treinamento e o outro que possuirá os *clusters* restantes e que será chamado de conjunto de teste. O conjunto de treinamento irá conter uma seleção de exemplos de *clusters* com a maior quantidade possível de referências e com um alto grau de pureza, ou seja, com alta probabilidade de que todas as referências pertencentes a cada *cluster* sejam de um mesmo autor. Já o conjunto de testes irá incluir todas as referências ainda não incluídas no conjunto de treinamento, e que serão utilizadas nos experimentos para avaliação da qualidade do método.

3.2 Etapa supervisionada

A Figura 3.2 resume a etapa supervisionada. Os elementos da figura são descritos nesta seção. Inicialmente, o sistema de desambiguação aprende funções de similaridade apropriadas baseado: (i) nos *clusters* puros produzidos pela etapa não supervisionada descrita anteriormente (a seta 1 na Figura 3.2); e (ii) na realimentação do usuário (seta 6), ou seja, as referências ambíguas que o usuário rotula. O classificador então utiliza essas funções de similaridade para determinar a autoria das referências ambíguas remanescentes (seta 5).

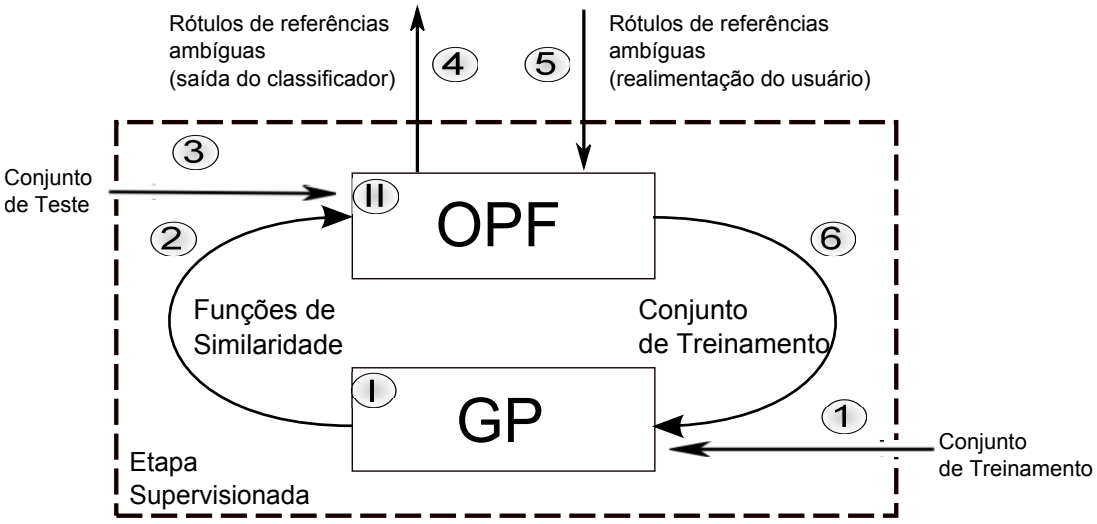


Figura 3.2: Etapa supervisionada do sistema de desambiguação.

O processo realizado para aprender as funções de similaridade é implementado utilizando um sistema de Programação Genética (PG) (módulo *B* na Figura 3.2), que foi apresentado na Seção 2.3. PG é um mecanismo evolucionário muito utilizado em problemas de otimização. Soluções de um problema alvo são representadas como indivíduos de uma população que evolui ao longo de gerações, por meio de operações genéticas (cruzamento, mutação e reprodução). Esse processo continua até que um critério de parada

seja atingido. No final, as melhores soluções (indivíduos) são selecionadas. Em nosso problema, um indivíduo representa uma função de similaridade entre referências, que pode ser utilizado para calcular a similaridade entre duas referências (seta 4).

Para utilizarmos o sistema de desambiguação em situações práticas, temos que empregar uma técnica robusta e rápida de aprendizado. Nós adotamos o classificador Floresta de Caminhos Ótimos (OPF) [55, 56] (módulo A), que foi apresentado na Seção 2.4. O OPF modela a tarefa de classificação em um problema de partição em grafo. O grafo é particionado em regiões discretas ótimas, chamadas Árvores de Caminhos Ótimos, que possuem a raiz em algum objeto representativo da classe (protótipos). O critério de otimalidade é dado pela função de custo do caminho, que guia o processo de competição entre os protótipos. Em nosso sistema de desambiguação interativo, cada classe do OPF representa um *cluster* de referências para um mesmo autor. E o valor de similaridade definido pela PG é utilizado como peso nas arestas ligando cada par de referências.

3.2.1 Descobrindo uma função de similaridade entre referências utilizando PG

Nós propusemos a utilização de Programação Genética (PG), cujo algoritmo é descrito na Seção 2.3, para descobrir uma função de similaridade entre referências apropriada. Um indivíduo PG é representado como uma árvore binária, em que todo nó interno da árvore é um operador matemático, e todo nó folha, conhecido como *terminal*, representa uma variável ou uma constante. Um exemplo de um indivíduo PG é apresentado na Figura 3.3.

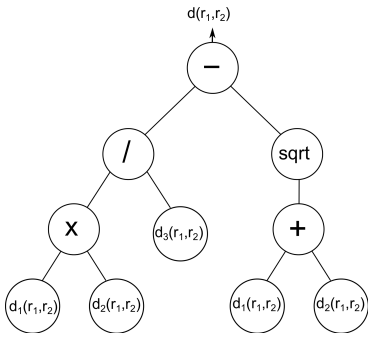


Figura 3.3: Exemplo de indivíduo PG.

Neste exemplo, um indivíduo combina os valores de três diferentes atributos (d_1, d_2 e d_3) – o valor de similaridade ou distância (por exemplo, Distância do Cosseno) entre duas referências r_1 e r_2 – em um valor único ($d(r_1, r_2)$). Este indíviduo corresponde à função $f(d_1(r_1, r_2), d_2(r_1, r_2), d_3(r_1, r_2)) = \frac{d_1(r_1, r_2) \times d_2(r_1, r_2)}{d_3(r_1, r_2)} - \sqrt{d_1(r_1, r_2) + d_2(r_1, r_2)}$.

Quanto mais alto o valor de $d(r_1, r_2)$, maior a probabilidade das referências r_1 and r_2 pertencerem ao mesmo autor.

Usando esse modelo de representação, uma população de indivíduos inicial é criada de maneira aleatória. Assim que a população inicial é gerada, o processo evolucionário começa. O primeiro passo é avaliar a qualidade da solução produzida por cada indivíduo. Para isso, uma função que mede a qualidade de um determinado indivíduo em resolver o problema é utilizada para atribuir uma nota para cada indivíduo. Depois, os indivíduos com as melhores notas são selecionados para construir a próxima geração. Depois disso, uma nova população é criada aplicando-se, para cada um dos indivíduos, transformações genéticas, como a *reprodução*, *cruzamento* e *mutação*.

O operador de reprodução seleciona os melhores indivíduos e os copia para a próxima geração. O operador de cruzamento combina o material genético de dois pais trocando uma sub-árvore de um pai com parte de outro. E o operador de mutação seleciona pontos aleatoriamente na árvore que representa alguns indivíduos e troca as sub-árvores existentes nesses pontos por novas sub-árvores aleatoriamente geradas.

Neste trabalho, nós adotamos o sistema de PG proposto em [67] para combinar funções de similaridade entre referências. A entrada para o sistema PG é um conjunto de *clusters* com referências. Esta entrada é convertida em uma lista de pares de referências, considerando todas as possíveis combinações. Um indivíduo PG é utilizado para calcular a similaridade entre duas referências, e o valor é usado para classificar todos os pares. Um bom indivíduo deve atribuir altos valores de similaridade para pares de referências que pertencem ao mesmo autor. Suponha que as referências r_1 e r_2 pertencem ao mesmo autor, por exemplo a_1, enquanto as referências r_3 e r_4 pertencem aos autores a_2 e a_3, respectivamente. Seja F uma função que recebe um par de referências (r_i, r_j) como entrada e retorna 1, se r_i e r_j pertencem ao mesmo grupo ($F((r_1, r_2)) = 1$) e 0, caso contrário ($F((r_1, r_3)) = 0$). A Tabela 3.1 apresenta um exemplo de classificação de pares de referências de acordo com um indivíduo PG, e o respectivo valor da função F.

Tabela 3.1: Qualidade dos indivíduos.

Pares de referências	Valor da similaridade PG	Valor de F
(r_2, r_3)	0,92	0
(r_1, r_2)	0,89	1
(r_1, r_3)	0,76	0
(r_3, r_4)	0,69	0
(r_2, r_4)	0,65	0
(r_1, r_4)	0,45	0

A nota de um indivíduo é calculada em função da qualidade de uma classificação de pares de referências. Uma classificação ideal deveria possuir todos os pares em que a

função F retorna 1 nas primeiras posições. Nesse caso, a precisão é máxima, ou seja, existe uma posição na classificação que separa todos os pares que pertencem ao mesmo autor ($F = 1$) dos outros pares. Em outros casos, a posição que maximiza a precisão pode ser determinada (em nossa tabela de exemplo, a segunda posição).

Essa nota é definida como o número de pares corretamente classificados, dividido pelo número total de pares. Em nosso exemplo, o número de pares corretamente classificados, ou seja, a quantidade de pares de referências acima da linha dupla que possuem valores de $F = 1$, somado ao número de pares abaixo da linha dupla que possuem valores de $F = 0$, é igual a 5, e o número total de pares é 6; logo, a nota do indivíduo é igual a $5/6 = 0,83$.

3.2.2 Classificação de referências ambíguas utilizando o classificador OPF

Utilizamos um sistema de classificação baseado no classificador Floresta de Caminhos Ótimos (OPF) [55,56] para atribuir novas referências aos autores adequados. Nesta seção, iremos definir como o conjunto de treinamento e a função encontrada pelo PG serão aplicados pelos algoritmos do classificador OPF descritos na Seção 2.4.

Seja T o conjunto de treinamento definido em termos de *clusters* puros de referências, i.e., cada *cluster* contém apenas referências para o mesmo autor. O conjunto T inclui os *clusters* definidos na etapa não supervisionada do algoritmo de desambiguação e também as referências rotuladas pelo usuário nas iterações de realimentação. É construído o grafo $G = (V, A)$, um grafo completo com pesos nas arestas, em que um conjunto de vértices de G é o conjunto T de referências, e E é o conjunto de arestas. A Figura 3.4(a) ilustra o conjunto de treinamento com 11 referências dividido em três *clusters* (três autores).

Os arcos (r_i, r_j) conectando as referências r_i e r_j no grafo G possuem custo $d(r_i, r_j)$, uma função inversamente proporcional à similaridade entre r_i e r_j. O peso $d(r_i, r_j)$ é definido por um indivíduo PG (discutido na Seção 3.2.1) que combina valores de similaridade entre as referências r_i e r_j, baseado nos nomes dos autores, título do trabalho e título da publicação. As arestas do grafo completo não são mostradas na Figura 3.4(a) para não prejudicar a visibilidade da imagem.

Seja $\lambda(r_i)$ uma função que retorna o rótulo da classe da referência r_i (o nome do autor). Para cada arco (r_i, r_j) na MST, se $\lambda(r_i) \neq \lambda(r_j)$, então r_i e r_j são selecionados como protótipos e adicionados ao conjunto S. A Figura 3.4(b) ilustra a definição dos protótipos e a Figura 3.4(c) ilustra a Floresta de Caminhos Ótimos gerada para os protótipos definidos na Figura 3.4(b).

Seja $D = \{C_1, C_2, \ldots, C_k\}$ um conjunto de k *clusters* de referências a serem rotuladas. Na fase de classificação, para todo $r_i \in C_j$ ($C_j \in D$), o caminho ótimo com terminal r_i pode ser facilmente identificado encontrando qual nó do treinamento $r\star \in T$ fornece um

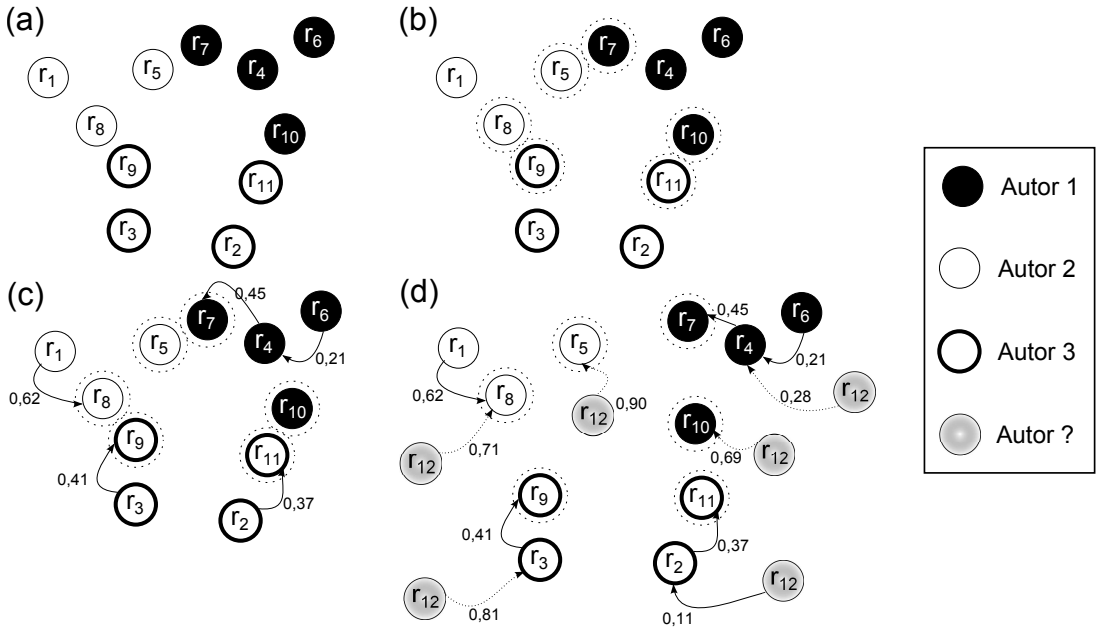


Figura 3.4: (a) Exemplo de um conjunto de treinamento em um grafo com três classes representadas por nós pretos, brancos e brancos delimitados por linhas grossas; (b) protótipos (limitados por círculos pontilhados) obtidos da MST; (c) floresta de caminhos ótimos gerada dos protótipos; e (d) classificação da referência r_{12} através de seu nó predecessor r_2 na floresta de caminhos ótimos.

caminho de custo mínimo de r_i para qualquer nó em S. O nó $r\star$ é o predecessor $P(r_i)$ no caminho ótimo com terminal r_i. A referência r_i é então classificada como $\lambda(r\star)$. A Figura 3.4(d) ilustra a fase de classificação. Neste exemplo, a referência r_{12} é atribuída para a mesma classe que r_2, o que é ilustrado na Figura 3.5.

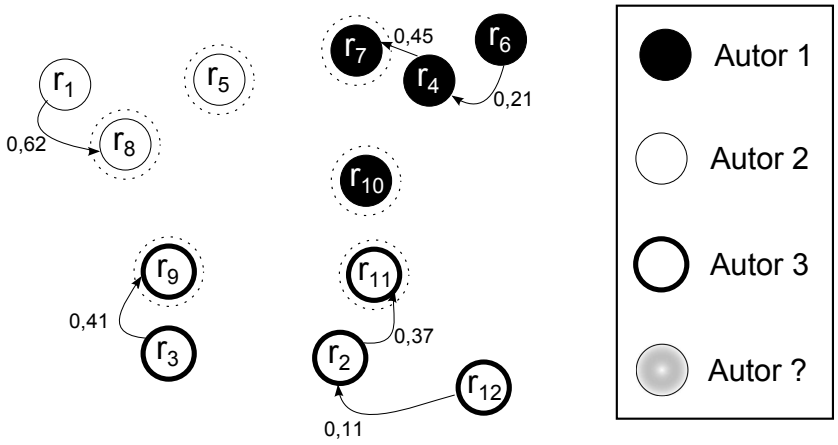


Figura 3.5: Referência r_{12} é classificada como pertencente a mesma classe que r_2.

3.2.3 Identificação de novos autores

Algumas instâncias do conjunto de treinamento não cobrem todos os possíveis autores dos grupos ambíguos, o que torna a etapa de classificação um problema de conjunto aberto. De modo a tratar esse problema, na fase de classificação do método, nós definimos um limiar para o custo τ que irá nortear a criação de novos *clusters*. O valor de τ é definido pela Equação 3.1, ou seja, τ é o custo máximo da distância entre qualquer nó r_i no conjunto de treinamento e a raiz da árvore (protótipo), representado por $r_\star$, à qual a referência r_i foi conectada pela etapa de treinamento do algoritmo:

$$\tau = \max \text{caminhoMinimo}(r_i, r_\star) \; \forall \; r_i \in T. \tag{3.1}$$

Esse valor representa o limiar entre a mais fraca conexão entre uma referência no conjunto de treinamento a seu respectivo protótipo, e relações estabelecidas por caminhos de custo maior que esse são consideradas fracas.

O Algoritmo 7 descreve como adaptamos o Algoritmo 3 para tratar o problema de classificação de conjunto aberto. Para cada referência r_i, denotamos como $d_\star(r_i)$ o custo do caminho ótimo de r_i para um protótipo $r_\star$ (linha 3). Se $d_\star(r_i) \geq \tau$, então um novo *cluster* é criado para representar a aparição de um novo autor, e a referência r_i é atribuída para este novo *cluster* (linha 6), pois a conexão da referência r_i no grafo com a árvore em que ela melhor se encaixa é mais fraca do que o necessário. Caso contrário, o algoritmo segue o fluxo normal e a referência é adicionada ao mesmo *cluster* que representa a classe $\lambda(r\star)$ (linha 8).

Algoritmo 7 Classificador OPF: Fase de classificação

1: **Para** cada $r_i \in D$ **Faça**
2: Calcular o caminho ótimo de r_i para cada protótipo em S
3: Seja $d_\star(r_i, r_s) \; \forall \; r_s \in S$
4: **Se** $d_\star(r_i) \geq \tau$ **então**
5: Adicione r_i ao conjunto de protótipos S
6: Crie um novo *cluster* que irá conter r_i
7: **Senão**
8: Atribua r_i para o mesmo *cluster* que contém r_s
9: **Fim Se**
10: **Fim Para**

Por exemplo, dado o grafo da Figura 3.5(d), se a próxima referência a ser adicionada for r_{13}, primeiro calculamos os caminhos mínimos dessa referência em cada árvore de classificação existente, como ilustrado na Figura 3.6.

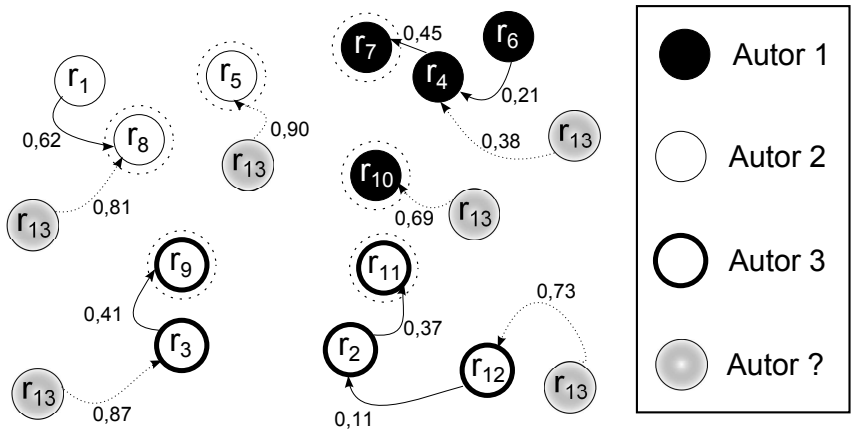


Figura 3.6: Referência r_{13} é adicionada ao grafo e o caminho em cada árvore encontrado.

Identificamos que o caminho de custo ótimo de r_{13} até algum protótipo é 0,69, através do caminho (r_{10},r_{13}). Observando a Figura 3.5, podemos calcular que o valor de τ é 0,66, pois este é o custo do caminho (r_6,r_7), que é o caminho de custo máximo em todo o grafo. Dessa forma, como $d_\star(r_{13})$ (0,69) é maior que τ (0,66), um novo *cluster* C_j é criado e a referência r_{13} é adicionada a C_j e passa a fazer parte do conjunto de protótipos S. O estado final após a criação da nova classe é ilustrada na Figura 3.7.

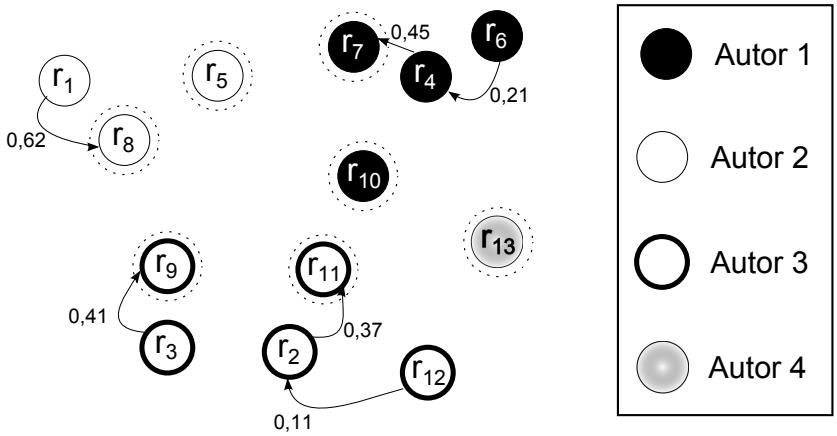


Figura 3.7: Referência r_{13} é definida como um novo protótipo e a classe Autor 4 é criada.

3.2.4 Seleção de casos para a realimentação do usuário

Nossa estratégia para encontrar os casos mais informativos para a interação com o usuário é selecionar uma quantidade k, definida pelo usuário, de referências que estão mais próximas ao limiar τ. O objetivo dessa escolha é selecionar os k casos que ficaram mais próximos da fronteira entre as possíveis decisões que o algoritmo precisa tomar. Assim teremos os k novos exemplos confiáveis para serem utilizados como parte do conjunto de treinamento da próxima iteração que irão maximizar o potencial de aprendizagem.

Desse modo, o usuário é sempre guiado a rotular as referências que serão mais informativas para o motor de aprendizado. Ao compararmos com os métodos puramente supervisionados, que atualmente são considerados o estado da arte para a solução do problema da desambiguação de nomes de autores, nosso método propicia ao usuário uma grande economia com relação ao tempo dispensado à criação de exemplos para o treinamento do algoritmo.

Por exemplo, seja $\tau = 0{,}69$, com $k = 2$, e durante uma determinada iteração do algoritmo sejam classificadas as referências r_{14},r_{15},r_{16},...,r_{23} e os respectivos caminhos de custo mínimo no grafo de classificação da Tabela 3.2. Os dois exemplos que o usuário precisaria rotular são as referências r_{15} e r_{18}, pois são as que mais se aproximaram do limiar τ.

Tabela 3.2: Valores do caminho de custo mínimo para cada referência classificada.

Referência	r_{14}	r_{15}	r_{16}	r_{17}	r_{18}	r_{19}	r_{20}	r_{21}	r_{22}	r_{23}
$d_\star(r_i)$	0,85	**0,65**	0,54	0,40	**0,72**	0,90	0,47	0,56	0,59	0,37

No próximo capítulo, são apresentados os experimentos realizados acompanhados de uma análise comparativa entre os resultados obtidos pelo nosso método e pelos outros métodos. Além disso, a configuração experimental utilizada é apresentada.

Capítulo 4

Experimentos e Resultados

Este capítulo apresenta os experimentos realizados para avaliar o método de desambiguação proposto neste trabalho.

4.1 Configuração experimental

Nesta seção, são apresentadas todas as informações necessárias para entender e possibilitar a reprodução dos experimentos cujos resultados iremos discutir na Seção 4.2.

4.1.1 Coleções

Para propósitos de avaliação, nós utilizamos duas coleções derivadas da *DBLP Computer Science Bibliography* (DBLP)[1].

DBLP

A primeira coleção derivada da DBLP, de agora em diante referida como DBLP, contém 4287 referências associadas com 220 autores distintos, o que significa uma média de aproximadamente 20 referências por autor. Pequenas variações dessa coleção vêm sendo utilizadas em diversos outros trabalhos [25, 27, 28, 36–38, 57]. A versão original foi criada por Han et al. [2004], que rotularam as referências manualmente. Para isso, utilizaram as páginas dos autores na internet, nome de afiliação, e-mail e nomes de coautores no formato completo, e também enviaram emails para alguns autores para confirmar algumas autorias, eliminando as referências para as quais eles não obtiveram informação suficiente para identificar o autor correto. Eles também substituíram os títulos das publicações abre-

[1]Disponível em `http://www.informatik.uni-trier.de/~ley/db/` (Acessado em Jul. 2013).

viados por seus nomes por extenso, obtidos da DBLP. Nós utilizamos 11 grupos ambíguos extraídos por Han et al. [2004], com algumas correções feitas por Ferreira et al. [2010][2].

KISTI

A segunda coleção derivada da DBLP, de agora em diante referida como KISTI, foi construída pelo *Korea Institute of Science Technology Information* [40] para a desambiguação de nomes de autores homônimos em inglês. Os 1000 mais frequentes nomes de autores nos registros de citação da DBLP no final de 2007 foram obtidos, juntamente com seus registros. Depois disso, para cada nome de autor em cada registro de citação, uma referência foi construída. Para desambiguar essa coleção, os autores buscaram no mecanismo de busca do *google* o sobrenome do autor juntamente com o título do trabalho, visando encontrar páginas de publicações pessoais. As primeiras 20 páginas web encontradas para cada consulta foram manualmente verificadas para identificar a página de publicações pessoal para cada registro de autoria. Essas páginas identificadas foram então utilizadas para desambiguar os registros. Esta coleção possui 37613 registros de citações, 881 grupos de pessoas com o mesmo nome e 6921 autores[3]. Pelo fato de executarmos um grande número de experimentos e comparações, em nossa avaliação nós utilizamos somente os 40 grupos ambíguos mais frequentes, ou seja, os grupos com o maior número de autores ambíguos. Esses grupos são potencialmente os mais difíceis para desambiguarmos. Neste subconjunto, existem 7841 registros de citações, 40 grupos de pessoas com o mesmo nome e 1132 autores, com uma média de 28,3 diferentes autores ambíguos por grupo (com um máximo de 59).

4.1.2 Métodos de referência

Utilizamos como base para comparação o método proposto por Ferreira et al. [25], referenciado de agora em diante como SANDRef. Com essa técnica é obtida a solução que é considerada o estado da arte para o problema da desambiguação de nomes de autores utilizando realimentação de relevância. SANDRef obtém melhores resultados que outros métodos baseados em SVM [39], ou Naïve Bayes (NB) [36], também utilizados como base para comparação. Nós também incluímos o método SAND [28] como base de comparação. SAND é similar ao SANDRef, porém ele não considera nenhuma informação de realimentação.

Em todos os experimentos, utilizamos o melhor resultado obtido para o método SANDRef, que foi atingido após rotular o máximo de referências definido pelo método, que é

[2]Disponível em `http://www.lbd.dcc.ufmg.br/lbd/collections/disambiguation` (Acessado em Jul. 2013).

[3]Disponível em `http://www.kisti.re.kr` (Acessado em Jul. 2013).

de em média 6 por grupo ambíguo.

4.1.3 Métricas de avaliação

Para avaliarmos a efetividade de nossa abordagem de desambiguação de nomes de autores, utilizamos a medida K e a medida F1. Estas métricas permitem atribuir uma pontuação numérica à qualidade dos *clusters* gerados em uma solução. Elas também já foram utilizados antes para medir a qualidade de métodos de desambiguação [25, 27].

Antes de definirmos as duas métricas, entretanto, é necessário introduzir dois conceitos: *clusters* teóricos e *clusters* empíricos. Os *clusters* teóricos representam os agrupamentos ideais de referências de autores que seriam gerados por uma atribuição manual, ou seja, *clusters* totalmente puros e coesos. Já os *clusters* empíricos são os *clusters* gerados pelos métodos que visam automatizar o processo de desambiguação de nomes de autores.

Métrica K

A medida K [44] calcula um valor que leva em consideração duas grandezas inversamente proporcionais: a média de pureza dos *clusters* (ACP) e a média da pureza dos autores (AAP). Dado um grupo ambíguo, ACP avalia a pureza dos *clusters* empíricos com respeito aos *clusters* teóricos para este grupo ambíguo. Assim, se o *cluster* empírico é puro, ou seja, contém apenas referências associadas ao mesmo autor, o valor correspondente do ACP será 1. O ACP é definido como

$$\text{ACP} = \frac{1}{N} \sum_{i=1}^{e} \sum_{j=1}^{t} \frac{n_{ij}^2}{n_i}, \tag{4.1}$$

onde N é o número total de referências em um grupo ambíguo, t é o número de *clusters* teóricos no grupo ambíguo, e é o número de *clusters* empíricos no grupo ambíguo, n_i é o número total de referências no *cluster* empírico i e n_{ij} é o número total de referências no *cluster* empírico i que também estão no *cluster* teórico j.

Para um dado grupo ambíguo, AAP avalia a fragmentação dos *clusters* empíricos em relação aos *clusters* teóricos. Se os *clusters* empíricos não estão fragmentados, o valor correspondente de AAP será 1. AAP é definido como

$$\text{AAP} = \frac{1}{N} \sum_{j=1}^{t} \sum_{i=1}^{e} \frac{n_{ij}^2}{n_j}, \tag{4.2}$$

onde n_j é o número total de referências no *cluster* teórico j.

A medida K é definida como a média geométrica entre os valores de ACP e AAP

$$K = \sqrt{\text{ACP} \times \text{AAP}}, \tag{4.3}$$

que avalia a pureza e a coesão dos *clusters* empíricos extraídos por cada método.

Métrica F1

F1 pareada (pF1) é a métrica F1 [59] calculada utilizando precisão e revocação pareadas. A precisão pareada (pP) é calculada como $pP{=}\frac{a}{a+c}$, onde a é o número de pares de referências em um *cluster* empírico que estão (corretamente) associados com o mesmo autor, e c é o número de pares de referências em um *cluster* empírico que não correspondem ao mesmo autor. Revocação pareada (pR) é calculada como $pR{=}\frac{a}{a+b}$, onde b é o número de pares de referências associados com o mesmo autor que não estão no mesmo *cluster* empírico. A medida F1 é definida como:

$$p\text{F1} = 2 \times \frac{pP \times pR}{pP + pR}. \tag{4.4}$$

A precisão representa a eficácia de um método em identificar quando duas referências não representam o mesmo autor, e a revocação representa a eficácia de um método em identificar quando duas referências representam o mesmo autor. Em um caso hipotético em que um método gera somente um cluster com todas as referências, a precisão seria 0 e o revocamento 1. Em outro caso hipotético, em que um método gera um cluster para cada referência, a precisão seria 1 e a revocação 0. Dessa forma, utilizar a precisão ou a revocação isoladamente não representaria a real qualidade de um método, então utiliza-se a medida F1 que combina esses dois objetivos conflitantes.

4.1.4 Parâmetros do método de desambiguação

Um algoritmo baseado em programação genética possui diversos parametros. Na Tabela 4.1 apresentamos os valores definidos empiricamente e que utilizamos em nossos experimentos. Nós não identificamos variação significativa em nossos resultados com outras configurações, portanto escolhemos utilizar a descrita na Tabela 4.1, que apresentou melhor tempo de execução entre todas as avaliadas.

Os indivíduos PG combinam valores de similaridade entre referências definidos em termos do título do trabalho e da publicação. Esses valores são obtidos calculando a similaridade do cosseno, a similaridade Jaccard, Jaro-Winkler e a Distância de Levenshtein nesses campos das referências.

Tabela 4.1: Valores dos parâmetros para o modelo PG.

Parâmetros PG	Valor
Método para população inicial	gerado aleatoriamente
Tamanho da população inicial	200
Altura máxima do individuo	6
Número de iterações	10
Operadores	$\times$, $\div$, $+$, cos(), exp()

4.1.5 Protocolo experimental

Para efeito de comparação, nós utilizamos o mesmo protocolo experimental empregado em [25]. Utilizamos uma seleção aleatória de 50% de cada grupo ambíguo (conjunto de teste). A parte restante da coleção foi utilizada para treinamento pelos métodos do SVM [39] e NB [36], pois estes são métodos supervisionados que requerem dados de treinamento. Cada experimento foi executado 10 vezes, e o valor médio de cada método aplicado para o conjunto de teste foi computado com um intervalo de confiança de 95%.

Nós simulamos a presença de um usuário ideal realimentando o sistema, uma estratégia típica para avaliar sistemas de recomendação de relevância [15, 16, 18]. Em nosso caso, o usuário ideal sempre atribui a referência ambígua para o autor correto em cada iteração. Como nosso sistema não possui interface para testes com um usuário real, simulamos o usuário ideal utilizando os rótulos que os dados utilizados nos experimentos possuem; em outras palavras, esse "usuário" sempre irá rotular corretamente as referências.

4.2 Resultados

As Figuras 4.1 e 4.2 mostram os valores de K e $F1$ para os métodos avaliados na coleção DBLP. Nessas figuras, SAND e SANDRef são as bases para comparação (veja Seção 4.1.2); GOPF representa o método de desambiguação GP-OPF proposto *sem* realimentação do usuário; GOPFRef-3 e GOPFRef-5 representam o método de desambiguação proposto *com* realimentação de relevância considerando, respectivamente, 3 e 5 referências rotuladas em cada iteração por grupo ambíguo.

Como pode ser observado, quando o número de iterações aumenta, o método de desambiguação proposto atinge melhores resultados. Nós também notamos que quanto mais referências são rotuladas por iteração, melhores são os resultados. De fato, a partir da sexta iteração, GOPFRef-5 é estatisticamente melhor (p-*value* $\leq 0,05$) que todas as bases de comparação. Nota-se também que GOPFRef-5 já é estatisticamente melhor que SANDRef, a melhor base para comparação, na quarta iteração. Nesse ponto, entretanto,

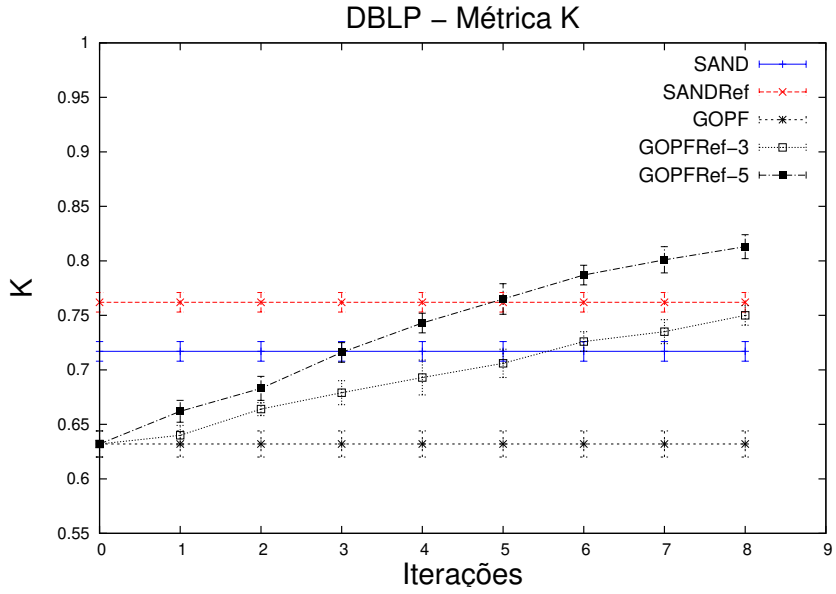


Figura 4.1: Valores de K para os métodos de desambiguação avaliados na coleção DBLP.

apenas 20 referências por grupo ambíguo (4 iterações × 5 referências por iteração) foram rotuladas no método GOPFRef-5.

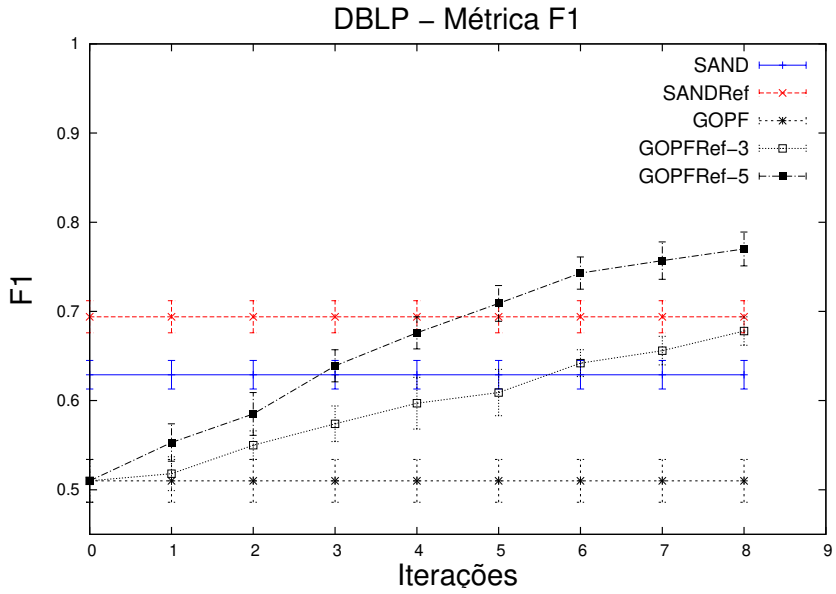


Figura 4.2: Valores de F1 para os métodos de desambiguação avaliados na coleção DBLP.

As Figuras 4.3 e 4.4 apresentam os valores de medida de K e $F1$ para os métodos avaliados na coleção KISTI. Os resultados são similares aos encontrados na coleção DBLP. Nota-se, entretanto, que para essa base de dados, com apenas 3 iterações (15 referências

rotuladas por grupo ambíguo), GOPFRef-5 é estatisticamente melhor ($p\text{-}value \leq 0,05$) que todos os métodos da base de comparação para a métrica K. Para a medida F1, GOPFRef-5 atinge os mesmos resultados que SANDRef na terceira iteração, mas é melhor da quarta em diante.

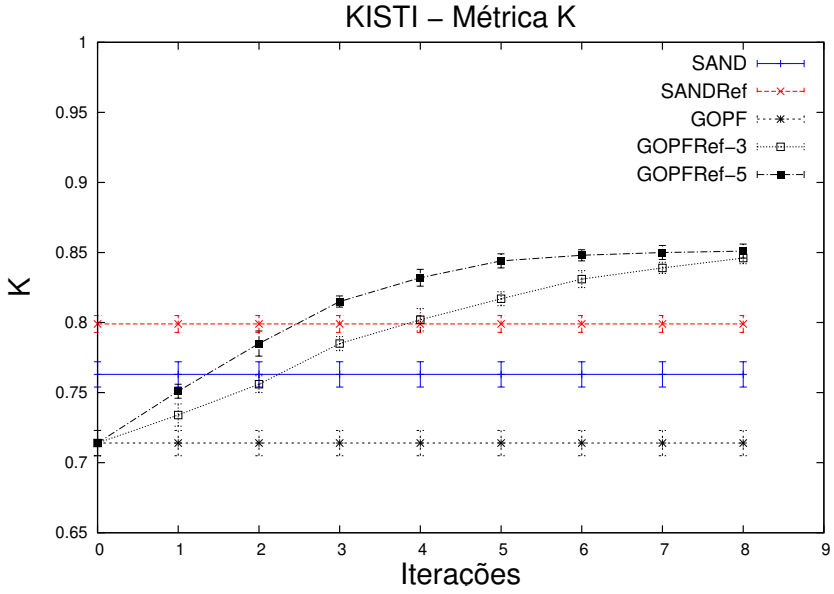


Figura 4.3: Valores de K para os métodos de desambiguação avaliados na coleção KISTI.

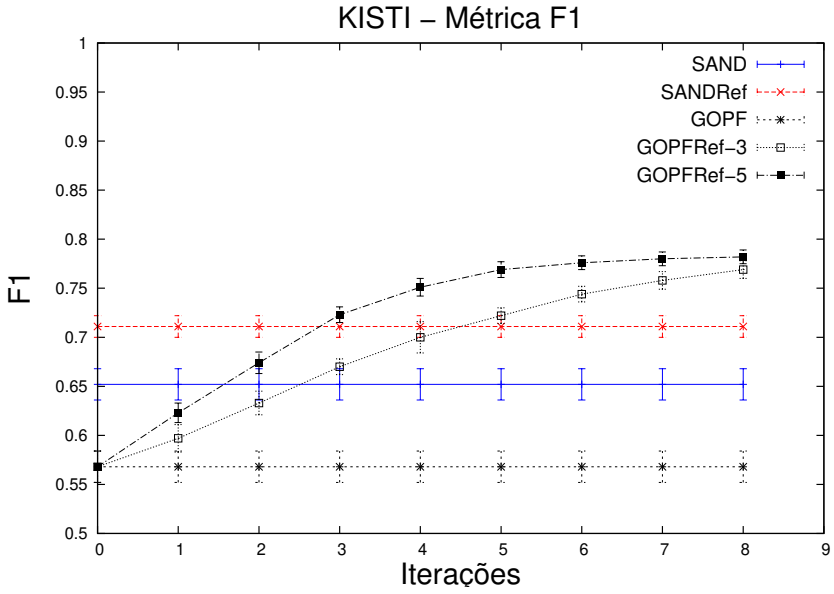


Figura 4.4: Valores de F1 para os métodos de desambiguação avaliados na coleção KISTI.

A Tabela 4.2 agrega as médias dos resultados (com desvio padrão) dos métodos propostos e todos os métodos utilizados como base para comparação nas coleções DBLP e KISTI. Nessa tabela, nós também exibimos os resultados dos dois métodos baseados em SVM [39] e Naïve Bayes (NB) [36], treinados com 50% de cada coleção. Os resultados dos métodos propostos consideram a execução de 8 iterações. Os métodos destacados na Tabela 4.2 indicam os melhores resultados estatísticos obtidos em cada uma das coleções, e é possível observar que GOPFRef-5 obtém melhor eficácia em ambas.

Tabela 4.2: Comparação dos métodos propostos com as bases de comparação nas bases do DBLP e do KISTI (melhores resultados em negrito).

Collection	Method	K	F1
DBLP	SAND	0,717 (0,013)	0,629 (0,022)
	SANDRef	0,762 (0,013)	0,694 (0,025)
	SVM	0,799 (0,008)	0,721 (0,010)
	NB	0,736 (0,009)	0,647 (0,012)
	GOPFRef-3	0,750 (0,009)	0,678 (0,016)
	GOPFRef-5	**0,813 (0,011)**	**0,770 (0,019)**
KISTI	SAND	0,763 (0,012)	0,652 (0,022)
	SANDRef	0,799 (0,008)	0,711 (0,016)
	SVM	0,761(0,007)	0,576 (0,006)
	NB	0,745 (0,004)	0,601 (0,006)
	GOPFRef-3	0,846 (0,004)	0,769 (0,009)
	GOPFRef-5	**0,851 (0,005)**	**0,782 (0,007**)

Nós também comparamos os métodos considerando a média dos resultados obtidos por grupo ambíguo. A Figura 4.5 apresenta os resultados referentes às métricas K na base da DBLP. Os resultados são similares aos obtidos na base de dados KISTI com a medida F1. GOPRef-5 se refere ao resultado considerando 8 iterações de realimentação de relevância.

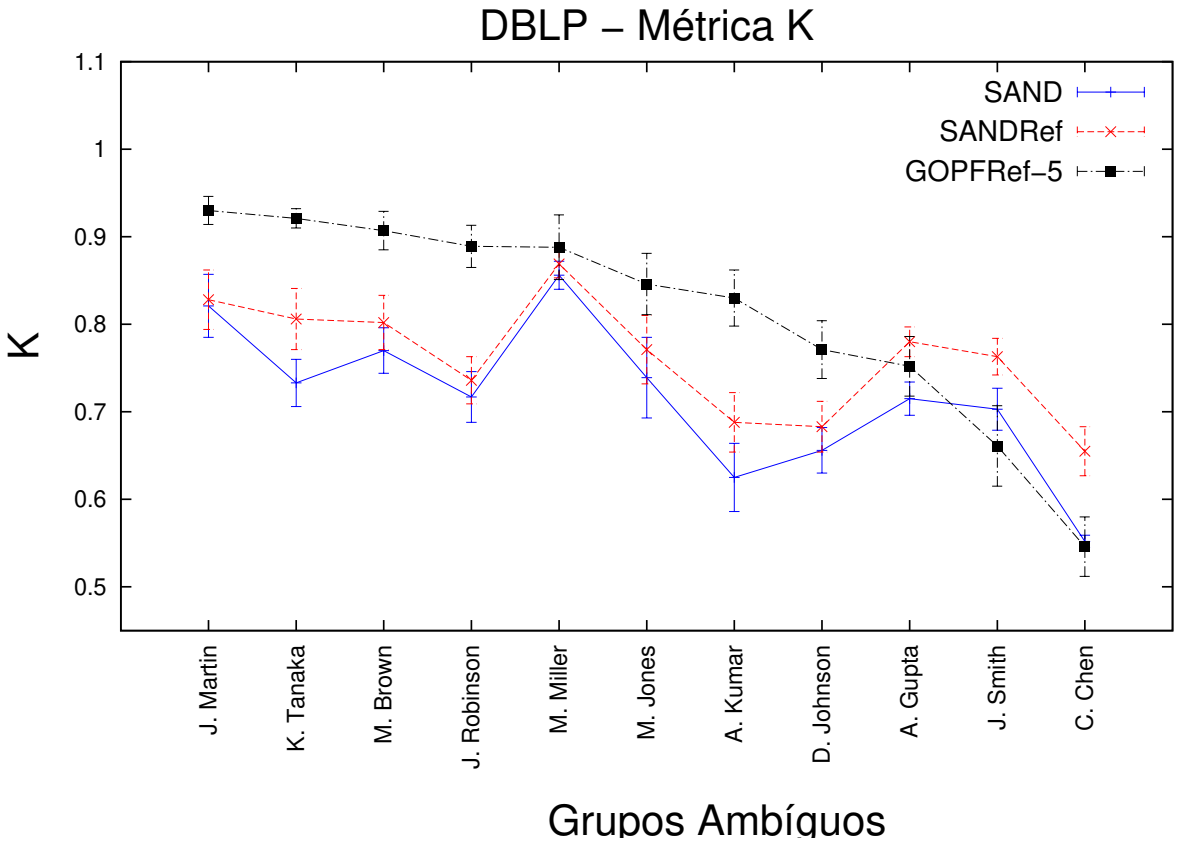


Figura 4.5: Valores de K para os métodos avaliados para cada grupo ambíguo na coleção DBLP.

Pudemos notar que GOPRef-5 superou ou igualou todos os métodos utilizados como base para comparação em 9 dos 11 grupos ambíguos. Os piores resultados foram observados para os grupos mais ambíguos (J. Smith e C. Chen). Nós acreditamos que se mais referências fossem rotuladas por iteração, ao invés de apenas 5, o resultado da desambiguação poderia ser ainda melhor. Portanto, nós também conduzimos experimentos para avaliar o impacto do número de referências rotuladas por iteração nos grupos mais ambíguos.

A Figura 4.6 mostra os resultados da métrica K para o grupo ambíguo relacionado ao autor C. Chen na coleção da DBLP. Resultados similares foram observados para os outros grupos ambíguos. Como pode ser observado, os melhores resultados foram produzidos quando mais referências são rotuladas por iteração, atingindo os melhores resultados da sexta iteração em diante, com empate estatístico com SANDRef na quarta e quinta iteração.

Outro ponto que pode ocasionar erros no grupo ambíguo C. Chen é que este grupo é referente a um nome de origem asiática, e os coautores das citações pertencentes a este grupo em sua grande maioria também são asiáticos. Nomes asiáticos têm por características serem pequenos e muito comuns; assim a heurística de coautoria, utilizada na etapa não-supervisionada do método, não é muito eficaz e o método é treinado com exemplos errados. A solução para casos como esse é aumentar a quantidade de referências rotuladas pelo usuário, desta forma o usuário é capaz de corrigir as imperfeições do conjunto de treinamento gerado.

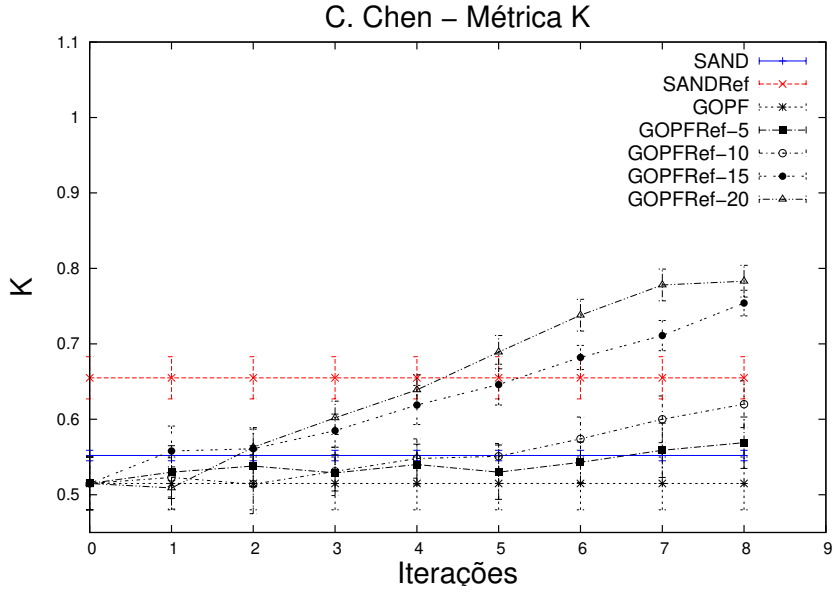


Figura 4.6: Valores da métrica K para os métodos avaliados para o grupo ambíguo relacionado ao autor **C. Chen**.

Capítulo 5

Conclusões e Trabalhos Futuros

Bibliotecas Digitais (BDs) são responsáveis por prover serviços a instituições e pesquisadores. A qualidade desses serviços é fundamental para que as análises e estudos realizados possam ser confiáveis. Um dos principais e mais complexos desafios enfrentados pelos administradores de BDs para melhorar a confiabilidade dos serviços prestados é a desambiguação de nomes de autores. Neste trabalho, nós apresentamos um novo método para desambiguação que utiliza realimentação do usuário para definir modelos de classificação que atribuem referências para autores.

Foi proposta uma abordagem na qual os usuários (por exemplo, administradores de BD) provêm a realimentação rotulando referências ambíguas, e o sistema proposto utiliza essa informação, junto com os *clusters puros* gerados pela etapa supervisionada, em um esquema de classificação híbrido. Nossa abordagem de classificação combina um sistema de Programação Genética (PG), que é utilizado para definir funções de similaridade adequadas às referências, com o classificador Floresta de Caminhos Ótimos (FCO), uma abordagem baseada em grafos que utiliza pesos nas arestas baseadas em PG para atribuir as referências inseridas para os autores corretos.

Nós conduzimos um conjunto de experimentos visando avaliar o desempenho do sistema de desambiguação proposto em comparação com os métodos do estado da arte em bases de dados bem conhecidas. Os resultados obtidos demonstram a efetividade do método proposto. Em poucas iterações, e com um baixo número de referências rotuladas por iteração, o método proposto supera os utilizados como base de comparação em até 5%. Além disso, o método proposto reduz drasticamente a quantidade de intervenção manual usualmente necessária para se obter resultados similares. Ele supera inclusive alguns métodos de desambiguação que são considerados o estado da arte [25], entre eles um que também explora alguma realimentação do usuário.

Um artigo com os resultados deste trabalho, intitulado "*A relevance feedback approach for the author name disambiguation problem*" [33], foi apresentado na *13th ACM/IEEE-*

CS Joint Conference on Digital Libraries, que foi realizada em julho de 2013 em Indianápolis/IN (EUA).

5.1 Extensões e trabalhos futuros

A partir deste trabalho, existem diversas extensões possíveis. Alguns exemplos são:

- Avaliação do método proposto em outras coleções de dados que são frequentemente utilizadas por outros métodos citados no Capítulo 2, como MEDLINE[1] e BDBComp[2].
- Inclusão de funções de similaridade entre referências não textuais no sistema de programação genética, como por exemplo, a função apresentada em [52], chamada de Distância *quasi-clique*, que determina a similaridade entre duas referências baseadas no grafo de coautorias das referências em relação ao resto da coleção.
- Realização de experimentos considerando usuários não ideais, ou seja, que podem errar ao rotular referências na etapa de realimentação do método.
- Adaptação do método para a utilização de mais de uma função de similaridade encontrada pelo sistema de programação genética. As funções podem ser combinadas através de uma etapa de votação, como proposto por Ferreira et al. [29] para o problema de recuperação de imagem por conteúdo.
- Comparação da etapa de classificação do método com algoritmos para problemas de classificação de conjunto aberto (*open-set classifiers*), que são classificadores utilizados quando o conjunto de treinamento não representa todas as classes possíveis, e a etapa de classificação precisa prever que novas classes podem surgir [12].
- Construção de interface para receber a realimentação do usuário, e com isso, possibilitar a realização de experimentos e avaliação da abordagem proposta com usuários reais.
- Avaliação de métricas para analisar a qualidade dos resultados obtidos, como por exemplo, utilizar o coeficiente de silhueta dos *clusters* gerados [61].

[1] `http://www.nlm.nih.gov/bsd/pmresources.html` (Acessado em Jul. de 2013).
[2] `http://www.lbd.dcc.ufmg.br/bdbcomp/` (Acessado em Jul. de 2013).

Referências Bibliográficas

[1] F. S. P. Andrade, J. Almeida, H. Pedrini, and R. da S. Torres. Fusion of local and global descriptors for content-based image and video retrieval. In *Iberoamerican Congress on Pattern Recognition*, pages 845–853, 2012.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Publishing Company, 2nd edition, 2008.

[3] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. Genetic programming: An introduction: On the automatic evolution of computer programs and its applications. *The Morgan Kaufmann Series in Artificial Intelligence*, 1997.

[4] I. Bhattacharya and L. Getoor. A latent dirichlet model for unsupervised entity resolution. In *Proceedings of the Sixth SIAM International Conference on Data Mining*, volume 124, pages 47–58, Bethesda, MD, USA, 2006.

[5] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data*, 1(1):5:1–36, 2007.

[6] A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *The Journal of Machine Learning Research*, 6:1579–1619, 2005.

[7] R. Calumby, R. da S. Torres, and M. A. Gonçalves. Multimodal retrieval with relevance feedback based on genetic programming. *Multimedia Tools and Applications*, pages 1–29, 2012.

[8] A. P. Carvalho, A. A. Ferreira, R. Silva, A. H. F. Laender, M. A. Gonçalves, and A. Veloso. Incremental unsupervised name disambiguation in cleaned digital libraries. *Journal of Information and Data Management*, 2(3):289–304, 2011.

[9] W. W. Cohen, P. D. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI-03 Workshop on Information Integration on the Web*, pages 73–78, Acapulco, Mexico, 2003.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 2001.

[11] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[12] F. O. Costa, M. Eckmann, W. J. Scheirer, and A. Rocha. Open set source camera attribution. In *2012 25th SIBGRAPI Conference on Graphics, Patterns and Images*, pages 71–78. IEEE, 2012.

[13] R. G. Cota, A. A. Ferreira, M. A. Gonçalves, A. H. F. Laender, and C. Nascimento. An unsupervised heuristic-based hierarchical method for name disambiguation in bibliographic citations. *Journal of the American Society for Information Science and Technology*, 61(9):1853–1870, 2010.

[14] A. Culotta, P. Kanani, R. Hall, M. Wick, and A. McCallum. Author disambiguation using error-driven machine learning with a ranking loss function. In *Proceedings of the International Workshop on Information Integration on the Web*, pages 32–37, Vancouver, Canada, 2007.

[15] A. T. da Silva, J. A. dos Santos, A. X. Falcão, R. da S. Torres, and L. P. Magalhães. Incorporating multiple distance spaces in optimum-path forest classification to improve feedback-based learning. *Computer Vision and Image Understanding*, 116(4):510–523, 2012.

[16] A. T. da Silva, A. X. Falcão, and L. P. Magalhães. A new cbir approach based on relevance feedback and optimum-path forest classification. *Journal of WSCG*, 18(1-3):73–80, 2010.

[17] H. M. de Almeida, M. A. Gonçalves, M. Cristo, and P. P. Calado. A combined component approach for finding collection-adapted ranking functions based on genetic programming. In *ACM SIGIR*, pages 399–406, 2007.

[18] F. F. dos Santos, R. da S. da Silva, A. T.and Torres, A. X. Falcão, L. P. Magalhães, and R. A. C. Lamparelli. Interactive classification of remote sensing images by using optimum-path forest and genetic programming. In *14th International Conference on Computer Analysis of Images and Patterns (CAIP)*, pages 300–307, 2011.

[19] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231, Portland, Oregon, 1996.

[20] A. X. Falcão, J. Stolfi, and R. de A. Lotufo. The image foresting transform: Theory, algorithms, and applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(1):19–29, 2004.

[21] W. Fan, P. Pathak, and M. Zhou. Genetic-based approaches in ranking function discovery and optimization in information retrieval - a framework. *Decision Support Systems*, 47(4):398–407, 2009.

[22] X. Fan, J. Wang, X. Pu, L. Zhou, and B. Lv. On graph-based name disambiguation. *ACM Journal of Data and Information Quality*, 2(2):10:1–23, February 2011.

[23] F. A. Faria, A. Veloso, H. M. Almeida, E. Valle, R. da S. Torres, M. A. Gonçalves, and W. Meira Jr. Learning to rank for content-based image retrieval. In *ACM MIR*, pages 285–294, 2010.

[24] A. A. Ferreira, M. A. Gonçalves, and A. H. F. Laender. Contributions for solving the author name ambiguity problem in bibliographic citations. Master's thesis, Federal University of Minas Gerais, Junho 2012.

[25] A. A. Ferreira, T. M. Machado, and M. A. Gonçalves. Improving author name disambiguation with user relevance feedback. *Journal of Information and Data Management*, 3(3):332–347, 2012.

[26] A. A. Ferreira, R. Silva, M. A. Gonçalves, A. Veloso, and A. H. F. Laender. A brief survey of automatic methods for author name disambiguation. *SIGMOD Record*, 41(2):15–26, 2012.

[27] A.A. Ferreira, R. Silva, M.A. Gonçalves, A. Veloso, and A.H.F. Laender. Active associative sampling for author name disambiguation. In *Proceedings of the 12th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 175–184. ACM, 2012.

[28] A.A. Ferreira, A. Veloso, M.A. Gonçalves, and A.H.F. Laender. Effective self-training author name disambiguation in scholarly digital libraries. In *Proceedings of the 10th Annual Joint Conference on Digital libraries*, pages 39–48. ACM, 2010.

[29] C. D. Ferreira. Recuperação de imagens com realimentação de relevância baseada em programação genética. Master's thesis, Institute of Computing-Unicamp. Supervisor: Ricardo Torres, 2007.

[30] C. D. Ferreira, J. A. Santos, R. da S. Torres, M. A. Gonçalves, R. C. Rezende, and W. Fan. Relevance feedback based on genetic programming for image retrieval. *Pattern Recognition Letters*, 32(1):27–37, 2011.

[31] E. A. Fox and G. Marchionini. Toward a worldwide digital library. *Communications of the ACM*, 41(4):29–32, 1998.

[32] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.

[33] T. A. Godoi, R. da S. Torres, A. M. B. R. Carvalho, M. A. Gonçalves, A. A. Ferreira, W. Fan, and E. A. Fox. A relevance feedback approach for the author name disambiguation problem. In *Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 209–218. ACM, 2013.

[34] D. Halliday, R. Resnick, and J. Walker. *Fundamentals of Physics*. John Wiley, 1988.

[35] J. M. Hammersley and P. Clifford. Markov fields on finite graphs and lattices. 1971. *Unpublished manuscript, cited in [Ish81]*, 1971.

[36] H. Han, C. L. Giles, H. Zha, C. Li, and K. Tsioutsiouliklis. Two supervised learning approaches for name disambiguation in author citations. In *Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 296–305, Tuscon, USA, 2004.

[37] H. Han, W. Xu, H. Zha, and C. L. Giles. A hierarchical naive Bayes mixture model for name disambiguation in author citations. In *Proceedings of the 2005 ACM Symposium on Applied Computing*, pages 1065–1069, Santa Fe, New Mexico, USA, 2005.

[38] H. Han, H. Zha, and C. L. Giles. Name disambiguation in author citations using a k-way spectral clustering method. In *Proceedings of the 5th ACM/IEEE Joint Conference on Digital Libraries*, pages 334–343, Denver, CO, USA, 2005.

[39] J. Huang, S. Ertekin, and C. L. Giles. Efficient name disambiguation for large-scale databases. In *Proceedings of the European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 536–544, Berlin, Germany, 2006.

[40] I. Kang, P. Kim, S. Lee, H. Jung, and B. You. Construction of a large-scale test set for author disambiguation. *Information Processing and Management*, 47(3):452–465, May 2011.

[41] I. Kang, S. Na, S. Lee, H. Jung, P. Kim, W. Sung, and J. Lee. On co-authorship for author disambiguation. *Information Processing & Management*, 45(1):84–97, 2009.

[42] R. E. Kass and A. E. Raftery. Bayes factors. *Journal of the American Statistical Association*, 90(430):733–795, 1995.

[43] A. H. F. Laender, M. A. Gonçalves, R. G. Cota, A. A. Ferreira, R. L. T. Santos, and A. J. C. Silva. Keeping a digital library clean: new solutions to old problems. In *Proceedings of the ACM Symposium on Document Engineering*, pages 257–262, 2008.

[44] I. Lapidot. Self-Organizing-Maps with BIC for Speaker Clustering. Technical report, IDIAP Research Institute, Martigny, Switzerland, 2002.

[45] D. Lee, J. Kang, P. Mitra, C. L. Giles, and B. On. Are your citations clean? *Communications of the ACM*, 50(12):33–38, 2007.

[46] B. M Leiner. The scope of the digital library. *DLib Working Group on Digital Library Metrics*, 1998.

[47] V. I Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet Physics Doklady*, volume 10, pages 707–710, 1966.

[48] F. H. Levin and C. A. Heuser. Evaluating the use of social networks in author name disambiguation in digital libraries. *Journal of Information and Data Management*, 1(2):183–197, 2010.

[49] M. Levin, S. Krawzyk, S. Bethard, and D. Jurafsky. Citation-based bootstrapping for large-scale author disambiguation. *Journal of the American Society for Information Science and Technology*, 63(5):1030–1047, 2012.

[50] Y. Li, A. Wen, Q. Lin, R. Li, and Z. Lu. Incorporating user feedback into name disambiguation of scientific cooperation network. In *Proceedings of the 12th International Conference on Web-age Information Management*, WAIM'11, pages 454–466, 2011.

[51] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. The MIT Press, 2012.

[52] B. On, E. Elmacioglu, D. Lee, J. Kang, and J. Pei. Improving grouped-entity resolution using quasi-cliques. In *Proceedings of the 6th IEEE International Conference on Data Mining*, pages 1008–015, 2006.

[53] B. On and D. Lee. Scalable name disambiguation using multi-level graph partition. In *Proceedings of the 7th SIAM International Conference on Data Mining*, pages 575–580, Minneapolis, Minnesota, USA, 2007.

[54] B. On, D. Lee, J. Kang, and P. Mitra. Comparative study of name disambiguation problem using a scalable blocking-based framework. In *Proceedings of the 5th*

ACM/IEEE Joint Conference on Digital Libraries, pages 344–353, Denver, CO, USA, 2005.

[55] J. P. Papa, A. X. Falcão, V. H. C. Albuquerque, and J. M. R. S. Tavares. Efficient supervised optimum-path forest classification for large datasets. *Pattern Recognition*, 45(1):512–520, 2012.

[56] J. P. Papa, A. X. Falcão, and C. T. N. Suzuki. Supervised pattern classification based on optimum-path forest. *International Journal of Imaging Systems and Technology*, 19(2):120–131, 2009.

[57] D. A. Pereira, B. A. Ribeiro-Neto, N. Ziviani, A. H. F. Laender, M. A. Gonçalves, and A. A. Ferreira. Using web information for author name disambiguation. In *Proceedings of the 2009 ACM/IEEE Joint Conference on Digital Libraries*, pages 49–58, 2009.

[58] E. Rahm and H. H. Do. Data cleaning: Problems and current approaches. *IEEE Data Engineering Bulletin*, 23(4):3–13, 2000.

[59] C. J. Van Rijsbergen. *Information Retrieval, 2nd edition*. Butterworths, London, 1979.

[60] L. M. Rocha, F. A. M. Cappabianco, and A. X. Falcão. Data clustering as an optimum-path forest problem with applications in image analysis. *International Journal of Imaging Systems and Technology*, 19(2):50–68, 2009.

[61] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(1):53–65, 1987.

[62] J. A. Santos and R. da S. Torres. Reconhecimento semi-automático e vetorização de regiões em imagens de sensoriamento remoto. Master's thesis, Institute of Computing-Unicamp. Supervisor: Ricardo Torres, 2009.

[63] J. M. Soler. Separating the articles of authors with the same name. *Scientometrics*, 72(2):281–290, 2007.

[64] Y. Song, J. Huang, I. G. Councill, J. Li, and C. L. Giles. Efficient topic-based unsupervised name disambiguation. In *Proceedings of the 7th ACM/IEEE Joint Conference on Digital Libraries*, pages 342–351, Vancouver, BC, Canada, 2007.

[65] P. Tan et al. *Introduction to data mining*. Pearson Education India, 2007.

[66] J. Tang, A. C. M. Fong, B. Wang, and J. Zhang. A unified probabilistic framework for name disambiguation in digital library. *IEEE Transactions on Knowledge and Data Engineering*, 24(6):975–987, 2012.

[67] R. da S. Torres, A. X. Falcão, M. A. Gonçalves, J. P. Papa, B. Zhang, W. Fan, and E. A. Fox. A genetic programming framework for content-based image retrieval. *Pattern Recognition*, 42(2):283–292, 2009.

[68] V. I. Torvik and N. R. Smalheiser. Author name disambiguation in medline. *ACM Transactions on Knowledge Discovery from Data*, 3(3):1–29, 2009.

[69] P. Treeratpituk and C. L. Giles. Disambiguating authors in academic publications using random forests. In *Proceedings of the 2009 ACM/IEEE Joint Conference on Digital Libraries*, pages 39–48, Austin, TX, USA, 2009.

[70] A. Veloso, A. A. Ferreira, M. A. Gonçalves, A. H.F. Laender, and W. Meira Jr. Cost-effective on-demand associative author name disambiguation. *Information Processing & Management*, 48(4):680 – 697, 2012.

[71] X. Wang, J. Tang, H. Cheng, and P.S. Yu. Adana: Active name disambiguation. In *Proceedings of the 11th International Conference on Data Mining*, pages 794–803, Vancouver, Canada, 2011.

[72] W. E. Winkler. The state of record linkage and current research problems. In *Statistical Research Division, US Census Bureau*. Citeseer, 1999.