UMA ABORDAGEM BASEADA NA WEB PARA RESOLUÇÃO DE ENTIDADES E CRIAÇÃO DE ARQUIVOS DE AUTORIDADE

DENILSON ALVES PEREIRA

UMA ABORDAGEM BASEADA NA WEB PARA RESOLUÇÃO DE ENTIDADES E CRIAÇÃO DE ARQUIVOS DE AUTORIDADE

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

Orientador: Berthier Ribeiro de Araújo Neto
Co-Orientador: Nivio Ziviani

Belo Horizonte - MG

04 de dezembro de 2009

DENILSON ALVES PEREIRA

A WEB-BASED APPROACH FOR ENTITY RESOLUTION AND CREATION OF AUTHORITY FILES

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Advisor: Berthier Ribeiro de Araújo Neto
Co-Advisor: Nivio Ziviani

Belo Horizonte - MG

December 4, 2009

Alves Pereira, Denilson
D1234p A Web-based Approach for Entity Resolution and Creation of Authority Files / Denilson Alves Pereira. — Belo Horizonte - MG, 2009
xxxviii, 94 f. : il. ; 29cm

Tese (doutorado) — Federal University of Minas Gerais
Orientador: Berthier Ribeiro de Araújo Neto

Co-Orientador: Nivio Ziviani

1. Entity Resolution. 2. Authority File. 3. Web Search Engine. 4. Name Disambiguation.
5. Clustering. I. Título.

CDU 519.6*82.10

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

FOLHA DE APROVAÇÃO

Uma abordagem baseada na Web para resolução de entidades e criação de arquivos de autoridade

DENILSON ALVES PEREIRA

Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. BERTHIER RIBEIRO DE ARAÚJO NETO - Orientador
Departamento de Ciência da Computação - UFMG

PROF. NIVIO ZIVIANI - Co-orientador
Departamento de Ciência da Computação - UFMG

PROF. CARLOS ALBERTO HEUSER
Departamento de Informática - UFRGS

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

PROF. MARCO ANTÔNIO CASANOVA
Departamento de Informática - PUC - RJ

PROF. ALBERTO HENRIQUE FRADE LAENDER
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 04 de dezembro de 2009.

To my wife Fabiana and my son João Vítor.
To my parents Duarte and Tereza.
To my aunt Conceição.

Agradecimentos

Em especial, agradeço do fundo do meu coração a minha esposa Fabiana e ao meu filho João Vítor. À Fabiana, por todo amor, incentivo e apoio irrestrito durante todos esses anos. Ao João Vítor, que mesmo sem entender o porquê da minha ausência, adorava os poucos momentos que passávamos juntos.

A Deus por me dar força e sabedoria para chegar ao final.

Aos meus pais e irmãos pelo carinho e incentivo – em especial a minha mãe, que com sua visão foi a principal responsável por eu ter chegado até aqui.

Ao meu orientador, Prof. Berthier Ribeiro-Neto, por tudo que me ensinou; ao meu co-orientador, Prof. Nivio Ziviani, pelo seu profissionalismo e dedicação; ao Prof. Alberto Laender, por suas detalhadas revisões que muito contribuíram para este trabalho; ao Prof. Marcos Gonçalves, por suas valorosas sugestões; aos demais membros da banca, Profs. Carlos Heuser, Edleno Silva e Marco Antonio Casanova, pelas suas avaliações e contribuições e aos funcionários do DCC, sempre muito competentes.

Aos colegas e amigos do Latin: Alan, Álvaro, Anísio, Claudine, Daniel, David, Fabiano, Guilherme, Hendrickson, Humberto, Marco Cristo, Marco Modesto, Rickson, Thierson, Tupy, Vinícius, Wallace e Wladmir. Ao amigo Anderson Almeida pelas valorosas discussões técnicas. Aos colegas e amigos com quem convivi em algum momento no DCC: Borghetti, Evandrino, Guilherme Tavares, Maurício, Max, Pio e Ruiter. Aos colegas e amigos do UNI-BH. And to my English Teacher, Rosely Veiga, por suas revisões e ensinamentos na língua inglesa.

E, finalmente, à FAPEMIG e ao CNPq, pelo incentivo financeiro.

Resumo

Repositórios de dados na Web normalmente contêm referências para milhares de entidades do mundo real. Não é incomum que múltiplas entidades compartilhem um mesmo rótulo (homônimos) e que variações distintas de rótulos sejam associadas a uma mesma entidade (sinônimos), o que frequentemente conduz a interpretações ambíguas. Variações e erros de grafias, siglas e formas abreviadas contribuem para tornar o problema ainda mais difícil. Resolver este problema requer identificar quais rótulos correspondem a uma mesma entidade do mundo real, um processo conhecido como resolução de entidades. Uma abordagem para resolver o problema é selecionar um identificador de autoridade para cada entidade, bem como, uma lista de formas variantes usadas para referenciá-la — uma estrutura de dados conhecida como arquivo de autoridade.

O objetivo desta tese é propor um novo método de gerar arquivos de autoridade baseado em informações disponíveis na Web. O método consiste em coletar informações sobre as referências a entidades, submetê-las como consultas a uma máquina de busca Web, analisar o conjunto resposta e extrair informações para desambiguar as referências. Um arcabouço genérico e configurável, chamado WER (do inglês, Web-based Entity Resolution), foi implementado e validado. Experimentos com três bases de dados distintas mostram que esse método supera os métodos de referência selecionados para comparação, alcançando ganhos na métrica pairwise F1 de até 125%. Uma especialização do arcabouço para resolução de nomes de autores em citações bibliográficas também é apresentada. Tal solução usa heurísticas para identificar documentos contendo citações de um único autor e um procedimento de agrupamento que coloca em um mesmo grupo citações encontradas em um mesmo documento. É feita uma comparação dessa solução específica com a solução genérica.

Palavras-chave: Resolução de Entidades, Arquivos de Autoridade, Máquina de Busca Web, Desambiguação de Nomes, Agrupamento.

Abstract

Web data repositories usually contain references to thousands of real-world entities, originated from multiple sources. It is not uncommon that multiple entities share a same label (polysemes) and that distinct label variations are associated with the same entity (synonyms), which frequently leads to ambiguous interpretations. Further, spelling variants, acronyms, abbreviated forms, and misspellings compound to make the problem worst. Solving this problem requires identifying which labels correspond to a same real-world entity, a process that is known as entity resolution. One approach to solve the entity resolution problem is to select an authority identifier for each entity, as well as a list of variant forms—a data structure known as an authority file.

The objective of this thesis is to propose a new method of generating authority files based on information available in the Web. The method consists of gathering information on entity references, submitting them as queries to a Web search engine, parsing the answer set, and extracting information to disambiguate references. We present the design, implementation, and validation of a generic and configurable framework for entity resolution and creation of authority files, called WER—Web-based Entity Resolution. Experiments on three distinct datasets suggest that the method far outperforms selected baselines, achieving gains in the pairwise F1 metric up to 125%. We also present a specialization of the framework for author name resolution in bibliographic citations. It uses heuristics to identify documents containing citations of a single author and a clustering procedure that groups citations in a same document together. We compare such specific solution with the generic one.

Resumo Estendido

Introdução

A tarefa de identificar diferentes referências a uma mesma entidade do mundo real em grandes repositórios de dados é árdua. Por exemplo, espera-se que serviços de catálogo de produtos disponíveis na Web gerem informação relacionada aos produtos de interesse do usuário. Entretanto, dado que rótulos similares podem ser usados para referenciar produtos distintos em páginas diferentes da Web, produtos não relacionados à intenção do usuário podem aparecer no conjunto resposta. Para ilustrar, enquanto "Impressora HP Officejet J3680 tudo-em-uma Fax, Scanner, Copiadora", "Impressora Multifuncional HP CB071A#A2L J3680 Officejet" e "Impressora Multifuncional Hewlett Packard Officejet 3680 (CB071A)" se referem a uma mesma impressora, "Impressora HP Officejet J4580 tudo-em-uma Fax, Scanner, Copiadora" corresponde a uma impressora diferente.

Bibliotecas digitais necessitam manter metadados de citações bibliográficas coletados de várias fontes e enfrentam um problema similar. Além de dados replicados, é comum encontrarmos nomes de autores ambíguos em citações bibliográficas. A ambiguidade pode ocorrer devido à existência de múltiplos autores com um mesmo nome (homônimos) ou diferentes variações de nomes para um mesmo autor (sinônimos).

Dadas várias referências a entidades, tais como rótulos de produtos ou citações bibliográficas, o processo de identificar quais delas correspondem a uma mesma entidade do mundo real é conhecido como resolução de entidades. Um mesmo registro, composto de um ou mais atributos, pode conter referências a múltiplas distintas entidades. Por exemplo, uma citação bibliográfica pode conter atributos tais como nomes dos autores, título do trabalho e nome do veículo de publicação, cada atributo correspondendo a uma entidade distinta. Isto é, nós podemos querer encontrar os registros que se referem a um mesmo veículo de publicação ou que se referem a um mesmo autor ou que se referem a um mesmo artigo (útil para encontrar citações replicadas) — três entidades distintas. Nos dois primeiros casos, a entidade a ser desambiguada consiste de um atributo do

registro, e no último caso, ela consiste de todo o registro. Assim, está claro que a entidade a ser desambiguada deve ser fornecida como entrada. Nós frequentemente nos referimos ao registro composto de atributos como referência a entidade. Entretanto, pode ser que os atributos sirvam com referência a múltiplas entidades. Nesse caso, é necessário deixar explícito qual atributo será considerado como a referência a entidade a ser desambiguada.

Nos casos de identificar réplicas, alguns trabalhos também tratam o problema de juntar as referências replicadas gerando uma forma canônica para cada entidade. Nos casos em que a entidade a ser desambiguada consiste de somente um dos atributos das referências a entidades, alguns trabalhos também obtêm um nome de autoridade (nome canônico) para cada entidade e mantêm uma lista de formas variantes usadas para referenciar a ela, gerando um arquivo de autoridade. A abordagem deste trabalho é gerar arquivos de autoridade, nós não temos como objetivo juntar referências a entidades duplicadas.

Nesta tese, nós apresentamos uma abordagem baseada na Web para o problema de resolução de entidades, a qual nós nos referimos como WER (do inglês *Web-based Entity Resolution*). Nós construímos um arcabouço genérico e configurável para tratar com várias instâncias do problema de resolução de entidades, o qual permite o desenvolvimento de soluções fáceis e convenientes para novos domínios. Nós discutimos uma solução específica para resolução de nomes de autores, codificada usando nosso arcabouço.

Dado um conjunto de referências a entidades de um domínio específico, nossa solução WER é composta de quatro passos: (1) de cada referência a uma entidade, gere uma consulta, submeta-a a uma máquina de busca Web e colete os m documentos do topo do conjunto resposta, (2) do conjunto de documentos coletados para cada consulta, extraia alguns atributos tais como URLs, títulos, textos dos documentos, nomes e siglas usadas para referir à entidade, (3) aplique um procedimento de agrupamento para agrupar as referências a entidades, tal que cada grupo represente um entidade distinta do mundo real, e (4) gere um arquivo de autoridade selecionando um nome em cada grupo para ser o nome canônico da entidade. Nossa abordagem é singular, porque combina os atributos originais com os atributos extraídos da Web para melhorar o procedimento de agrupamento.

As principais contribuições desta tese são: (1) um novo método que usa informação disponível na Web como uma fonte adicional de evidência para resolução de entidades e criação de arquivos de autoridade. Esse método, o qual inclui heurísticas específicas para extrair informação pertinente de documentos Web, conduz a resultados melhores quando comparados a soluções estado da arte para o problema; (2) um arca-

bouço configurável que permite modelar convenientemente diferentes estratégias para resolução de entidades e avaliar seus resultados; (3) uma validação empírica da nossa solução para o problema de resolução de entidades com experimentos extensivos; (4) uma implementação personalizada de nossa solução para desambiguar nomes de autores (em citações bibliográficas), bem como sua comparação com uma implementação variante (da mesma solução) sobre nosso arcabouço configurável.

A Abordagem WER

No problema de resolução de entidades, nós recebemos como entrada um conjunto de registros $\mathcal{R} = \{r_1, r_2, ..., r_n\}$, onde cada registro r_i tem atributos $r_i.A_1, r_i.A_2, ..., r_i.A_k$, os quais nós chamamos de atributos específicos do domínio. A referência para a entidade a ser desambiguada, a qual nós nos referimos como entidade primária, pe, também é fornecida como entrada. O problema de resolução de entidades consiste em (i) determinar o conjunto $\mathcal{E} = \{e_1, e_2, ..., e_m\}$ de entidades distintas do mundo real relacionadas à entidade primária pe e (ii) associar a cada registro r_i a correspondente (correta) entidade $e_j \in \mathcal{E}$. O número m de entidades distintas não é fornecido como entrada.

Um arquivo de autoridade é um índice de registros de autoridade, onde cada registro representa uma entidade, mantém um cabeçalho para ser o rótulo de autoridade e uma lista de variações de rótulos usados para se referir à entidade, chamados de referências cruzadas. Os cabeçalhos e as referências cruzadas são usadas por um sistema de busca quando um usuário pesquisa por dados relacionados a uma entidade. Por exemplo, em um arquivo de autoridade de nomes de autores, cada registro representa um autor, seu nome completo poderia ser escolhido para ser o cabeçalho enquanto os outros nomes poderiam ser mantidos como formas alternativas usadas para se referir àquele autor. Um usuário pode pesquisar por publicações de um autor usando qualquer um de seus nomes. Dado um conjunto de registros $\mathcal{R}$ e uma entidade primária pe, um arquivo de autoridade pode ser gerado depois de se resolver o problema de resolução de entidades. Desde que cada grupo representa uma entidade, é suficiente selecionar, para cada um, um cabeçalho relacionado a pe para ser o rótulo de autoridade da entidade.

A abordagem WER usa a Web com uma fonte de informação adicional para resolução de entidades e criação de arquivos de autoridade. Segue abaixo um resumo dos principais passos do nosso método:

- Passo 1 (Coletando informação da Web): Submeta consultas a uma máquina de busca Web e colete os m documentos do topo dos conjuntos respostas. Cada consulta é formada pelos valores de um subconjunto de atributos específicos do

domínio de cada registro r_i na entrada. O subconjunto de atributos usados para compor a consulta é selecionado pelo usuário.

- Passo 2 (Extraindo informação dos documentos): Extraia atributos inferidos da Web para cada registro r_i analisando os documentos no conjunto resposta associado a sua consulta. Os atributos inferidos da Web são, por exemplo, URLs, títulos, textos dos documentos, nomes e siglas. Em princípio, todos podem ser usados para referir à entidade.

- Passo 3 (Agrupando registros): Agrupe os registros usando uma combinação dos atributos originais específicos do domínio e dos inferidos da Web.

- Passo 4 (Gerando um arquivo de autoridade): Gere um arquivo de autoridade selecionando um nome em cada grupo para ser o nome canônico da entidade.

Extraindo Informação dos Documentos

Os documentos retornados por cada consulta associada ao registro r_i são usados para se extrair novos atributos para r_i (Passo 2), os quais nós nos referimos como atributos *inferidos da Web*. Os detalhes desse passo são apresentados a seguir.

Extraindo Atributos de cada Documento

Nesta tese, nós propomos um método para extrair os seguintes cinco atributos: URL, título e texto de um documento, cabeçalho e sigla para a entidade primária. Formalmente, seja $\mathcal{D}_i$ o conjunto dos m documentos no topo do conjunto resposta da consulta associada ao registro r_i. O conjunto de todos os documentos coletados forma a subcoleção de documentos Web $\mathcal{D}$. De cada documento $d_j \in \mathcal{D}$, nós extraímos valores para os seguintes atributos: URL ($d_j.B_{url}$), título ($d_j.B_{titulo}$), texto ($d_j.B_{texto}$), cabeçalho ($d_j.B_{cabec}$) e sigla ($d_j.B_{sigla}$). A extração dos três primeiros atributos é direta. $d_j.B_{url}$ é a URL associada com o documento, $d_j.B_{titulo}$ é o título do documento, isto é, o texto entre os elementos HTML <title> e </title>, e $d_j.B_{texto}$ é o conteúdo do texto do documento.

Para extrair o cabeçalho associado com o documento $d_j \in \mathcal{D}_i$, nós primeiro quebramos o texto do documento em frases, onde cada frase é formada por uma sequência de palavras delimitadas por marcas de pontuação. Depois, nós extraímos a frase mais similar ao valor do atributo de r_i corresponde à entidade primária. A similaridade é baseada em uma função tal como o coeficiente de similaridade de Jaccard. Abreviações

simples são expandidas, como aquelas que casam com o início de uma palavra. Se a expansão for bem sucedida, a forma expandida é considerada como uma candidata para o cabeçalho.

Quando o valor do atributo do cabeçalho é um texto curto (uma ou duas palavras) e a palavra é identificada como uma sigla, então essa sigla é expandida contra as frases extraídas de d_j (isto é, para cada letra na sigla tenta-se associar uma palavra). A frase com o mais alto coeficiente de expansão, mais alto que um dado limiar, é escolhido como o cabeçalho. Para siglas de entidades bem conhecidas, a máquina de busca geralmente retorna a sigla e o nome longo da entidade nos documentos, porque as duas formas estão frequentemente juntas.

Do texto de d_j, nós extraímos uma sigla (se ela existir) que casa com o cabeçalho extraído. Candidatas a sigla devem ter pelo menos duas letras maiúsculas. O método de expansão tenta casar letras iniciais, abreviações e conversões comuns, como "2" e "to" em inglês. Um coeficiente de expansão é computado como a taxa de símbolos da sigla que foram expandidos. A sigla com o mais alto coeficiente de expansão, mais alto que um dado limiar, é extraída para representar a sigla associada com d_j.

Compondo os Atributos Inferidos da Web

Usando os atributos extraídos dos documentos $\mathcal{D}_i$ retornados pela consulta associada ao registro r_i, nós definimos valores para os seguintes atributos inferidos da Web para r_i: conjunto de URLs dos documentos ($r_i.B_{url}$), conjunto de títulos dos documentos ($r_i.B_{titulo}$), conjunto de textos dos documentos ($r_i.B_{texto}$), cabeçalho de r_i ($r_i.B_{cabec}$), e sigla de r_i, se ela existir ($r_i.B_{sigla}$). Esses atributos são definidos como explicado a seguir.

Os valores para os atributos conjuntos de URLs, títulos e textos para o registro r_i são obtidos dos documentos $d_j \in \mathcal{D}_i$, respectivamente, como:

$$r_i.B_{url} = \bigcup_j d_j.B_{url}$$

$$r_i.B_{titulo} = \bigcup_j d_j.B_{titulo}$$

$$r_i.B_{texto} = \bigcup_j d_j.B_{texto}$$

Nós selecionamos o cabeçalho com a mais alta soma de similaridades em relação aos outros cabeçalhos extraídos de $\mathcal{D}_i$ como sendo o atributo cabeçalho para o registro

r_i. Seja $simH(d_j.B_{cabec}, d_k.B_{cabec})$ a função que retorna a similaridade entre as cadeias de caracteres $d_j.B_{cabec}$ e $d_k.B_{cabec}$. Então, nós computamos a soma de similaridades entre as cadeias de caracteres $d_j.B_{cabec}$ e $d_k.B_{cabec}$, $\forall d_j, d_k \in \mathcal{D}_i$, como segue:

$$sumSim(d_j.B_{cabec}) = \sum_{k \neq j} simH(d_j.B_{cabec}, d_k.B_{cabec}) \tag{1}$$

E então, o valor para o atributo cabeçalho para o registro r_i é computado como:

$$r_i.B_{cabec} = d_j.B_{cabec}, \text{ tal que } d_j.B_{cabec} \text{ tem a máxima } sumSim(d_j.B_{cabec}) \tag{2}$$

Nós selecionamos a sigla mais frequente obtida dos documentos $d_j \in \mathcal{D}_i$ como o valor do atributo sigla para o registro r_i. Seja $aCount(d_j.B_{sigla})$ a função que conta o número de ocorrências de cada sigla distinta e não nula $d_j.B_{sigla}$. Então, o valor para o atributo sigla para o registro r_i é computado como:

$$r_i.B_{sigla} = d_j.B_{sigla}, \text{ tal que } d_j.B_{sigla} \text{ tem a máxima } aCount(d_j.B_{sigla}) \tag{3}$$

Agrupando Registros

Para agrupar dados (Passo 3), nós usamos os valores dos atributos específicos do domínio e atributos inferidos da Web. Cada grupo é esperado representar uma entidade distinta do mundo real. O procedimento de agrupamento computa a similaridade para-a-par dos registros r_i e r_j usando a função de similaridade, $sim(r_i, r_j)$, baseada em uma combinação linear de funções de similaridade que comparam os valores dos atributos associados com cada registro. Seja $F_p(r_i, r_j)$ a função de similaridade para o p^{esimo} atributo dos registros r_i and r_j, e seja w_p um peso associado com o p^{esimo} atributo. Então, nós definimos

$$\begin{cases} sim(r_i, r_j) = \sum_p w_p \times F_p(r_i, r_j) \\ \sum_p w_p = 1 \end{cases} \tag{4}$$

As funções de similaridade para cada um dos atributos inferidos da Web são computadas como a seguir.

$$F_{url}(r_i, r_j) = \begin{cases} 1.0 & \text{se } |r_i.B_{url} \cap r_j.B_{url}| \geq q, q > 0 \\ \frac{p}{q} & \text{se } |r_i.B_{url} \cap r_j.B_{url}| = p, 0 \leq p < q \end{cases}$$

onde q é um parâmetro definido empiricamente, correspondendo ao número requerido de URLs em comum nos dois conjuntos.

$$F_{titulo}(r_i, r_j) = simT(r_i.B_{titulo}, r_j.B_{titulo})$$

Analogamente à função $F_{titulo}(r_i, r_j)$, nós definimos as funções $F_{texto}(r_i, r_j)$ e $F_{cabec}(r_i, r_j)$ usando as funções de similaridade $simX(r_i.B_{texto}, r_j.B_{texto})$ e $simH(r_i.B_{cabec}, r_j.B_{cabec})$, respectivamente, onde $simT$, $simX$ e $simH$ são quaisquer funções de similaridade de cadeias de caracteres, tais como coeficiente de Jaccard ou similaridade do cosseno.

Finalmente,

$$F_{sigla}(r_i, r_j) = \begin{cases} 1.0 & \text{se } r_i.B_{sigla} = r_j.B_{sigla} \text{ e } r_i.B_{sigla} \text{ não é nulo} \\ 0.0 & \text{se } r_i.B_{sigla} \neq r_j.B_{sigla} \\ v & \text{se } r_i.B_{sigla} \text{ é nulo e } r_j.B_{sigla} \text{ é nulo}, 0.0 \leq v \leq 1.0 \end{cases}$$

onde v é um parâmetro definido empiricamente, usado quando não há nenhuma sigla em comum.

Para o agrupamento de registros, nós podemos usar um procedimento de agrupamento baseado em qualquer técnica, tal como K-vizinhos-mais-próximos (KNN) ou o agrupamento aglomerativo hierárquico (HAC).

Gerando um Arquivo de Autoridade

No Passo 4, nós geramos um arquivo de autoridade. Desde que cada grupo c_k gerado no Passo 3 representa uma entidade distinta do mundo real $e_k \in \mathcal{E}$ então, ele representa um registro de autoridade ar_k, e nós armazenamos uma ligação para os registros que ele agrupa. Ainda, nós armazenamos o conjunto de valores dos atributos de seus registros correspondentes à entidade primária, o qual compreende o atributo referências cruzadas $ar_k.A_{cruz}$. E ainda, nós definimos seu atributo cabeçalho $ar_k.A_{cabec}$, computado como segue.

Analogamente à Eq. (1) e à Eq. (2), nós computamos o atributo cabeçalho $ar_k.A_{cabec}$ para o registro de autoridade ar_k, onde cada $r_i.B_{cabec}$ é o cabeçalho do

registro $r_i \in c_k$, como segue:

$$sumSim(r_i.B_{cabec}) = \sum_{j \neq i} simH(r_i.B_{cabec}, r_j.B_{cabec})$$

$ar_k.A_{cabec} = r_i.B_{cabec}$, tal que $r_i.B_{cabec}$ tem a máxima $sumSim(r_i.B_{cabec})$

Se uma sigla existir para qualquer registro em c_k, nós computamos o atributo sigla $ar_k.A_{sigla}$ para o registro de autoridade ar_k e o concatenamos ao cabeçalho, dando mais informação sobre a entidade. Analogamente à Eq. (3), ele é computado como:

$ar_k.A_{sigla} = r_i.B_{sigla}$, tal que $r_i.B_{sigla}$ tem a máxima $aCount(r_i.B_{sigla})$

Configurações do Arcabouço WER

Nós implementamos um arcabouço genérico e configurável para resolução de entidades e criação de arquivos de autoridade. O arcabouço é composto de classes, correspondendo aos passos da abordagem WER. Essas classes contêm operações básicas para resolução de entidades e criação de arquivos de autoridade genéricos, e podem ser estendidas para aplicações de domínio específico.

O arcabouço WER é configurável e o usuário pode, por exemplo, definir os atributos dos registros, selecionar os atributos para compor as consultas, o atributo correspondente à entidade primária, o tipo do documento a ser coletado (texto resumido ou texto completo), as funções de similaridade a serem usadas para comparar cadeias de caracteres na extração de informação e no procedimento de agrupamento (ex: distância de edição, coeficiente de Jaccard e cosseno), o procedimento de agrupamento e os pesos a serem usados nas funções de similaridade do agrupamento.

O arcabouço WER também implementa algumas métricas para avaliar os agrupamentos, tais como a métrica K, pairwise F1 e cluster F1. Ele é útil para simulações, permitindo modelar diferentes estratégias para resolução de entidades e para medir seus resultados.

Avaliação Experimental

Para demonstrar a efetividade dos atributos inferidos da Web, nós avaliamos o método WER usando funções de similaridade aplicadas somente a eles, sem combiná-los com os atributos específicos do domínio. O objetivo dos nossos experimentos foi comparar tal estratégia contra um método de referência (*baseline*) que usa o mesmo procedimento de agrupamento e as mesmas funções de similaridade, porém, aplicadas somente a atributos específicos do domínio.

Nos experimentos mostrados, para todas as funções de similaridade que comparam atributos do tipo cadeia de caracteres, nós usamos o coeficiente de similaridade de Jaccard, para ambos, o método de referência e o WER. Nós experimentamos outras funções de similaridade tais como distância de edição, cosseno e Jaccard combinado com distância de edição. Para todas elas, as diferenças nos resultados foram similares à Jaccard. E ainda, nós ajustamos os pesos das contribuições das funções de similaridade para obter os melhores resultados, tanto para o método de referência quanto para o WER. Para o procedimento de agrupamento, os resultados relatados se referem ao KNN, desde que ele se mostrou melhor do que o HAC, que também experimentamos.

Para obter informação da Web, nós submetemos consultas usando a API Google Search e coletamos os 10 textos resumidos no topo do resultado de cada consulta.

Nós avaliamos o WER aplicado aos seguintes três problemas: resolução de descrições de impressoras, títulos de veículos de publicação e nomes de autores. Cada um dos problemas usou uma base de dados distinta. A base de dados de descrições de impressoras foi coletada a partir de consultas ao *Google's Product Search* usando descrições de impressoras obtidas de sítios de fabricantes de impressoras. Ela é composta de 2.169 cadeias de caracteres distintas de descrições de impressoras, distribuídas em 158 grupos, contendo em média $13,7$ cadeias por grupo. A base de dados para o problema de resolução de títulos de veículos de publicação contém citações bibliográficas coletadas a partir de consultas ao *Google Scholar* usando nomes de professores de quatro departamentos de ciência da computação das principais universidades americanas. Ela é composta de 16.689 registros de citações contendo 8.399 cadeias de caracteres distintas para o título do veículo de publicação. Nós avaliamos uma amostra aleatória dessa base contendo 691 cadeias de caracteres distintas, distribuídas em 110 grupos, contendo em média $6,3$ cadeias por grupo. E a base de dados para o problema de resolução de nomes de autores contém citações bibliográficas coletadas da DBLP. Ela é composta de 8.442 registros de citações, com 480 autores distintos, divididos em 14 grupos ambíguos.

A Tabela 1 apresenta os resultados comparando o WER e o método de referência

para cada um dos três problemas. A terceira linha de cada comparação mostra os ganhos do WER, os quais são estatisticamente significativos, exceto para a métrica cluster F1 (cF1) para os dois últimos problemas. Os resultados em cluster F1 são baixos porque as bases de dados contêm muitos grupos grandes, contendo cada um muitas formas variantes, o que torna difícil obter grupos totalmente corretos.

Método	Problema	K (%)	pF1 (%)	cF1 (%)
WER	impressora	76.2 ± 1.8	63.9 ± 2.0	6.3 ± 1.0
baseline	impressora	51.6 ± 2.1	28.4 ± 1.9	0.2 ± 0.2
	ganhos do WER	**47.7**	**125.0**	**3050.0**
WER	veic. publicação	75.4 ± 3.2	37.5 ± 3.6	27.6 ± 3.3
baseline	veic. publicação	67.2 ± 3.5	19.3 ± 2.9	25.1 ± 3.2
	ganhos do WER	**12.2**	**94.3**	10.0
WER	autores	72.6 ± 5.0	66.4 ± 9.3	3.8 ± 1.7
baseline	autores	55.9 ± 3.6	37.3 ± 7.7	5.1 ± 2.2
	ganhos do WER	**28.1**	**78.0**	-25.5

Tabela 1. Comparação do WER contra métodos de referência (*baselines*) para os problemas de resolução de descrições de impressoras, veículos de publicação e nomes de autores. Todos os métodos usam a função de similaridade baseada no coeficiente de Jaccard para comparar cadeias de caracteres e o procedimento de agrupamento KNN. Cada célula mostra o valor obtido pelo método para as métricas K, pairwise F1 (pF1) e cluster F1 (cF1), e seus intervalos com 95% de confiança. Para cada problema, a terceira linha mostra os ganhos do WER, cujos valores em negrito são estatisticamente significantes.

Manualmente verificando os resultados dos agrupamentos, nós pudemos confirmar que cadeias de caracteres similares tais como "Impressora HP Officejet J3680 tudo-em-uma Fax, Scanner, Copiadora" e "Impressora HP Officejet J4580 tudo-em-uma Fax, Scanner, Copiadora", descrevendo impressoras distintas, foram colocadas em um mesmo grupo pelo método de referência e em grupos distintos pelo WER. Também, cadeias de caracteres com poucas palavras em comum, tais como "Impressora HP Officejet J3680 tudo-em-uma Fax, Scanner, Copiadora" e "Impressora Multifuncional HP CB071A#A2L J3680 Officejet", descrevendo uma mesma impressora, foram colocadas em um mesmo grupo pelo WER e em grupos distintos pelo método de referência. Em ambos os casos, as URLs foram importantes para desambiguá-las. Resultados similares ocorrem nas outras duas bases de dados, sendo que no problema de títulos de veículos de publicação as siglas também desempenham um papel importante na desambiguação.

Nós analisamos casos de falhas do WER e encontramos erros para desambiguar, por exemplo, impressoras que possuem variações de um mesmo modelo, tais como "Impressora Laser HP P3005", "Impressora Laser HP P3005d" e "Impressora Laser HP P3005dn". Elas têm cadeias de caracteres similares e suas consultas retornam

algumas URLs em comum, o que faz com que o WER coloque-as em um mesmo grupo, mesmo sendo impressoras distintas. Nesse caso, o método de referência também produz resultados incorretos.

Nós também avaliamos a qualidade dos cabeçalhos dos arquivos de autoridade gerados pelo WER, comparando-os com as cadeias de caracteres originais correspondentes às descrições de impressoras, aos títulos de veículos de publicação e aos nomes de autores. Os resultados mostram que o WER obtém melhores descrições para $40,7\%$ das impressoras, com uma taxa de erro de $7,4\%$, obtém melhores títulos de veículos de publicação para $25,0\%$ dos títulos, com uma taxa de erro de $6,5\%$, e obtém um nome expandido correto para $27,4\%$ dos autores, com uma taxa de erro de $4,7\%$.

Uma Solução Específica para Resolução de Nomes de Autores

Nossa proposta de extrair informação da Web para ajudar no processo de desambiguação de entidades pode ser especializado para algumas aplicações. Nesse caso, nós podemos usar conhecimento específico sobre a aplicação para aplicar heurísticas que podem melhorar os resultados. Nós aplicamos essa ideia para desenvolver uma solução específica para resolução de nomes de autores.

Em nossos experimentos na resolução de nomes de autores, nós submetemos consultas a uma máquina de busca objetivando encontrar documentos contendo publicações dos autores. E então, nós confiamos no fato de que se duas ou mais consultas retornam documentos em comum, isso pode indicar que tais documentos contêm publicações de um mesmo autor. De fato, isso é verdade para muitos documentos. No entanto, há muitos outros que contém publicações de autores distintos, tais como textos de artigos ou páginas de artigos em bibliotecas digitais que também contém suas referências bibliográficas, ou listas de artigos de uma revista ou edição de uma conferência. Nós podemos tirar vantagem de conhecer tais propriedades específicas da aplicação e usar algumas heurísticas para determinar quais são os documentos que podem efetivamente ajudar no processo de desambiguação de nomes de autores.

Nossa solução específica é similar a nossa solução genérica no sentido que nós submetemos consultas a uma máquina de busca usando os valores dos atributos das citações de entrada, coletamos e extraímos informação dos documentos dos conjuntos respostas das consultas e aplicamos um procedimento de agrupamento para agrupar citações de um mesmo autor. A principal diferença é que nós usamos conhecimento específico sobre a aplicação para classificar os documentos em ordem de importância

para o processo de desambiguação. Nossa solução específica é baseada em: (1) identificar documentos de um único autor, isto é, documentos contendo citações de uma única pessoa, tal como um currículo, (2) ponderar os documentos, gerando uma ordem de sua importância no processo de desambiguação e (3) aplicar um procedimento de agrupamento específico usando os documentos de um único autor e o ordenamento dos documentos.

Para identificar um documento de um único autor, nós procuramos pelo nome de autor no título da página de um documento, em sua URL e no início de seu texto. No último caso, nós tentamos identificar currículos, onde o nome do autor aparece no início do documento junto com palavras tais como "curriculum vitae", "currículo", "nome", "endereço" e "fone". Se um nome de autor aparece sozinho em um desses três lugares e o mesmo nome aparece em todas as citações encontradas no documento, então o documento é considerado de um único autor.

A importância de cada documento no processo de desambiguação é quantificada usando a frequência inversa do servidor (IHF, *inverse host frequency*). A IHF quantifica a raridade relativa de um servidor na Internet, semelhante à frequência inversa do documento (IDF), usada em recuperação de informação. A ideia é que páginas de sítios Web raros têm mais importância no processo de desambiguação do que as de sítios comuns, tais como páginas de bibliotecas digitais.

Finalmente, nós usamos um procedimento de agrupamento aglomerativo hierárquico que agrupa citações encontradas em documentos de um único autor com alto IHF. Tais documentos são provavelmente currículos ou páginas pessoais contendo citações de uma única pessoa. Para os documentos de um único autor com baixo IHF, o WER agrupa somente aquelas citações que forem também encontradas juntas em outros documentos. Tais documentos são provavelmente páginas de autores em bibliotecas digitais, e nós procuramos por mais evidências para agrupar suas citações, desde que bibliotecas digitais podem conter erros.

Resultados experimentais mostram que nossa solução específica para resolução de nomes de autores produz melhores resultados do que nossa solução genérica e do que outros dois métodos não supervisionados propostos na literatura. E é estatisticamente equivalente a um método supervisionado baseado no SVM, tendo a vantagem de não necessitar de dados para treinamento.

Conclusões

Nós apresentamos um novo método que usa informação extraída da Web para resolução de entidades e criação de arquivos de autoridade. Nós propusemos uma sequência de passos para submeter consultas a uma máquina de busca, extrair informação dos documentos no conjunto resposta e criar agrupamentos de registros baseados nas informações extraídas. Nós tiramos vantagem dos sofisticados procedimentos de busca e dos grandes repositórios implementados pelas máquinas de busca modernas. Nosso método é genérico e pode ser aplicado a diferentes tipos de entidades. Nós implementamos um arcabouço configurável contendo operações básicas para resolução genérica de entidades e criação de arquivos de autoridade, que pode ser estendido para aplicações de domínio específico. Ele é útil para simulações, permitindo modelar diferentes estratégias para resolução de entidades e medir seus resultados.

Nossos resultados indicam grandes ganhos na qualidade da resolução de entidades quando aplicados em três bases de dados distintas, alcançando ganhos de até 125% na métrica pairwise F1. Na criação de arquivos de autoridade, nosso método é capaz de estender formas abreviadas e obter nomes canônicos de boa qualidade, e também pode produzir siglas automaticamente. Nós também apresentamos uma solução especializada para resolução de nomes de autores, que obtém melhores resultados quando comparados a métodos de referência não supervisionados, e é estatisticamente empatado com um método supervisionado, o qual requer rotulação humana e muito tempo de treinamento.

Há muitas direções interessantes que podemos seguir para trabalhos futuros. Em nossos experimentos, nós não obtivemos bons resultados usando os atributos conjunto de títulos e conjunto de textos inferidos da Web. Nós pretendemos investigar melhor como usá-los porque eles contêm informação valorosa. Também, nós pretendemos estudar o uso de métodos supervisionados tais como o SVM ou o Random Forests para combinar os pesos e as funções de similaridades.

No campo de bibliotecas digitais, para nossa solução de resolução de nomes de autores, nós pretendemos extrair informações tais como e-mail, endereço, afiliação e nomes completos de autores dos documentos de um único autor para ajudar no processo de desambiguação. Essas informações estão normalmente disponíveis em currículos dos autores. Extração de nomes completos de autores é também importante para criar arquivos de autoridade de nomes de autores. Nós também podemos combinar estratégias para desambiguar nomes de autores e títulos de veículos de publicação ao mesmo tempo. Usando nosso método, nós pretendemos desenvolver melhores estratégias para avaliar a qualidade da pesquisa das instituições brasileiras.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

In large-scale data repositories, it is often hard to recognize distinct references to the same real-world entity. For instance, online product catalogs are expected to generate information related to products of the user interest. However, given that similar labels can be used to refer to distinct products in different Web pages, products that are not related to the user intention might appear in the answer set. To illustrate, while "HP Officejet J3680 All-in-One Printer, Fax, Scanner, Copier", "HP CB071A#A2L J3680 Officejet Multifunction Printer", and "Hewlett Packard Officejet 3680 All-in-One Printer (CB071A)" refer to the same printer, "HP Officejet J4580 All-in-One Printer, Fax, Scanner, Copier" refers to a different one.

Digital libraries face a similar challenge, as they need to keep bibliographic citation metadata collected from several sources. It is usual to find ambiguous author names in bibliographic citations. Ambiguity may occur due to the existence of multiple authors with the same name (polysemes) or different name variations for the same author (synonyms). Also, they need to identify multiple distinct references to the same article. Such problem is referred to in the literature as the "entity resolution problem" [Benjelloun et al., 2008; Bhattacharya and Getoor, 2007; Bilenko et al., 2003; Tejada et al., 2001].

To illustrate, consider the set of bibliographic references of a book composed of several chapters written by different contributors. Chapter authors use distinct variant forms, or labels, to refer to the various conferences and journals containing the papers they cited. For instance, the "International Conference on Very Large Data Bases" can be written of several forms, such as "VLDB", "Conference on Very Large Databases", and "Int. Conf. on Very Large Data Bases (VLDB)". We would like to normalize the labels to avoid confusion and missinterpretation. An appealing solution to normalize labels is to associate with each conference or journal a unique identifier as well as a

list of the variant labels used to refer to it—a data structure known as "authority file" [Auld, 1982; French et al., 2000].

Given many entity references, such as product labels or bibliographic citations, entity resolution is the process of identifying which of them correspond to a real-world entity. A same record, composed of one or more attributes, may contain references to multiple distinct entities. For instance, a bibliographic citation record may contain attributes such as author names, work title, and publication venue title, each attribute corresponding to a distinct entity. That is, we may want to find the entity references that refer to the same publication venue, or that refer to the same author, or that refer to the same article (useful for finding replicated citations)—three distinct entities. In the first two cases, the entity to be disambiguated consists of an attribute of the record, and in the latter case, it consists of the whole record. Thus, it should be clear that the entity to be disambiguated has to be provided as input. We frequently refer to the record composed of attributes as the entity reference. However, it might be that the attributes serve as references to multiple entities. In this case, it is necessary to make explicit which attribute will be considered as the entity reference to disambiguated.

In the case of identifying replicas, some previous works also deal with the problem of merging the replicated records generating a canonical form for each entity [Benjelloun et al., 2008; Wick et al., 2009]. If the entity to be disambiguated consists of only one of the attributes of the records, some previous works generate an authority name (canonical name) for each entity and keep a list of variant forms that can be used to refer to it, generating an authority file [French et al., 2000; Warner and Brown, 2001]. In here, we aim at generating authority files, without concern to the merging of replicated records.

1.1 Objectives of the Thesis

The main objective of this thesis is to study the entity resolution problem and the creation of authority files. We propose a solution that uses information available on the Web as a source of additional knowledge to disambiguate entity references and to obtain canonical names for entities. Our hypothesis is that we can obtain such additional knowledge submitting queries to a Web search engine and extracting information from the documents returned by the queries. We study a generic solution that may be applied to several distinct datasets and a specific solution for author name resolution, a problem faced by digital libraries.

Solutions to the entity resolution problem usually adopt basic attributes of entities

for disambiguating purposes. For example, to disambiguate author names in bibliographic citations, some previous works [Cota et al., 2007; Han et al., 2005] use only basic attributes such as author names, work title, and publication venue title. Other works [Bhattacharya and Getoor, 2007; Huang et al., 2006; Song et al., 2007] use additional attributes such as abstracts, keywords, e-mails, and affiliations of authors. The problem with these approaches is that such additional attributes are sometimes difficult to obtain.

We extract from the Web additional attributes that can be used to disambiguate entities. The Web is a valuable source of evidence where we can find curricula vitae of authors whose references are ambiguous or obtain acronyms that can be used to expand abbreviated publication venue titles. The challenge is to formulate the most suitable queries to a Web search engine and extract from the answer sets information that can be used effectively for disambiguation purposes.

In this thesis, we present an approach to solve the entity resolution problem, which we refer to as WER—Web-based Entity Resolution. We built a generic and configurable framework for dealing with various instances of the entity resolution problem, which allows the easy and convenient development of solutions to new domains. We also discuss a specific solution for author name resolution, encoded using our framework.

1.2 Contributions of the Thesis

The major contributions of the thesis are:

- A new method that uses information available on the Web as a source of additional evidence for entity resolution and creation of authority files [Pereira et al., 2008, 2009a]. This method, which includes specific heuristics to extract pertinent information from Web documents, leads to improved results when compared with state-of-the-art solutions for the problem (Chapter 4);
- A configurable framework that allows conveniently modeling different strategies for entity resolution and assessing their results (Chapter 4);
- An empirical validation of our solution for the entity resolution problem with extensive experiments (Chapter 5);
- A customized implementation of our solution for disambiguation author names (in bibliographic citations) [Pereira et al., 2009b], as well as its comparison with a variant implementation (of the same solution) on our configurable framework (Chapter 6).

The results of this thesis can be used to improve the quality and the efficacy of online product catalog services, in which a user can, for example, compare prices of products. In digital libraries, it may improve the automatic gathering of statistics about electronic documents such as statistics involving authors and their affiliations. In this case, the first step is to reliably identify individuals or institutions before determining how many and what papers were written by each one. Our work may also be useful to identify the profile of scientific publications of institutions. In this case, we can identify publication venues associated with each of the publication of a institution, and then generate statistical data about, for example, venue titles and venue types (journal, conference, workshop) in which an institution publishes more articles.

1.3 Organization of the Thesis

The rest of this thesis is organized as follows. In Chapter 2, we discuss the related work on entity resolution and authority files. In Chapter 3, we present basic concepts that are usefull for a better understanding of the subsequent chapters. In Chapter 4, we describe details of the Web-based approach. In Chapter 5, we report our experiments and their results using three distinct datasets. In Chapter 6, we discuss a specific solution for author name resolution comparing it with our generic solution. Finally, in Chapter 7, we present some conclusions and future work.

Chapter 2

Related Work

In Section 2.1, we review previous research related to the entity resolution problem in its more general approach. As our work also produces authority files, in Section 2.2, we review the literature on this subject. In Section 2.3, we describe the work related to personal name disambiguation, since we propose a specific solution for author name disambiguation.

2.1 Entity Resolution

The entity resolution problem has been studied in various disciplines under different names such as record linkage [NewCombe et al., 1959], identity uncertainty [Pasula et al., 2002], citation matching [Lee et al., 2007], merge/purge [Hernández and Stolfo, 1995], deduplication [Sarawagi and Bhamidipaty, 2002], among others. The first articles on the task of matching equivalent records that differ syntactically (record linkage) are from the 1950s and the 1960s with the works of NewCombe et al. [1959] and Fellegi and Sunter [1969]. The latter formulated a series of statistical methods for dealing with the problem of matching population records, which are the base for subsequent works on record linkage. Febrl [Febrl, 2009] is an example of a tool that implements such methods. A good survey on deduplication can be found in [Elmagarmid et al., 2007].

Works on entity resolution aim at identifying entity references judged to represent the same real-world entity. In some cases, the goal is to identify replicated references whose attribute values differ syntactically [Bilenko et al., 2003; Sarawagi and Bhamidipaty, 2002]. For example, when the objective is to identify customer entities in customer databases coming from different subsidiaries of a company. In other cases, the goal is to identify references to the same entity, but the references

do not represent replicas [Bhattacharya and Getoor, 2007; Cohen, 1998]. For example, when the objective is to allocate bibliographic citations to the corresponding author. The citations allocated to each author are not necessarily replicas. In addition, after identifying replicas, some works deal with the problem of merging records generating a canonical form for each entity [Benjelloun et al., 2008; Wick et al., 2009], whereas some others also obtain an authority name for each entity [French et al., 2000; Pereira et al., 2008]. We do not propose solutions to merge replicated references, but we generate authority files by selecting an authority name for each disambiguated entity.

Traditional approaches to entity resolution compare records measuring the similarity of their attribute values based on a distance metric, such as edit distance or cosine. Several works have studied algorithms for approximate string matching that can be used for entity resolution. Cohen et al. [2003a,b] compared several string metrics for the task of matching names. They evaluated string metrics applied to individual attributes of the entities and proposed a supervised strategy to combine similarity metrics on all attributes to handle entity matching. French et al. [2000] investigated strategies to use traditional string matching applied to affiliations of authors and proposed a combination of Jaccard similarity coefficient with edit distance, which was also used in our experiments generating good results. Lawrence et al. [1999a,b] proposed several algorithms for matching citations from different sources based on edit distance, word matching, phrase matching, and attribute extraction.

Some approaches use adaptive supervised algorithms that learn string similarity measures from labeled data. Bilenko et al. [2003] represent every pair of records using a vector of features that describe similarity between individual record attributes. Their method uses training data in form of matched and unmatched record pairs to train an algorithm for classifying record pairs as duplicate and nonduplicate. Then they use the distance from the hyperplane, in a SVM classifier, as the measure of confidence in the pair of records referring to the same entity.

Based on the same idea for training on matched and unmatched pairs, Tejada et al. [2001] developed an entity identification system that applies a set of general string transformations and an active learning technique that learns rules, using decision trees, for mapping records. In [Tejada et al., 2002], they extended their system to learn to weight transformations for a specific application domain. Another approach using supervised learning was proposed by Pasula et al. [2002] to find replicas of bibliographic citations. They create a relational probabilistic model for the domain that involves constructing a generative model for individual attributes and using a Markov chain Monte Carlo procedure to obtain matching decisions.

Another supervised approach to learn similarity functions is based on genetic

programming [Koza, 1992]. A genetic programming based approach to replica identification in data repositories was developed by de Carvalho et al. [2008]. The idea is to automatically suggest similarity functions that combines the best evidence available among the record attributes, in order to identify two records referring to the same entity.

Supervised machine learning techniques achieve high accuracy but require laborious human labeling and expensive training time. To circumvent the problem of only a small amount of labeled data being available, Blum and Mitchell [1998] proposed a model in which unlabeled data can be used to enlarge the training set. The idea is to use two views of an example that are redudant but not completely correlated in which two learning algorithm are trained separately on each view, and then each algorithm's prediction on new unlabeled examples is used to augment the training set of the other.

Unsupervised techniques have also been proposed. Cohen [1998] presents a query language that identifies references to the same entity providing approximate answers in a database system. His model converters the attribute values in the database into vectors of text and ranks them using the vector space model. Chaudhuri et al. [2005] developed a clustering technique to identify record replicas that captures local structural properties of data to characterize groups of replicated records. However, their solution works well for applications where the sizes of groups of replicas are small, which is not usually the case for applications that collect data from the Web. Like we do, Carvalho and da Silva [2003] also use a linear combination to weight the contribution of each attribute to compose the pairwise similarity function for identifying replicas. They evaluate four distinct strategies to compose the function using the vector space model for computing similarity.

A recent line of works has taken the relationship among records into account. Dong et al. [2005] proposed a generic framework based on a dependency graph that exploits the association among entity references belonging to multiple related classes. First, their method uses the relationship among references to provide evidence for reconciliation decisions. Then, it propagates information between reconciliation decisions for different pairs of references, and after, it addresses the lack of information in each reference by using the enriched information already obtained in the previous steps. Bhattacharya and Getoor [2007] proposed to resolve entities collectively based on the relationship among entity references, as in the bibliographic domain where author names in papers are connected by coauthor links. They use a greedy agglomerative clustering algorithm where, at any stage, the current set of clusters reflects the current belief about the mapping of the references to entities. The similarity between two clusters is dynamic, reflecting the relationship among the entity references in the

clusters.

Considering efficiency and scalability issues, Benjelloun et al. [2008] proposed a generic entity resolution framework that views the functions used to compare and merge records as black-boxes, and they developed strategies that minimize the number of invocations to such black-boxes. Several works follow the theory suggested by Fellegi and Sunter [1969], in which the efficiency is achieved by some type of blocking scheme that reduces the number of record pairs to be compared. The idea is to split records into buckets [Jaro, 1989] or canopies [McCallum et al., 2000], and to compute similarities only among references in the same block. Bilenko et al. [2006] proposed a mechanism to automatically learn the optimal blocking function. Given similarity predicates on different record attributes, their algorithm finds a nearly optimal combination of those predicates as the blocking function. The problem with blocking strategies is that an error in the blocking does related records never being put together. Other works propose to improve the efficiency by increasing the parallelism of the entity resolution [Benjelloun et al., 2007; sik Kim and Lee, 2007]. Our work, on the other hand, focused on improving effectiveness.

Our work uses the Web as a source of additional information for entity resolution, as well as Elmacioglu et al. [2007] do. Their proposal is to use a Web search engine to measure how frequently two entity references appear together with a same information on the Web. If this frequency is high, an entity reference is considered a duplicate of the other. Kan and Tan [2008] examined the problem and the solutions for entity resolution in digital library metadata. They point that the Web is a fruitful source of evidence for entity resolution if carefully cleaned and utilized.

2.2 Authority Files

Our work also produces authority files. Following, we discuss authority control in traditional digital libraries and works on automated authority control.

2.2.1 Authority Control in Traditional Digital Libraries

Authority control is not a new problem [Auld, 1982], but its importance is increasing with the creation of a variety of new digital libraries and bibliographic indexing systems, accessible through the Web. Traditional digital libraries maintain authority files and work to maintain them consistent to reduce ambiguity. To illustrate, the Virtual International Authority File (VIAF) project [VIAF, 2008] combines the authority files

of three institutions (US Library of Congress, the Deutsche Nationalbibliothek, and the Bibliothèque Nationale de France) into a single authority service.

From experience in conducting the US Library of Congress name authority file [Library of Congress, 2009], Tillett [2000] focuses on how the authority control performed by libraries can help the Web and suggest some of the next steps in making this tremendous resource of authority records available and used internationally. In [Tillett, 2001], she explores different models of how a pool of authority records for bibliographic entities may be used on the Web. The objectives of such initiative are: (1) facilitate sharing in order to reduce cataloguing costs for libraries, as well as for museums, archives, rights management agencies, etc, (2) simplify creation and maintenance of authority records internationally, and (3) enable users to access information in the language, the script, and the form they prefer. More recently, Tillett [2004] explores a new view of International Federation of Library Institutions and Associations' Universal Bibliographic Control (UBC) in a future vision of a virtual international authority file as a building block for the Semantic Web, and reinforces the importance of the authority control to improve the precision of searches in large databases or on the Web.

Snyman and van Rensburg [2000] discuss the effort to standardize author names using a unique number, called INSAN. In the INSAN model, an access control file and access records can be created instead of an authority file with authority records. It means that the variation of a name is linked without having any name one declared as authoritative. Variations of a name are simply linked to the author name number and it provides access to the bibliographic records of the publications. However the acceptance of the INSAN is far from being real.

Xia [2006] proposes improvements to the identification of author names in digital repositories. First, he presents an analysis of current name authorities in digital repositories, and some features of existing repository applications. Second, he proposes two solutions to improve the identification of author names: (1) using composite identifiers which combine the name of the author, the date of the publication, and author affiliation, (2) requiring the authors to input the variants of their names, if any, at the time of depositing articles.

There are many other works in literature about authority control and related contexts. Authority control has been exhaustively discussed in papers of the Information Sciences area, in which the main goals are: to propose models to create and maintain catalog systems in traditional digital libraries, to develop searching systems for them, to integrate authority files from different digital libraries, and to use bibliographic entities on the Web. On the point of view of Computer Science, which is the focus of our

work, we are interested in propose solutions that can deal with a specific point in the authority control - the automated creation of authority files.

2.2.2 Automated Authority Control

Our work on creating publication venue authority files is close to the research by French et al. [2000]. They investigate the use of a number of approximate string matching techniques and introduce the notion of approximate word matching for creating authority files. They experimented their techniques creating an authority file of affiliations for researchers in the Astrophysics Data System. They also evaluated the human effort involved in their method. Their techniques can also be applied to publication venue authority files. A key difference between our work and their is that we use citations found on the Web as input data, while they restrict the input to a set of citations provided in the beginning. As a result, our method also produces acronyms and venue types automatically.

Hong et al. [2004] propose a solution to the problem of updating and searching for authors and publication venues in digital libraries. They address three core issues in name authority control: changing in the name of a bibliographic entity, splitting of a bibliographic entity into multiple ones, and merging of multiple bibliographic entities into a single one. Our work neither approaches system support for updating and searching nor presents specific solution to the three core issues in name authority control, although our Web-based method could be adapted to automatically solve these three core issues.

Lee [2007], continuing the previous work, proposes a framework to identify evolving metadata, updating, and searching digital libraries. Objects in digital libraries evolve over time, and consequently, their associated metadata evolves as well. The challenge is to quickly and accurately identify evolving metadata and fix them when needed. The article shows the importance of the problem and investigates scalable algorithmic solution and system support to it. Our work is related to Lee's on the identification of evolving metadata, presenting a solution based on authority files, but we do not address the system incorporation of the identified evolving metadata.

Detection of name variants is not a problem restricted to bibliographic citations. DiLauro et al. [2001] and Warner and Brown [2001] use data mining and several types of evidence to disambiguate names and create an automated name authority control of composers, lyricists, and arrangers in the Levy Sheet Music Collection. The name disambiguation system is a learning system which requires training with a representative subset of the collection on which it will operate, differing from our work, which does

not require training.

Davis et al. [2003] describe a system that locates the occurrences of named entities within a text, when given *a priori* a set of related names included in authority lists. A tool was developed to identify references to a single art object in texts related to images of that object in a digital collection. The tool, using computational linguistic techniques, takes as input an art object identifier and sequentially removes modifiers, making it become more general, and then locates occurrences of the variant terms among the noun phrases in the text. In our work, we also need to extract variant names from a text, although our goal is different. While they aim at finding object identifier variants to say whether a text is about an object, we take that a text returned by a search engine is supposed to be about this object and our purpose is to find variant names and other evidences referring to it.

2.3 Personal Name Disambiguation

Entity resolution when applied to solve references to people is called personal name disambiguation, and specific solutions have been studied for such problem. In this case, the records are not usually replicated and we want simply to find references belonging to each person. Following, we discuss solutions for personal name disambiguation on the Web and for author name disambiguation in bibliographic citations.

2.3.1 Personal Name Disambiguation on the Web

Submitting queries to a Web search engine looking for pages of a specific person is one of the most common query types nowadays [Kalashnikov et al., 2008]. However, pages related to several distinct namesakes are often returned, and the most popular search engines do not cluster the answer set, leaving the task of disambiguation to the user. Deciding which web pages refer to each person is a specific case of entity resolution. This problem differs from the traditional entity resolution, and from our work, because the input records (the Web pages) do not have explicit attributes - an algorithm needs to infer and extract them.

Personal name disambiguation on the Web relies mainly on external features such as social networks, link structures, and attributes extracted from documents such as e-mail, postal address and phone numbers to identify real individuals. Bekkerman and McCallum [2005] present two unsupervised frameworks for disambiguating people with the same name. The first method is based on the link structure of Web pages, and the second method employs a cluster technique named Agglomera-

tive/Conglomerative Double Clustering. Differently from works that search for people individually, they treat the problem of disambiguating a group of people who are known to be somewhat connected by a social network, and use this extra information to aid the disambiguation.

Bollegala et al. [2008b] propose an unsupervised algorithm which extracts uniquely identifying key phrases from the Web to disambiguate people with the same name. These phrases could then be added to the query to narrow down the search to a specific namesake. The first step of their method is to collect a set of documents that cover all the namesakes for a given name. They assume that each document in the collection represents exactly one namesake. Then, these documents are clustered such that each cluster represents a different namesake, and key phrases are extracted from each cluster to identify the namesake it represents.

Bollegala et al. [2008a] propose an approach to find aliases of a given name from the Web. Alias is the term used to describe words or multi-word expressions that are used to refer to an entity. They explore the problem of finding various references to a particular entity. They exploit a set of known names and their aliases as training data, and extract lexical patterns that convey information related to aliases of names from text snippets returned by a Web search engine. The patterns are then used to find candidate aliases of a given name. The extracted candidates are ranked using various ranking scores computed using the hyperlink structure on the Web and page-counts retrieved from a search engine.

Kalashnikov et al. [2008] present an approach to cluster Web pages referring to the same person. They developed a technique for classifying co-occurrence information collected from the Web. For those pairs of Web pages that are not sufficiently similar to group together, their algorithm submits queries to a Web search engine that combines information extracted from each page. The co-occurrence counts in the answer sets are used as evidence to decide whether the pairs are related or not.

2.3.2 Author Name Disambiguation

Our specific solution for author name resolution deals with the problem of disambiguating author names in the context of bibliographic citations generally found in digital libraries. In this context, basic metadata from citations are normally available, such as author and coauthor names, work titles and publication venue titles. Lee et al. [2005] and Lee [2007] define the terminology for the problems of polysemes and synonyms in the context of digital libraries, which they named, respectively, mixed citation and split citation problems, and they present distinct solutions for them. Some works deal

with only one of these problems. In [Kang et al., 2009] and [Tan et al., 2006], for example, the authors deal only with the mixed citation problem, although the approach used by the latter can also be applied to solve the split citation problem. On et al. [2005], on the other hand, deal only with the split citation problem. Their main objective is to present blocking schemes to reduce the number of comparison among names, diminishing the time complexity for disambiguating.

Some works use only basic citation metadata to disambiguate author names [Cota et al., 2007; Han et al., 2004, 2005]. Han et al. [2004] present two methods based on supervised learning techniques. The first method uses a naive Bayes inference model and the second one is based on Support Vector Machines (SVM). Han et al. [2005] use the K-way spectral clustering method to disambiguate author names. They evaluated the contribution of each citation attribute (coauthor names, work title, and publication venue title) on name disambiguation, besides investigating the effect of other characteristics such as size of the dataset and schemes of feature weighting. In [Cota et al., 2007], the authors propose a heuristic-based hierarchical clustering method to deal with the author name disambiguation problem. The method successively fuses clusters of citations of compatible authors based on several heuristics and similarity functions on the citation metadata. In each phase, the information of fused clusters is aggregated providing more information for the next round of fusion.

Other works use additional metadata such as abstracts and keywords of works, e-mails and affiliations of authors to disambiguate author names [Huang et al., 2006; Song et al., 2007]. The problem with this approach is that these additional metadata are sometimes difficult to obtain. Using such metadata, Huang et al. [2006] propose a framework in which a blocking method first creates candidate classes of authors with similar names and then DBScan, a density-based clustering method, is used for clustering citations by author. Likewise, Song et al. [2007] propose to use a probability distribution of topics for author name disambiguation. Their method is based on probabilistic latent semantic analysis and latent Dirichlet allocation models to learn a probability distribution of topics.

Besides basic metadata, our work also uses additional information obtained from the Web. This same idea is also used in [Kang et al., 2009] and [Tan et al., 2006]. Kang et al. [2009] explore the net effects of coauthorship on author name disambiguation and use a Web-assisted technique of acquiring implicit coauthors of the target author to be disambiguated. Pairs of names are submitted as queries to Web search engines to retrieve documents that contain both author names, and these documents are scanned to extract new author names as coauthors of the original pair. As in our work, they also had problems trying to extract information from documents and to

identify documents that contain publications. To solve the former, they use a simple regular expression matching strategy, which does not scale. However, they do not deal with the latter problem, suggesting further investigation as future work. In our work, we propose to use a local search engine as a scalable solution to extract information from documents, and we use some heuristics to identify documents that contain publications of a single author.

Tan et al. [2006] also make use of input citations to submit queries to a Web search engine and then use the URLs in the answer sets as features to disambiguate author names, considering that common URLs returned by queries related to distinct citations may indicate that the citations are of the same author. The difference to our work is that they do not exploit the content of the documents returned by the queries, what may mislead the results since many returned URLs do not refer to documents containing publications or the documents are not of a single author.

Chapter 3

Basic Concepts

A typical entity resolution algorithm is composed of a pairwise similarity function and a clustering algorithm. The pairwise similarity function compares two entity references and returns how similar they are. The comparison is based on a similarity metric applied to some of attributes of the input records. For string type attributes, many string-based similarity functions have been proposed in the literature. The clustering algorithm then clusters records based on their similarities. If it works propertly, the records in the same cluster will all be relative to the same entity of the world (ideal case). Non-pairwise entity resolution algorithms using classification and clustering techniques have also been proposed in the literature. For a better understanding of such concepts and of the subsequent chapters, Section 3.1 describes string-based similarity functions, Section 3.2 introduces pairwise similarity functions, and Section 3.3 presents classification and clustering techniques.

3.1 String-based Similarity Functions

String-based similarity functions map a pair of strings s_i and s_j into a real number r, where a larger value of r indicates higher similarity between s_i and s_j. The number r can be normalized to produce a result between zero and one, where zero means no similarity and one means totally similar strings. The reader can also find the term *distance functions* in literature, which is analogous to similarity functions, except that smaller values indicate higher similarity.

String-based similarity functions can be separated into two groups, according to the techniques adopted to measure similarities: character-based techniques and token-based techniques. The former rely on character edit operations, such as deletions, insertions, substitutions and subsequences comparison, while the latter transform strings

into sets (or bags) of tokens (where each token is a word) on which similarity computations are conducted. There are also hybrid similarity functions that consider a combination of two other similarity functions. We detail some of these similarity functions in the next sections. Other functions and a comparison among them can be found in [Bilenko et al., 2003; Cohen et al., 2003a,b].

3.1.1 Character-based Similarity Functions

Following, we describe edit distance, Jaro, and Jaro-Winkler character-based similarity functions.

3.1.1.1 Edit Distance Similarity

The best-known character-based similarity function is Levenshtein distance [Levenshtein, 1966]. The Levenshtein distance between two strings is given by the minimum number of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character. For example, the Levenshtein distance between "University of Virginia" and "University of Victoria" is 4, since we need at least four edits to change one into the other. A generalization of the Levenshtein distance (Damerau-Levenshtein distance) [Damerau, 1964] allows the transposition of two characters as an operation. Edit distance was introduced by Vladimir Levenshtein who generalized this concept with multiple edit operations, but did not include transpositions in the set of basic operations. Damerau [1964] also noted that over 80% of all spelling errors are because of one of the four simple editing errors: insertion, deletion, substitution, and transposition. So, whenever the edit distance is equals to 1, it is almost certainly a spelling variant or misspelling.

From now on, we call the Damerau-Levenshtein distance simply as edit distance. The most common way of calculating edit distance is by the dynamic programming approach, which is $O(mn)$ time complexity, where m and n are the length of the smallest and the largest strings, respectively. If the edit distance is required to be at most k, then the best practical results are $O(kn)$ time complexity in the worst case [Baeza-Yates and Navarro, 1998].

An edit distance function can be converted into a similarity function as follows. Let s_i and s_j be two strings, $ed(s_i, s_j)$ be the edit distance between s_i and s_j, and $min(|s_i|, |s_j|)$ be the length of the shortest string. Then, the edit distance similarity

between s_i and s_j is given by:

$$edSim(s_i, s_j) = 1 - \frac{ed(s_i, s_j)}{min(|s_i|, |s_j|)} \tag{3.1}$$

3.1.1.2 Jaro and Jaro-Winkler Similarities

Jaro similarity [Jaro, 1989, 1995] is based on the number and order of common characters between two strings. Given strings $s_i = a_1...a_K$ and $s_j = b_1...b_L$, define a character a_l in s_i that matches with s_j if and only if there is a $b_p = a_l$ in s_j such that $l - H \leq p \leq l + H$, where $H = min(|s_i|, |s_j|)/2$. Let $s'_i = a'_1...a'_K$ be the characters in s_i that match with s_j (in the same order they appear in s_i), and let $s'_j = b'_1...b'_L$ be characters in s_j that match with s_i. Then define a transposition for s'_i, s'_j as a position l such that $a'_l \neq b'_l$. Let $T_{s'_i,s'_j}$ be one-half the number of transpositions for s'_i and s'_j. The Jaro similarity for s_i and s_j is:

$$Jaro(s_i, s_j) = \frac{1}{3} \times \left(\frac{|s'_i|}{|s_i|} + \frac{|s'_j|}{|s_j|} + \frac{|s'_i| - T_{s'_i,s'_j}}{s'_i} \right)$$

Jaro-Winkler similarity [Winkler, 1999] is a variant of the Jaro similarity that uses a prefix scale p to give higher ratings to strings that match from the beginning up to a prefix length l. The Jaro-Winkler similarity for two strings s_i and s_j is:

$$Jaro\text{-}Winkler(s_i, s_j) = Jaro(s_i, s_j) + p \times l \times (1 - Jaro(s_i, s_j))$$

l is the length of common prefix and p is a constant scaling factor for how much the score is adjusted upwards for having common prefixes. The standard value for this constant in Winkler's work is $p = 0.1$.

Jaro-Winkler similarity emphasizes the matches in the first few characters. The Jaro and Jaro-Winkler was developed primarily for short strings such as personal names.

3.1.2 Token-based Similarity Functions

Following, we describe Jaccard and cosine token-based similarity functions.

3.1.2.1 Jaccard Similarity Coefficient

Jaccard similarity coefficient is a token-based (or word-based) similarity function used to quantify the similarity between two strings. More formally, let $s_i \cap s_j$ be the set formed by intersecting the words in the strings s_i and s_j. Analogously, let $s_i \cup s_j$ be the

set formed by the union of the words in the strings. The Jaccard similarity coefficient $J(s_i, s_j)$ between the strings s_i and s_j is given by:

$$J(s_i, s_j) = \frac{|s_i \cap s_j|}{|s_i \cup s_j|} \tag{3.2}$$

where $|s_i \cap s_j|$ and $|s_i \cup s_j|$ are the number of words in each set.

Differently from edit distance similarity, Jaccard similarity coefficient considers neither the order of the words in a string nor repeated words. Although edit distance is robust for spelling variants, it can be completely defeated by permutations of words, in which case Jaccard similarity coefficient may work better.

3.1.2.2 Cosine Similarity

Cosine similarity is also a token-based similarity function that measures the similarity between a pair of strings using the vector space model [Baeza-Yates and Ribeiro-Neto, 1999]. Cosine similarity is usually used to measure the similarity between a query and each document in a collection, or to measure the similarity between two documents. In the context of this thesis, we describe the cosine similarity simply as a measure of similarity between two strings, being that a string may be extracted from a document.

Let $\mathcal{S} = \{s_1, s_2, ..., s_N\}$ be a collection of N strings. Let $\mathcal{K} = \{k_1, k_2, ...k_t\}$ be the set of all distinct words (or tokens, or index terms) that appear in strings of $\mathcal{S}$. To each pair (k_l, s_j), $k_l \in \mathcal{K}$, $s_j \in \mathcal{S}$ is associated a weight $w_{l,j} \geq 0$. A string s_j is, thus, represented as a vector of the word weights $\vec{s}_j = (w_{1,j}, w_{2,j}, ..., w_{t,j})$, where t is the total number of distinct words in the entire string collection. Each $w_{l,j}$ represents the importance of word k_l to describe the semantic content of the string s_j. Therefore, the strings are represented as t-dimensional vectors. Figure 3.1 shows the vectors corresponding to two strings s_i and s_j, in a space with two dimensions, i.e., containing two words k_1 and k_2.

The degree of similarity between two strings s_i and s_j is evaluated as the correlation between the vectors $\vec{s}_i$ and $\vec{s}_j$. This correlation can be quantified by the cosine of the angle between these two vectors, given by:

$$sim(s_i, s_j) = \frac{\sum_{l=1}^{t} w_{l,i} \times w_{l,j}}{\sqrt{\sum_{l=1}^{t} w_{l,i}^2} \times \sqrt{\sum_{l=1}^{t} w_{l,j}^2}} \tag{3.3}$$

The weight $w_{l,j}$ can be computed as [Baeza-Yates and Ribeiro-Neto, 1999]:

$$w_{l,j} = \frac{freq_{l,j}}{max_p \ \ freq_{p,j}} \times \log \frac{N}{n_i}$$

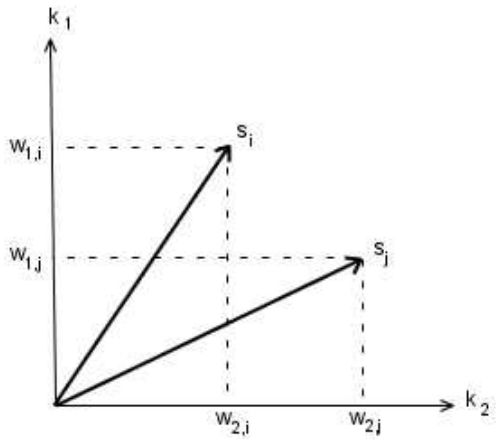

Figure 3.1. Representation of two strings s_i and s_j in the vector space model, considering only two words k_1 and k_2.

where $freq_{l,j}$ is the raw frequency of the word k_l in the string s_j, the maximum is computed over all words in the string s_j, n_i is the number of string in which k_l occurs, and N is the total number of strings in the collection. This word-weighting strategy is called *term-frequency* and *inverse-document-frequency*, *(tf-idf)* scheme. The term frequency reflects the idea that a word is more important in a string if it occurs many times in it. The inverse document frequency measures the rarity of the word k_l in the collection $\mathcal{S}$.

Differently from Jaccard similarity coefficient in which the similarity depends only on the set of words in the two strings being compared, the cosine similarity between two strings depends on the words in the other strings in a collection, since each word receives a weight that depends on its frequency and rarity in the collection.

3.1.3 Hybrid Similarity Functions

Similarity functions can be combined forming hybrid methods. Following, we describe some of such functions.

3.1.3.1 Edit Distance Jaccard Similarity

French et al. [2000] propose to combine edit distance with Jaccard similarity coefficient, relaxing the Boolean conditions for computing the intersection and union of two strings s_i and s_j in Equation 3.2. This is done as follows:

$$s_i \cap s_j = \{w \in s_i \mid \exists z \in s_j \wedge min(edSim(w, z)) < \delta\}$$

where w and z are words inside the strings and δ is a threshold on the edit distance. The union string $s_i \cup s_j$ is computed analogously. By using this approximate intersection and union operations to compute Jaccard similarities, they actually make use of an edit-distance-jaccard metric.

3.1.3.2 SoftTF-IDF Similarity

Bilenko et al. [2003] describe a "soft" version of the cosine similarity, called softTF-IDF, in which similar words are considered as well as words that appear in both strings to be compared.

Let sim' be a secondary similarity function that performs well on short strings (such as edit distance or Jaro-Winkler). Also, let s_i and s_j be the strings to be compared. Define $CLOSE(\theta, s_i, s_j)$ as the set of words $l \in s_i$ such that exists some $v \in s_j$ satisfying $sim'(l, v) > \theta$. Then:

$$softTF\text{-}IDF(s_i, s_j) = \frac{\sum_{l \in CLOSE(\theta, s_i, s_j)} w_{l,i} \times w_{l,j} \times max_{v \in s_j} sim'(l, v)}{\sqrt{\sum_{l=1}^{t} w_{l,i}^2} \times \sqrt{\sum_{l=1}^{t} w_{l,j}^2}}$$

where $w_{l,j}$ and t are as defined in Section 3.1.2.2.

3.1.3.3 Level Two Similarity

Monge and Elkan [1996] propose a recursive matching scheme for comparing two strings s_i and s_j. First, s_i and s_j are broken into substrings $s_i = a_1...a_K$ and $s_j = b_1...b_L$. Then, a level two similarity is defined as

$$l2Sim(s_i, s_j) = \frac{1}{K} \sum_{i=1}^{K} max_{j=1}^{L} sim'(a_i, b_j)$$

where sim' is some secondary similarity function.

3.2 Pairwise Similarity Functions

Pairwise similarity functions compare two records and return how similar they are. The comparison is based on a similarity metric applied to some attributes of the records. The simplest way of comparing two records is to represent each one as a concatenation of its attribute values, considering it as a single string, and to use a string-based similarity function as a pairwise similarity function. Another way is to consider each record

as composed of multiple attributes, to compute the similarity for each corresponding attribute, and to combine similarities from individual attributes in a meaningful manner. Segmenting the record in multiple attributes means that we may apply distinct similarity functions for each one of the attributes and fine tune the contribution of each one to compose the final pairwise similarity function.

Pairwise similarity functions may use an unsupervised or a supervised approach as we explain next.

3.2.1 Unsupervised Pairwise Similarity Functions

In an unsupervised approach, we specify a static strategy to compute the pairwise similarity function. A very simple strategy that exploits attributes of records is to use the average distance between corresponding attributes. Let $r_i.A_1, r_i.A_2, ..., r_i.A_n$ be n attributes of a record r_i, and let $sim(r_i.A_k, r_j.A_k)$ be a similarity function between the values of the attribute k of the records r_i and r_j. Then, the pairwise similarity function between r_i and r_j is computed as:

$$sim(r_i, r_j) = \frac{1}{n} \sum_{k=1}^{n} sim(r_i.A_k, r_j.A_k)$$

Another strategy is to weight the contribution of each attribute to compute the pairwise similarity function as follows:

$$\begin{cases} sim(r_i, r_j) = \sum_{k=1}^{n} w_k \times sim(r_i.A_k, r_j.A_k) \\ \sum_{k=1}^{n} w_k = 1 \end{cases}$$

where w_k is a weight to fine tune the contribution of the attribute A_k.

In both strategies, distinct similarity functions may be applied to each one of the attributes. One of the difficulties is to select the most appropriate similarity function to be applied to each attribute and its corresponding weight for a specific problem (or dataset). Bilenko et al. [2003] and Cohen et al. [2003a] showed that even strategies that have been tuned and tested on many previous problems can perform poorly on new and different problems. Unsupervised strategies, by nature, do not include previous knowledge about the specific problem they are solving. In practice, we would experiment several combinations of functions and weights on a sample of the dataset and choose suitable values to be applied to that dataset.

3.2.2 Supervised Pairwise Similarity Functions

In a supervised approach, knowledge about the problem to be solved is introduced into the similarity function. The idea is to provide the algorithm a set of labeling data (training data), which is used to learn strategies and then to label new entries (testing data). In spite of supplying the deficiency of unsupervised approaches, supervised methods only work well if a large amount of labeling data is available, and it comes from the same distribution as the testing data, what is usually unfeasible in large-scale datasets.

Supervised classifiers such as Support Vector Machine (SVM) and decision trees have been used as the pairwise similarity function [Bilenko and Mooney, 2003; Huang et al., 2006; Treeratpituk and Giles, 2009] (see more details about classifiers in Section 3.3.1).

A classifier can be used as a pairwise similarity function as follows. Using a set of labeling data as matching and nonmatching, the classifier can use learning methods to adaptively find a combined similarity function that is the most appropriate for a particular problem. Specifically, record pairs can be represented as feature vectors, using as features the similarities between corresponding attributes. Then, a classifier is trained using these feature vectors, and the learned classifier's confidence in the match class is used as the pairwise similarity metric.

Given a set of records composed of k attributes and a set of m similarity metrics, Bilenko and Mooney [2003] for example, represented any pair of records by an mk-dimensional vector. Each component of the vector represents the similarity between two attribute values of the records that is calculated using one of the m similarity metrics. They trained a binary SVM classifier using these feature vectors, and then used the distance from the hyperplane as the measure of confidence in the pair of records referring to the same entity. Treeratpituk and Giles [2009] implemented a similar idea using random forest, a classifier that combines a collection of decision trees.

Genetic programming [Koza, 1992] has also been used as an approach to find pairwise similarity functions. Genetic programming is an evolutionary algorithm-based methodology inspired by biological evolution to find computer programs that perform a user-defined task. It starts with a population of randomly created computer programs and iteratively applies the Darwinian reproduction operation and the genetic crossover (sexual recombination) operation, in order to breed better individual programs.

Genetic programming approach can be used to automatically generate pairwise similarity functions. In the entity resolution problem, a generated pairwise similarity function is able to combine and weight the best evidence available among the attributes

of the records, in order to identify two records referring to the same entity. That is, genetic programming approach is able to choose which attributes give the best contributions, which underlying similarity metrics must be applied to each attribute, and how to weight the contribution of each one. Another advantage of genetic programming is the fact that the final pairwise similarity function may use less attributes and less similarity metrics than other approaches, being more efficient, since time is spent computing the similarities between the values of only the most useful attributes. de Carvalho et al. [2008] experimented to use the genetic programming approach to deduplicate records, obtaining good results.

3.3 Classification and Clustering

Classification, which is also referred to as categorization [Sebastiani, 2002], is the task of automatically applying labels to data. For example, classifying a document as belonging to class *computer network* or to the class *database*. Clustering is the task of grouping related items together based on a similarity measure. The items do not necessarily correspond to a predefined class such as computer network or database. Clustering allows to uncover the implicit structure in the data, which can be used to relate the items.

Classification usually adopts a supervised learning approach. In this case, a set of fully labeled items (called training set) is provided to the algorithm, so it can learn a model. This model is then used to label new unlabeled items (called test set). Clustering, on the other hand, adopts an unsupervised learning approach, i.e., clustering algorithms learn entirely based on unlabeled data. The next two sections detail the concepts of classification and clustering.

3.3.1 Classification

Classification is a two-step process [Han and Kamber, 2000]. In the first step, from a set of labeled items, a model is built describing a predefined set of classes. Typically, the learned model is represented in the form of classification rules, decision trees, or mathematical formulae. In the second step, the model is used for classification. Firstly, the predictive accuracy of the model (or classifier) is estimated. The accuracy of a model on a given test set is the percentage of test set samples that are correctly classified by the model. For this purpose, class-labeled samples that are independent of the training samples are randomly selected. If the accuracy of the model is acceptable, the model can be used to classify unknown data.

Bayesian classifiers are examples of statistical classifiers, based on Bayes theorem [Montgomery and Runger, 1999]. They can predict class membership probabilities, such as the probability that a given sample belongs to a particular class. Naive Bayesian classifiers assume that the effect of an attribute value on a given class is independent of the values of the other attributes, which simplifies the computations involved. In spite of their naive design and apparently over-simplified assumptions, naive Bayes classifiers often work well in many complex real-world situations. Bayesian belief networks are also used for classification [Han and Kamber, 2000]. Differently from naive Bayesian classifiers, Bayesian belief networks allow the representation of dependencies among subsets of attributes.

A decision tree is a tree-like structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and leaf nodes represent classes or class distributions [Michell, 1996]. The topmost node in a tree is the root node. In order to classify an unknown sample, the attribute values of the sample are tested against the decision tree. A path is traced from the root to a leaf node that holds the class prediction for that sample. Decision trees can easily be converted into classification rules.

A SVM classifier [Vapnik, 1995] is defined over a vector space where the problem is to find a hyperplane with the maximal margin of separation between two classes. Classifying a sample corresponds to determining its position relative to this hyperplane.

As an example of using classification techniques for entity resolution, Han et al. [2004] present two supervised learning approaches for disambiguating author names in bibliographic citations. One approach uses the naive Bayes probability model and the other uses SVM.

Classification approaches can be applied to the entity resolution problem when the classes are known a priori. However, in the most practical situations, specially in large datasets, the classes are not previously known. For these cases, clustering approaches are more appropriated.

3.3.2 Clustering

Clustering is a data mining technique [Han and Kamber, 2000] that aims at taking a set of unlabeled records and grouping (clustering) them together in a way that records in the same cluster are similar to each other with respect to a given similarity measure, and records in different clusters are dissimilar.

We discuss some clustering methods in Section 3.3.2.1. We also describe in Section 3.3.2.2 evaluation metrics used to quantify how well the clusters were created.

3.3.2.1 Clustering Methods

Records in a clustering procedure are usually represented by a feature vector or via a similarity measure specified between any pair of records, providing a similarity matrix. For example, a set of documents (records) to be clustered may be represented by feature vectors whose features are the distinct words in the set of documents, and whose feature weights are obtained by the tf-idf scheme as described in Section 3.1.2.2. Then, the cosine similarity can be used as metric to cluster the documents. The set of documents may also be represented by a similarity matrix, where each position i, j (row i, column j) represents a similarity measure between the documents i and j. As a similarity metric, we can use any one of those described in Section 3.1. For records composed of multiple attributes, we can use similarity metrics as described in Section 3.2.

After representing the records, a clustering method is used to determine how to assign records to clusters. Several methods have been proposed in literature for clustering data. Good surveys of those methods can be found in [Jain et al., 1999; Berkhin, 2002; Chakrabarti, 2003; Croft et al., 2009]. Weka [Witten and Frank, 2005], a data mining software, implements several clustering algorithms.

Following, we describe two methods that we use in this thesis. Both methods require pairwise similarity functions.

Hierarchical Agglomerative Clustering Method

Hierarchical clustering is a clustering approach that builds clusters in hierarchical fashion, which may be represented in a tree structure called a dendrogram (Figure 3.3). Algorithms for hierarchical clustering are generally either agglomerative (bottom-up approach) or divisive (top-down approach) [Croft et al., 2009]. A hierarchical agglomerative clustering (HAC) algorithm starts with each record as a separate cluster, i.e., it begins with n clusters, each of which contains a single record. Then, the algorithm progressively merges clusters, until the desired number of clusters or a threshold in the similarity measure used to compare clusters to be reached. A hierarchical divisive clustering algorithm, on the other hand, begins with a single cluster that consists of all of the records, and then, progressively splits clusters, until the desired number of clusters or a threshold in the similarity measure is reached. Following, we detail the HAC algorithm.

Algorithm 1 shows how the HAC method works. Each record r_i is initially put into a single cluster, then in each iteration the algorithm calculates the similarity between each pair of clusters and selects the most similar pair to be merged. This process continues until the desired number of clusters or a threshold of similarity is

reached.

Algorithm 1 HAC - Hierarchical Agglomerative Clustering algorithm

Require: Records $\mathcal{R} = \{r_1, r_2, ..., r_n\}$;
Require: stop condition: the number k of clusters or a similarity threshold δ
default: $k = 1$, $\delta = 0$;
Ensure: Clusters $\mathcal{C} = \{c_1, c_2, ..., c_p\}$;

1: **for each** record r_i **do**
2: create new cluster c_i;
3: $c_i \leftarrow r_i$;
4: **end for**
5: $p \leftarrow n$;
6: **repeat**
7: **for each** cluster-pair(c_i, c_j) **do**
8: calculate the similarity $sim(c_i, c_j)$
9: **end for**
10: find the most similar cluster-pair(c_u, c_v)
11: **if** $sim(c_u, c_v) > \delta$ **and** $p > k$ **then**
12: merge clusters c_u and c_v
13: $p \leftarrow p - 1$;
14: **end if**
15: **until** $sim(c_u, c_v) \leq \delta$ **or** $p = k$

Records to be clustered can be represented as points in a multi-dimensional space. Figure 3.2 illustrates an example of the representation of 11 records in a two-dimensional space grouped in four clusters. The more similar their data points are, the earlier they are grouped together in a cluster. And Figure 3.3 shows a dendrogram example of the HAC method in which these same 11 records are merged until reaching only one cluster containing all records. By choosing a suitable level of similarity (threshold) we can get the desired number of clusters.

The function that calculates the similarity between a pair of clusters (line 7 of Algorithm 1) can be computed using one of the following three possibilities:

- **single link**: the similarity value between two clusters is computed as the maximal similarity value (i.e. minimal distance value) between a record of the first cluster and a record of the second cluster;

- **average link**: the similarity value between two clusters is computed as the average of similarity values between the records of the first cluster and the records of the second cluster;

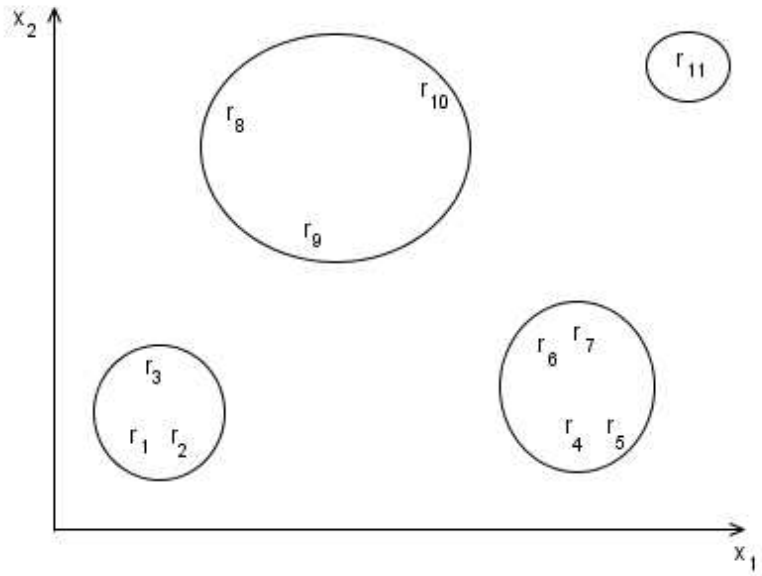

Figure 3.2. Points of 11 records representing four clusters.

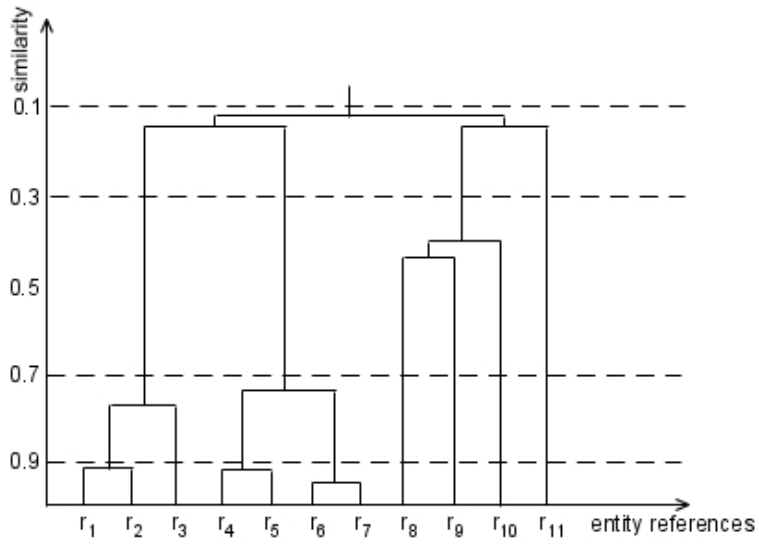

Figure 3.3. A dendrogram example presenting the progressive merge of clusters. A level of similarity higher than a threshold indicates the desired number of clusters. (Based on [Chakrabarti, 2003].

- **complete link**: the similarity value between two clusters is computed as the minimal similarity value (i.e. maximal distance value) between a record of the first cluster and a record of the second cluster.

The single link approach suffers from a chaining effect in which clusters may be forced together due to single record being close to each other, even though many of the records in each cluster may be very distant to each other. And the complete link approach suffers from a migration effect in which records in the regions of borders of clusters are not clustered as it should be.

K-Nearest Neighbor Clustering Method

A clustering procedure using the K-nearest-neighbor (KNN) [Karypis et al., 1999] method models records as a graph. Each vertex of the KNN graph represents a record r_i. An edge exists between two vertices v and u if u is among the k most similar points, higher than a given threshold, of v, or v is among the k most similar points, higher than a given threshold, of u. Each connected component of the graph represents a cluster.

Figure 3.4 illustrates the 1-, 2-, and 3-nearest-neighbor graphs of a simple data set [Karypis et al., 1999]. Representing the items to be clustered using a KNN graph captures the underlying population density of the space. Items in denser and sparser regions are modeled uniformly.

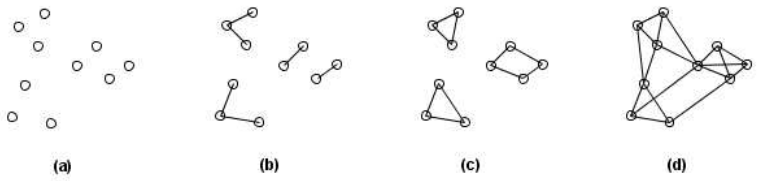

Figure 3.4. K-nearest-neighbor graphs from original data in a two-dimensional space: (a) original data, (b) 1-, (c) 2-, and (d) 3-nearest-neighbor graphs. (Taken from [Karypis et al., 1999]).

Algorithm 2 shows how the KNN method works. Each record r_i is initially put into a single cluster. The similarity between each pair of records is calculated, and then, for each record, its k most similar records are obtained. If the similarity between them is higher than a threshold, then the clusters containing those records are merged.

Algorithm 2 K-Nearest Neighbor Clustering algorithm

Require: Record $\mathcal{R} = \{r_1, r_2, ..., r_n\}$;
Require: the number k of neighbors;
Require: the similarity threshold δ;
Ensure: Clusters $\mathcal{C} = \{c_1, c_2, ..., c_p\}$;

1: **for each** record r_i **do**
2: create new cluster c_i;
3: $c_i \leftarrow r_i$;
4: **end for**
5: **for each** record-pair(r_i, r_j) **do**
6: calculate the similarity $sim(r_i, r_j)$
7: **end for**
8: **for each** record r_i **do**
9: find the k most similar record-pair(r_i, r_j)
10: **for each** r_j among the k most similar records to r_i **do**
11: **if** $sim(r_i, r_j) > \delta$ **then**
12: merge clusters containing records r_i and r_j
13: **end if**
14: **end for**
15: **end for**

3.3.2.2 Clustering Evaluation Metrics

Clustering quality can be evaluated based on the following metrics: K metric, pairwise F1, cluster F1, and ratio of cluster size [Cota et al., 2007; Huang et al., 2006; Lapidot, 2002; Song et al., 2007]. The K metric consists of the geometric mean between average cluster purity (ACP) and average entity purity (AEP), determining the equilibrium between the two metrics. ACP evaluates the purity of the generated clusters with respect to reference clusters manually generated, i.e., whether the generated clusters include only records belonging to the reference clusters. AEP evaluates the level of splitting of one entity into several clusters, i.e., how fragmented the generated clusters are. They are defined as:

$$ACP = \frac{1}{N} \sum_{i=1}^{q} \sum_{j=1}^{R} \frac{n_{ij}^2}{n_i}$$

$$AEP = \frac{1}{N} \sum_{j=1}^{R} \sum_{i=1}^{q} \frac{n_{ij}^2}{n_j}$$

$$K = \sqrt{ACP \times AEP}$$

where N is the number of records in the test dataset, R is the number of reference

clusters (manually generated), q is the number of clusters automatically generated, n_{ij} is the number of elements of cluster i belonging to cluster j, and n_i is the number of elements of cluster i.

Pairwise F1 (pF1) is defined as the harmonic mean of pairwise precision (pp) and pairwise recall (pr), where pairwise precision is measured by the fraction of pairwise records associated with the same entity in the clusters, and pairwise recall is the fraction of pairwise records associated with the same entity placed in the same cluster. They are defined as:

$$pp = \frac{\sum_{i=1}^{q} \sum_{j=1}^{R} C(n_{ij}, 2)}{\sum_{i=1}^{q} C(n_i, 2)}$$

$$pr = \frac{\sum_{i=1}^{q} \sum_{j=1}^{R} C(n_{ij}, 2)}{\sum_{j=1}^{R} C(n_j, 2)}$$

$$pF1 = \frac{2 \times pp \times pr}{pp + pr}$$

where $C(n, r)$ is the combination of r elements from a set of n elements, $C(n, r) = \frac{n!}{r! \times (n-r)!}$, $n \geq r$.

Likewise, cluster F1 (cF1) is a harmonic mean of cluster precision (cp) and cluster recall (cr), where cluster precision is the fraction of totally correct clusters to the number of clusters retrieved, and cluster recall is the fraction of true clusters retrieved. They are defined as:

$$cp = \frac{m}{q}$$

$$cr = \frac{m}{R}$$

$$cF1 = \frac{2 \times cp \times cr}{cp + cr}$$

where m is the number of totally correct clusters.

The ratio of cluster size (RCS) is defined as the number of clusters retrieved divided by the number of true clusters, as follows:

$$RCS = \frac{q}{R}$$

Chapter 4

The WER Approach

In the first section of this chapter, we exemplify the entity resolution problem. In the following sections, we present the formulation of the solution to the problem, detail its main steps, and discuss the WER—Web-based Entity Resolution framework.

4.1 Examples of Entity Resolution

In our examples, we consider the problem of resolving ambiguity in bibliographic citations from digital libraries such as DBLP [Ley, 2002] and CiteSeerX [Giles et al., 1998]. A citation contains bibliographic information such as author names, work title, publication venue title, and year published. Digital libraries usually collect citation metadata from several sources, and they need to resolve entity problems related to publication venue titles and author names, as we show in the following two examples.

4.1.1 Example 1

Given the four citations shown in Table 4.1, suppose we would like to find out which of the publication venue titles refer to the same publication venue. The problem arises because labels of publication venues such as journals, conferences, and workshops contain misspellings, spelling variants, and abbreviated forms, all of which make searching and retrieval more difficult. In the example of Table 4.1, the citations r_1 and r_3 refer to the same publication venue "Journal of Parallel and Distributed Computing", and the citations r_2 and r_4 refer to the same publication venue "IEEE International Conference on High Performance Computing".

Traditional approaches based on approximate string matching techniques [French et al., 2000] are ineffective to identify that citations r_2 and r_4 refer to the

Citation	Description
r_1	H. S. Paul, A. Gupta, and A. Sharma, "Finding a Suitable Checkpoint and Recovery Protocol for Distributed Applications", Journal of Parallel and Distributed Computing, 2006.
r_2	S. Karmakar and A. Gupta , "Fault-tolerant Topology Adaptation by Localized Distributed Protocol Switching", IEEE International Conference on High Performance Computing, 2007.
r_3	A. Gupta, D. Nelson, and H. Wang, "Efficient Embeddings of Ternary Trees into Hypercubes", J. Parallel Distr. Comp., 2003.
r_4	Z. Kamal, Ajay Gupta, L. Lilien, and A. Khokhar, "An Efficient MAP Classifier for Sensornets", HiPC Conference, 2006.

Table 4.1. Example of four citations (r_1 to r_4) containing each one its author names, work title, publication venue title, and year published, respectively.

same entity. Our proposal is to search for additional information on the Web to help in the disambiguation task. We submit each publication venue title as a query to a Web search engine. Then, we look for evidence in the answer set returned by each query to group citations referring to the same entity. For example, if two queries return common URLs in their answer sets, it can be an evidence that the corresponding publication venue titles refer to the same entity. Also, we can extract acronyms from the documents in the answer sets of the queries and use them as additional information. For example, if we find both the acronym "HiPC" and the title "IEEE International Conference on High Performance Computing" in documents returned by queries corresponding to citations r_2 and r_4, it can be an evidence that both publication venue titles refer to the same entity.

4.1.2 Example 2

Given the four citations shown in Table 4.1, suppose we would like to find out which of the author names refer to the same author. The most complicated case is to decide whether the authors whose surnames are "Gupta" are the same author or different authors. In this case, citations r_1 and r_2 are authored by "Arobinda Gupta", professor of the Indian Institute of Technology, and citations r_3 and r_4 are authored by "Ajay Gupta", professor of the Western Michigan University.

Author name ambiguity may occur due to the existence of multiple authors with the same name (polysemes) or different name variations for the same author (synonyms). More specifically, two related problems can be enumerated in this context: mixed citation and split citation [Lee, 2007; Lee et al., 2005]. The mixed citation problem occurs when citations of different authors are mixed, as multiple authors may have

the same or very similar name spellings. For example, the author "A. Gupta" may refer to "Arobinda Gupta" or "Ajay Gupta", two different people. The split citation problem occurs when citations of the same author are split under name variants. For example, the author "Ajay Gupta" may appear in multiple publications under different name abbreviations, such as "A. Gupta", "Ajay Gupta", or a misspelled name such as "Ajaj Gupta".

Traditional approaches are usually based on functions that measure the similarity among the attribute values, such as work title and publication venue title, or look for common coauthor names among citations to decide which citations belong to each one of the authors [Bhattacharya and Getoor, 2006; Cota et al., 2007; Han et al., 2005]. Such approaches could be ineffective for the citations in Table 4.1 since they are all from the same area and do not have common coauthors.

We submit data from each citation as a query to a Web search engine aiming at finding curricula vitae and Web pages containing publications of the authors. Then, we look for evidence in the answer set returned by each query to group citations referring to the same author. For example, we can look for common URLs or use domain-specific knowledge to identify single author documents (i.e., personal documents about only one person) in the answer sets of the queries, separating them from other types of documents such as the text of an article with its bibliographic references or a document with the list of articles from a journal or conference edition. Then, we group citations in the same document as being of the same author.

4.2 The WER Formulation

In the entity resolution problem, we are given a set of n records $\mathcal{R} = \{r_1, r_2, ..., r_n\}$, where each record r_i has attributes $r_i.A_1, r_i.A_2, ..., r_i.A_k$, which we refer to as domain-specific attributes. While each attribute might be a reference to a distinct entity of the world, they might all be used as references to the same entity. Thus, a reference to the entity to be disambiguated, which we refer to as primary entity pe, is also provided as input.

The problem of entity resolution consists of:

1. Determining the set $\mathcal{E} = \{e_1, e_2, ..., e_m\}$ of distinct real-world entities related to the primary entity pe, where m is the number of distinct entities, which is not provided as input;

2. Associating with each record r_i the corresponding (correct) entity $e_j \in \mathcal{E}$.

Figure 4.1 presents an example of some domain-specific attributes of a bibliographic citation record. In this example, the reference to the primary entity could be publication venue, or author, or article (all attributes). In the last case, we might want to find replicated citations that refer to the same article.

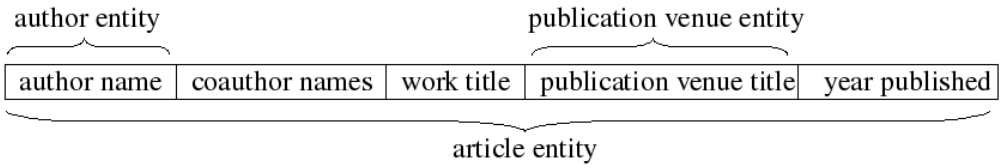

Figure 4.1. Example of a bibliographic citation record schema containing the attributes author name, coauthor names, work title, publication venue title, and year published, and its primary entities.

An authority file [Auld, 1982; French et al., 2000] is an index of authority records, where each record representing an entity is composed of a heading to be used as the entity label and a list of variant labels also used to refer to the entity, called cross references. The headings and the cross references can be used by a search system to answer queries related to an entity. For example, in an author name authority file, each record represents an author. The full name of the author could be selected as the heading while the other names could be maintained as alternative forms of referring to that author. A user can search for publications of an author using any name in the record.

Table 4.2 illustrates the two authority records generated from the Example 1 of Section 4.1. Considering that the primary entity is the publication venue, we obtain the entities "Journal of Parallel and Distributed Computing" and "IEEE International Conference on High Performance Computing".

$ar_1.A_{head}$	JPDC - Journal of Parallel and Distributed Computing
$ar_1.A_{cross}$	{"Journal of Parallel and Distributed Computing", "J. Parallel Distr. Comp."}
$ar_2.A_{head}$	HiPC - IEEE International Conference on High Performance Computing
$ar_2.A_{cross}$	{"IEEE International Conference on High Performance Computing", "HiPC Conference"}

Table 4.2. Two authority records, in this case relative to publication venues, that compose the authority file for the Example 1 of Section 4.1. The attribute $ar_i.A_{head}$ is the heading and the attribute $ar_i.A_{cross}$ contains the cross references of authority record i. The heading includes the acronym and the canonical title of the publication venue.

Given a set of records $\mathcal{R}$ and a primary entity *pe*, an authority file for the primary entity can be generated by solving the entity resolution problem. Since each cluster represents an entity, it is enough to select, for each one, a heading associated with *pe* to be the *canonical label* of the entity.

The WER approach uses the Web as a source of additional attributes for entity resolution and creation of authority files. Figure 4.2 illustrates the main steps of the process, described as follows:

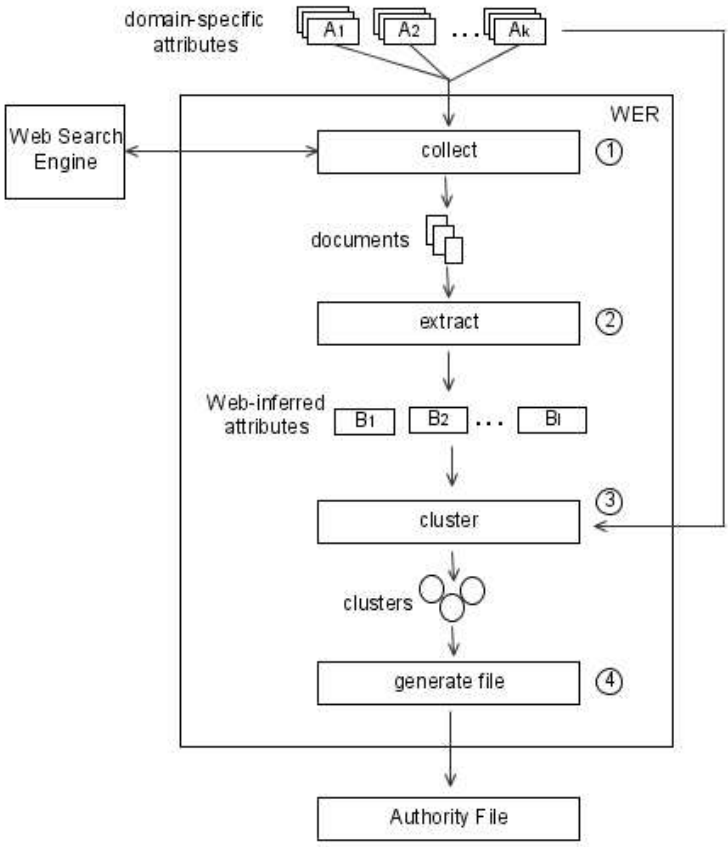

Figure 4.2. Four steps that compose the entity resolution algorithm in the WER Approach.

- Step 1 (Collecting information from the Web). Submit queries to a Web search engine and collect the top m documents in their answer sets. Each query is formed by a subset of the domain-specific attribute values of each record r_i in the input. The subset of attributes used to compose the query is selected by the user.

- Step 2 (Extracting information from the documents). Extract *Web-inferred* attributes for each record r_i by parsing the documents in the answer set of its

associated query. The Web-inferred attributes, referred to as B_1 to B_l in the figure, are, for example, URLs, titles, texts of the documents, names, and acronyms. In principle, they can all be used to refer to the entity. Section 4.3 presents more details of this step.

- Step 3 (Clustering records). Cluster the records using a combination of the original domain-specific and the Web-inferred attributes. Section 4.4 presents more details of this step.

- Step 4 (Generating an authority file). Generate an authority file selecting a name in each cluster to be the canonical name of the entity. Section 4.5 presents more details of this step.

4.3 Extracting Information from Documents

The documents returned by each query associated with record r_i are used to extract new attributes for r_i (Step 2), which we refer to as *Web-inferred* attributes. We extract three basic attributes from each document, which are the URL, the title, and the text. We also extract a heading for the primary entity, which is used to create the authority file. The heading is obtained from the text of the documents and it is computed as the most similar string to a reference to primary entity *pe* (using a similarity function such as the Jaccard similarity coefficient). Other attributes specific of the input data domain can also be extracted. An example is the acronym. They are common in domains that include names such as names of conferences, journals, and names of institutions (e.g.: universities and government departments).

In this thesis, we proposed a method to extract these five attributes. Once this step is concluded, each record r_i has attribute values given by a set of URLs, a set of titles, a set of texts, the most common heading, and the most common acronym.

Formally, let $\mathcal{D}_i$ be the set of the top m documents in answer set of the query associated with r_i. The set of all collected documents forms the Web document subcollection $\mathcal{D}$. From each document $d_j \in \mathcal{D}$, we extract values to the following attributes: URL ($d_j.B_{url}$), title ($d_j.B_{title}$), text content ($d_j.B_{text}$), and heading ($d_j.B_{head}$). We also consider additional attributes that are specific to the input data. For instance, an acronym ($d_j.B_{acro}$).

The extraction of the first three attributes is direct, as follows: (a) $d_j.B_{url}$ is the URL associated with the document, (b) $d_j.B_{title}$ is the title of the document, i.e., the text between the HTML tags <title> and </title>, and (c) $d_j.B_{text}$ is the text of the

document. The other two attributes, heading and acronym, require a more complex extraction procedure, as we now discuss.

4.3.1 Heading Extraction

To extract a heading associated with the document $d_j \in \mathcal{D}_i$, we first break the text of the document into phrases, where each phrase is formed by a sequence of words delimited by punctuation marks. To illustrate, consider that d_j contains the following text:

> *"ACM/IEEE Joint Conference on Digital Libraries.*
> *ACM/IEEE 2003 Joint Conference on Digital Libraries (JCDL 2003),*
> *27-31 May 2003, Houston, Texas, USA, Proceedings. IEEE Computer*
> *Society 2003, ..."*

From this document, we extract phrases such as: *"ACM/IEEE Joint Conference on Digital Libraries", ..., "Houston", "Texas", "USA", "Proceedings", and "IEEE Computer Society 2003"*

After that, we compute the similarity between each phrase and the value of the attribute of r_i related to the primary entity pe. The similarity is based on a function such as the Jaccard similarity coefficient. Then, we select the phrase with the highest similarity to be the value of the heading attribute $d_j.B_{head}$. For example, if the attribute value related to pe were "Joint Conference on Digital Libraries", the extracted heading would be "ACM/IEEE Joint Conference on Digital Libraries". We also try to expand simple abbreviations, like those that match the beginning of a word. If the expansion succeeds, we will consider the expanded form as a candidate for the heading.

When the value of the attribute of r_i related to pe is a short text (one or two words) and one of these words is identified as being an acronym by our algorithm, then instead of obtaining the most similar phrase as described above, we try to expand such acronym. The expansion algorithm match each letter, or a sequence of letters, in the acronym with the initial letters in the words of the phrases extracted from d_j. The phrase that matches with more letters of the acronym, more than a given threshold, is selected as being the heading attribute $d_j.B_{head}$. For well known entity acronyms, the search engine usually returns the acronym and the long entity name in the documents because the two forms are often together. For example, if the attribute value related to pe were "JCDL", the extracted heading for the previous example would be "ACM/IEEE Joint Conference on Digital Libraries", since "JCDL" is an acronym and its letters match with the initial letters in "Joint Conference Digital Libraries".

4.3.2 Acronym Extraction

From the text of d_j, we extract an acronym (if it exists) that matches the extracted heading. Candidates for acronym must have at least two uppercase letters. The expansion method tries to match initial letters, simple abbreviations and common conversions such as "2" and "to". This is also the approach adopted by Larkey et al. [2000]. An expansion coefficient is computed as the number of acronym symbols that were expanded. The acronym with the highest expansion coefficient, higher than a given threshold, is selected to represent the acronym associated with the document ($d_j.B_{acro}$).

Attribute values have been extracted from the documents, we associate them to record r_i, as follows.

4.3.3 URLs, Titles, and Texts

The URLs, titles, and texts associated with record r_i are obtained from documents $d_j \in \mathcal{D}_i$, as:

$$r_i.B_{url} = \bigcup_j d_j.B_{url}$$

$$r_i.B_{title} = \bigcup_j d_j.B_{title}$$

$$r_i.B_{text} = \bigcup_j d_j.B_{text}$$

4.3.4 Heading

The heading associated with record r_i is selected as being the most common value among the values of attributes $d_j.B_{head}$, $d_j \in \mathcal{D}_i$. We considered the most common value as being the heading with the highest sum of similarities in relation to the other headings extracted from $\mathcal{D}_i$, computed as follows. Let $simH(d_j.B_{head}, d_k.B_{head})$ be a function that returns the similarity between the strings $d_j.B_{head}$ and $d_k.B_{head}$. Then, we compute the sum of similarities among the strings $d_j.B_{head}$ and $d_k.B_{head}$, $\forall d_j, d_k \in \mathcal{D}_i$, as follows:

$$sumSim(d_j.B_{head}) = \sum_{k \neq j} simH(d_j.B_{head}, d_k.B_{head}) \tag{4.1}$$

Then, the heading Web-inferred attribute value for the record r_i is computed as:

$$r_i.B_{head} = d_j.B_{head}, \text{ such that } d_j.B_{head} \text{ has the maximum } sumSim(d_j.B_{head}) \tag{4.2}$$

We do not select the longest heading because it may introduce more noise, due to the strategy used to extract headings from phrases.

4.3.5 Acronym

The acronym associated with record r_i is selected as being the most common value among the values of the attributes $d_j.B_{acro}$, computed as follows. Let $aCount(d_j.B_{acro})$ be a function that counts the number of occurrences of each distinct and not null acronym $d_j.B_{acro}$, $d_j \in \mathcal{D}_i$. Then, the acronym Web-inferred attribute value for the record r_i is computed as:

$$r_i.B_{acro} = d_j.B_{acro}, \text{ such that } d_j.B_{acro} \text{ has the maximum } aCount(d_j.B_{acro}) \tag{4.3}$$

4.4 Clustering Records

To cluster records (Step 3), we use the domain-specific and the Web-inferred attributes. Each cluster is expected to represent a distinct real-world entity. For this, the clustering procedure computes the pairwise similarity of records r_i and r_j using a similarity function, $sim(r_i, r_j)$, based on a linear combination of ranking functions. Each ranking function associates a score with attribute values of the same attribute. Let $F_p(r_i, r_j)$ be the ranking function for the p^{th} attribute of records r_i and r_j, and let w_p be a weight associated with the p^{th} attribute. We define

$$\begin{cases} sim(r_i, r_j) = \sum_p w_p \times F_p(r_i, r_j) \\ \sum_p w_p = 1 \end{cases} \tag{4.4}$$

That is, we use a simple linear combination of ranking functions and determine the best weighted combination empirically. Our objective is to demonstrate that the inferred attributes obtained from the Web are useful to disambiguate entities, indepen-

dently of the strategy used to combine them.

The ranking functions for each one of the Web-inferred attributes are as follows.

$$F_{url}(r_i, r_j) = \begin{cases} 1.0 & \text{if } |r_i.B_{url} \cap r_j.B_{url}| \geq q, q > 0 \\ \frac{p}{q} & \text{if } |r_i.B_{url} \cap r_j.B_{url}| = p, 0 \leq p < q \end{cases}$$

where q is a parameter set empirically, corresponding to the required number of common URLs in the two sets.

$$F_{title}(r_i, r_j) = simT(r_i.B_{title}, r_j.B_{title})$$

Analogously to the $F_{title}(r_i, r_j)$, we define the functions $F_{text}(r_i, r_j)$ and $F_{head}(r_i, r_j)$ using the similarity functions $simX(r_i.B_{text}, r_j.B_{text})$ and $simH(r_i.B_{head}, r_j.B_{head})$, respectively, where $simT$, $simX$, and $simH$ are any string similarity function such as Jaccard similarity coefficient or cosine similarity.

Finally,

$$F_{acro}(r_i, r_j) = \begin{cases} 1.0 & \text{if } r_i.B_{acro} = r_j.B_{acro} \\ & \text{and } r_i.B_{acro} \text{ is not null} \\ 0.0 & \text{if } r_i.B_{acro} \neq r_j.B_{acro} \\ v & \text{if } r_i.B_{acro} \text{ is null} \\ & \text{and } r_j.B_{acro} \text{ is null}, 0.0 \leq v \leq 1.0 \end{cases}$$

where v is a parameter set empirically, used when there is no acronym in both records.

For clustering the records, we can use a clustering procedure based on any technique, such as the K-nearest-neighbor (KNN) or the hierarchical agglomerative clustering (HAC) [Croft et al., 2009].

4.5 Generating an Authority File

In Step 4, we generate an authority file. Since each cluster c_k generated in Step 3 represents a distinct real-world entity $e_k \in \mathcal{E}$ then it represents an authority record ar_k of the authority file, and we store a link to the records that it groups. Also, we store the set of attribute values of its records corresponding to the primary entity, which comprises the cross references attribute $ar_k.A_{cross}$. Moreover, we select its heading attribute $ar_k.A_{head}$, computed as follows.

Analogously to Eq. (4.1) and Eq. (4.2), we compute the heading attribute $ar_k.A_{head}$ for the authority record ar_k, where each $r_i.B_{head}$ is the heading of the record

$r_i \in c_k$, as follows:

$$sumSim(r_i.B_{head}) = \sum_{j \neq i} simH(r_i.B_{head}, r_j.B_{head})$$

$$ar_k.A_{head} = r_i.B_{head}, \text{ such that } r_i.B_{head} \text{ has the maximum } sumSim(r_i.B_{head})$$

If an acronym exists for any record in c_k, we compute the acronym attribute $ar_k.A_{acro}$ for the authority record ar_k and concatenates it to the heading, giving more information about the entity. Analogously to Eq. (4.3), it is computed as follows:

$$ar_k.A_{acro} = r_i.B_{acro}, \text{ such that } r_i.B_{acro} \text{ has the maximum } aCount(r_i.B_{acro})$$

4.6 The WER Framework

We implemented a generic and configurable framework for entity resolution and creation of authority files. The framework is composed of classes corresponding to the main steps of the WER algorithm. These classes contain basic operations for generic entity resolution and creation of authority files, and they can be extended for domain-specific applications.

The WER framework is configurable and the user can, for example, set the attributes of the records and select the attributes to compose the queries, the attribute corresponding to the primary entity, the type of document to be collected (text snippet or full text), the similarity function to be used to compare strings in the extraction of information and in the clustering procedure (e.g., edit distance, Jaccard coefficient, and cosine), the clustering procedure, and the weights to be used on the clustering similarity function.

The WER framework also implements some clustering metrics to evaluate experimental results, such as the k metric, pairwise F1, and cluster F1. It is usefull for simulations, permitting to model different strategies for entity resolution and to assess their results.

In the following sections, we present the details of the framework implementation, its user interface, and how to empirically determine the parameters of our approach.

4.6.1 Details of Implementation

We implemented the WER framework in the C++ language using object-oriented paradigm. Figure 4.3 presents the diagram of its main classes. The classes that keep information about the data generated in the four steps of our approach are described as follows:

- *Attribute*: stores the name and the value of an attribute of a record;
- *Record*: stores the list of attributes of a record;
- *Query*: stores the query submitted to a Web search engine related to a record. More than one record can generate the same query if the attributes selected to compose a query have the same values;
- *Document*: stores the URL, title, and the text of a document in the answer set of a query;
- *WebInfoDoc*: stores the heading and the acronym extracted from a document;
- *WebInfo*: stores the heading and the acronym selected from the set of documents related to a same record;
- *Cluster*: stores the heading and the acronym of a cluster, i.e., of an authority record. Each cluster also stores its set of records.

The other three classes (shadow classes in Figure 4.3) implement the operations of the four steps of our approach, described as follows:

- *Collector*: implements the Step 1. This class has operations to formulate queries to the input records, submit them to a Web search engine, and collect the top m documents in the answer set of each query. We implemented an interface to Google Search API [Google API, 2009]. To collect the full text of the documents, we used the libcurl C API[1];
- *Extractor*: implements the Step 2. This class has operations to extract a heading and an acronym from each document, and select a heading and an acronym to each record;

[1] http://curl.haxx.se/libcurl/c/

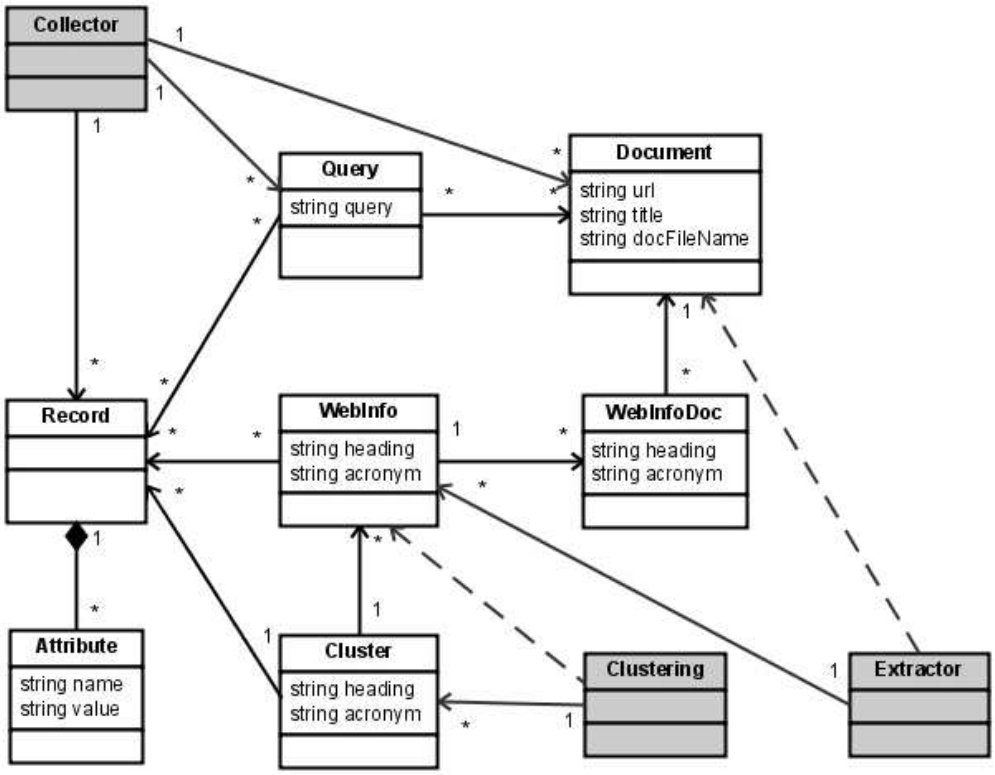

Figure 4.3. WER framework class diagram.

- *Clustering*: implements the Steps 3 and 4. This class has operations to cluster the input records and select a canonical name to each entity, generating an authority file. We implemented two clustering procedures: KNN and HAC (Section 3.3.2). We implemented the similarity functions for the Web-inferred attributes, described in Section 4.4, and the following string-based similarity functions: edit distance, Jaccard coefficient, edit distance combined with Jaccard coefficient, and cosine (Section 3.1). We also implemented the clustering evaluation metrics described in Section 3.3.2.2.

Solutions to new domains can be easily developed by specializing the framework code. In the object-oriented paradigm, this means create subclasses of more generic classes. For example, to implement our solution to author name resolution (Chapter 6), we created a subclass of the *WebInfoDoc* class, adding the attributes single author document, document author name, and IHF. We created a subclass of the *Extractor* class, adding operations to extract the previous attributes. We also created a subclass of the *Clustering* class, adding a new clustering procedure.

4.6.2 User Interface

The user interacts with the WER framework using a console interface. The input data and the parameter configurations are input by XML files, as we describe follows.

The input data file must contain a <*record*> tag for each record, and each record contains a tag for each of one of its attributes. The <*id*> tag is mandatory and its value must be unique among all records. It is used to refer to each record when the cluster quality is assessed. An example of an input data file is shown below:

```
<entities>
    <entity>
        <id>1</id>
        <description>HP Officejet J3680</description>
    </entity>
    <entity>
        <id>2</id>
        <description>HP J3680 printer</description>
    </entity>
</entities>
```

The parameter configurations are input by a XML file using the following tags:

- <qryAttributes>: attribute names to compose the queries;
- <stopwords>: list of stop words. Such stop words are removed when comparing strings in similarity functions;
- <docType>: type of document to be collected (S - text snippet, F - full text);
- <infoAttribute>: attribute corresponding to the primary entity;
- <acronymExceptions>: list of acronym exceptions;
- <simPhrase>: similarity function used in the heading extraction;
- <simFunctions>: similarity function parameters used in Eq. (4.4). For each attribute, its weight and similarity function are respectively informed. For the F_{url}, the last parameter is the value of q, and for F_{acro}, the value of v;
- <clusteringProcedure>: cluster procedure to be used;
- <clustThreshold>: clustering threshold to fuse cluster.

An example of a parameter configuration file is shown below:

```
<config>
    <qryAttributes>description</qryAttributes>
    <stopwords>of the in on and or with for from at</stopwords>
    <docType>S</docType>
    <infoAttribute>description</infoAttribute>
    <acronymExceptions>PDF DOC PS</acronymExceptions>
    <simPhrase>jaccard</simPhrase>
    <simFunctions>url 0.5 3 title 0.0 jaccard text 0.0 jaccard heading 0.5 jaccard
        acronym 0.0 0.7 description 0.0 editDistance</simFunctions>
    <clusteringProcedure>HAC</clusteringProcedure>
    <clustThreshold>0.65</clustThreshold>
</config>
```

The execution of the framework is composed of the following sequence of commands:

1. *authorityFile 1 collection.xml config.xml outFile pages*: submits queries to a Web search engine and collects the documents. *collection.xml* is the input data file name, *config.xml* is the parameter configuration file name, *outFile* is the directory where the output data is stored, and *pages* is the directory where the collected documents are stored.

2. *authorityFile 2 config.xml outFile*: extracts the Web-inferred attributes.

3. *authorityFile 3 config.xml outFile collection-man-clusters*: generates the clusters and the authority file. *collection-man-clusters* is a text file containing the manually generated clusters used to assess the quality of the automatically generated clusters. In this file, each line contains the identification of the input records in each cluster, delimited by a tab character. In the end of this step, a file containing the clusters and the evaluation metrics is generated.

4.6.3 How to Empirically Determine the Parameters

The WER approach requires many parameters to be empirically determined. These are:

- the weights and similarity functions of Eq. (4.4);
- the q parameter of the ranking function F_{url};

- the v parameter of the ranking function F_{acro}.

We suggest the following procedure to obtain them. Before generating an authority file for a test dataset, select another dataset from the same domain and fine tune the parameters in this dataset. Try several different combinations of parameters until the best result is obtained. Then, apply these parameters to the test dataset.

If the objective is to evaluate the method comparing it with other solutions, we can also use a k-fold cross validation procedure, similar to the method usually used in supervised approaches. In this case, the dataset is divided into k subsets, and the method is repeated k times. Each time, one of the k subsets is used as the test set and the other k-1 are put together to form a training set. The parameters are determined in the training set and applied to the test set. Then an average value for each metric is computed in the test set across all k runs.

Chapter 5

Experimental Evaluation

In this section, we present the experiments we conducted to evaluate the WER method. The WER method takes the domain-specific attribute values of the input records and uses the Web to extract inferred attribute values to them. Then, it applies ranking functions on domain-specific and Web-inferred attributes, and a clustering procedure to disambiguate them, generating an authority file.

Figure 5.1 presents the design of the experiments. The objective of the experiments is to demonstrate the effectiveness of the Web-inferred attributes B_i. We compare the WER strategy using only ranking functions applied to the attributes extracted from the Web against baselines that use the same clustering procedure, but the ranking functions are applied to domain-specific attributes A_i only.

In the experiments, we use the Jaccard coefficient, edit distance, cosine, and Jaccard coefficient combined with edit distance [French et al., 2000] metrics to compare attributes of the string type. For clustering data, we use the KNN and HAC clustering procedures as described in Section 3.3.2.1. The results were compared using the K, pairwise F1, and cluster F1 metrics described in Section 3.3.2.2.

We fine tuned the parameters and the contribution weights of the ranking functions to obtain the best results, for both the baselines and WER. We did not use a data sample to adjust such values, we adjusted them using all data in the test dataset, for both the baselines and WER.

To obtain information from the Web, we used the Google Search API [Google API, 2009] and collected the top 10 text snippets from each query.

The experiments were performed on the following three problems: printer description, publication venue title, and author name resolution. In the following sections, we report the experiments and results for each one of the three problems.

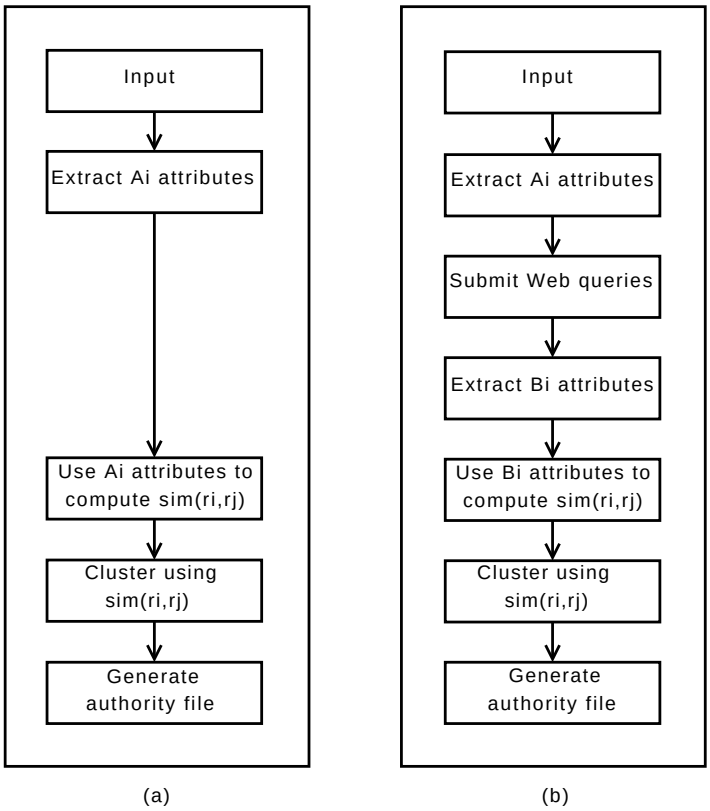

Figure 5.1. Design of the experiments. (a) Baseline—uses the domain-specific attributes A_i of the input records to compute $sim(r_i, r_j)$ (b) WER—uses the inferred attributes B_i extracted from the Web to compute $sim(r_i, r_j)$.

5.1 Experiments with Printer Description Resolution

In this section, we present the experiments to evaluate the WER method for the printer description resolution problem. It is a common problem faced by online product catalog services that need to consolidate product descriptions extracted from different Web pages into lists related to the same product.

To illustrate, consider the four printer descriptions presented in Table 5.1. The three first descriptions refer to the same printer, and the last description, despite very similar to the first, corresponds to a different printer.

HP Officejet J3680 All-in-One Printer, Fax, Scanner, Copier
HP CB071A#A2L J3680 Officejet Multifunction Printer
Hewlett Packard Officejet 3680 All-in-One Printer (CB071A)
HP Officejet J4580 All-in-One Printer, Fax, Scanner, Copier

Table 5.1. Examples of printer descriptions found in the test dataset.

5.1.1 Dataset

For testing, we used a real dataset of printer descriptions obtained by querying Google's Product Search[1]. First, we manually collected distinct printer descriptions from sites of several manufactures (HP, Epson, Canon and others). Then, we used these printer descriptions as queries to Google's Product Search and collected the titles of each document in the answer set. Finally, we manually selected, for each query, those distinct results that really corresponded to the queried printer. The resulting dataset is composed of 2169 distinct printer description strings, which we used for the experiments. These strings are distributed in 158 clusters, having an average of 13.7 strings per cluster, where the largest cluster has 27 strings, and the smallest, 2.

5.1.2 Results

For querying the Google search engine, we used the value of the description attribute. In the clustering step, the best combination of ranking functions used the pairs of URLs and headings, with weights 0.5 and 0.5, respectively. The parameter q in F_{url} function (the number of common URLs) was empirically set to 3.

Table 5.2 shows the results comparing WER with baselines, both using the KNN clustering procedure and four similarity functions: Jaccard, edit distance, Jaccard combined with edit distance, and cosine. For each comparison, we applied the corresponding similarity function to the printer description attribute for the baseline and the same similarity function to the heading attribute for WER, which also used the pair of URLs' attribute. Each cell shows the value for the K, pairwise (pF1), and cluster F1 (cF1) metrics. Averages were computed with 95% confidence interval. For each similarity function, the third line shows the gain of WER, which is statistically significant for all metrics.

WER was better than the baselines for all metrics. Comparing the similarity functions, we can observe that the results on Jaccard, cosine, and Jaccard with edit distance are statistically tied, although the Jaccard numbers are, in general, a little bit higher. Edit distance did not reach good results. The results on cluster F1 are low because the dataset has many large clusters containing many variant forms, making it difficult to obtain totally correct clusters.

Table 5.3 is similar to Table 5.2, except that the experiments for both, WER and baselines, used the HAC clustering procedure. The results also show that WER overtook the baselines on all metrics, and edit distance reached the worst numbers.

[1] http://www.google.com/products

Method	Function	K (%)	pF1 (%)	cF1 (%)
WER	jaccard	76.2 ± 1.8	63.9 ± 2.0	6.3 ± 1.0
baseline	jaccard	51.6 ± 2.1	28.4 ± 1.9	0.2 ± 0.2
	gain of WER	**47.7**	**125.0**	**3050.0**
WER	editDistance	70.9 ± 1.9	57.0 ± 2.1	2.9 ± 0.7
baseline	editDistance	34.4 ± 2.0	3.0 ± 0.7	0.0 ± 0.0
	gain of WER	**106.1**	**1800.0**	—
WER	jaccard-editDistance	73.2 ± 1.9	64.1 ± 1.7	4.8 ± 0.9
baseline	jaccard-editDistance	48.3 ± 2.1	20.8 ± 1.7	0.3 ± 0.2
	gain of WER	**51.6**	**208.2**	**1500.0**
WER	cosine	74.8 ± 1.8	64.5 ± 2.0	5.2 ± 0.9
baseline	cosine	42.6 ± 2.1	15.9 ± 1.5	0.0 ± 0.0
	gain of WER	**75.6**	**305.7**	—

Table 5.2. Comparison of WER with the baselines for four similarity functions using the KNN clustering procedure. Each cell shows the value for the K, the pairwise (pF1), and the cluster F1 (cF1) metrics.

Method	Function	K (%)	pF1 (%)	cF1 (%)
WER	jaccard	65.2 ± 2.0	55.8 ± 2.1	2.9 ± 0.7
baseline	jaccard	44.0 ± 2.1	28.9 ± 1.9	0.5 ± 0.3
	gain of WER	**48.2**	**93.1**	**480.0**
WER	editDistance	60.8 ± 2.1	50.9 ± 2.1	2.0 ± 0.6
baseline	editDistance	25.2 ± 1.8	9.5 ± 1.2	0.3 ± 0.2
	gain of WER	**141.3**	**435.8**	**566.7**
WER	jaccard-editDistance	65.1 ± 2.0	56.1 ± 2.1	2.4 ± 0.6
baseline	jaccard-editDistance	41.1 ± 2.1	24.2 ± 1.8	0.5 ± 0.3
	gain of WER	**58.4**	**131.8**	**380.0**
WER	cosine	65.4 ± 2.0	56.8 ± 2.1	3.2 ± 0.7
baseline	cosine	39.3 ± 2.1	22.5 ± 1.8	0.3 ± 0.2
	gain of WER	**66.4**	**152.4**	**966.7**

Table 5.3. Comparison of WER with the baselines for four similarity functions using the HAC clustering procedure. Each cell shows the value for the K, the pairwise (pF1), and the cluster F1 (cF1) metrics.

Comparing the results of both tables, we can see that the KNN clustering procedure produced better results than HAC.

Table 5.4 shows the results for WER using similarity functions applied separately to each one of the Web-inferred attributes. The objective is to evaluate the individual contribution of each Web-inferred attribute. All methods used the KNN clustering procedure and the comparison of the attributes used the Jaccard similarity function, except for the set of URLs. We do not use the acronym attribute because it is not common in printer descriptions.

As we can see in the table, the best results were obtained by using the set of URLs attribute values, indicating that if two references to the primary entity have at least three URLs in common in the answer set of their corresponding queries, they probably represent the same entity. Comparing the first and the last line in the table, we can see that the use of the heading attribute generated better results than its corresponding description attribute. Remember that the heading attribute value is obtained from the Web documents as a canonical label to the description attribute value. Moreover, the set of titles and the set of texts did not demonstrate to be good evidence to help in the disambiguation task when used as we proposed. However, they contain valuable information (the heading is extracted from each title and text, and it is a good evidence) and we need to investigate better their use.

Method	Attribute	K (%)	pF1 (%)	cF1 (%)
WER	heading	66.2 ± 2.0	56.0 ± 2.1	2.3 ± 0.6
WER	set of URLs	72.2 ± 1.9	60.5 ± 2.1	3.4 ± 0.8
WER	set of titles	49.3 ± 2.1	17.1 ± 1.6	0.4 ± 0.3
WER	set of texts	45.2 ± 2.1	5.3 ± 0.9	0.2 ± 0.2
baseline	description	51.6 ± 2.1	28.4 ± 1.9	0.2 ± 0.2

Table 5.4. Comparison of the contribution of each Web-inferred attribute used individually to compose the similarity function of WER. All methods used the KNN clustering procedure and the comparison of the attributes used the Jaccard similarity function, except for the set of URLs. Each cell shows the value for the K, the pairwise (pF1), and the cluster F1 (cF1) metrics.

We manually verified the clustering results for WER using the combination of the heading and set of URLs' attributes. We can observe that similar strings such as "HP Officejet J3680 All-in-One Printer, Fax, Scanner, Copier" and "HP Officejet J4580 All-in-One Printer, Fax, Scanner, Copier" were put together in the same cluster by the baseline and, correctly, in distinct clusters by WER. Also, strings with few words in common such as "HP Officejet J3680 All-in-One Printer, Fax, Scanner, Copier" and "HP CB071A#A2L J3680 Officejet Multifunction Printer", describing the same printer,

were put together in the same cluster by WER, and in distinct clusters by the baseline. In both cases, the URLs were important to disambiguate them.

Analyzing the WER failure cases, we found errors when disambiguating printers that have variations of the same model, such as "HP LaserJet P3005 Printer", "HP LaserJet P3005d Printer", and "HP LaserJet P3005dn Printer". They have similar strings and their queries return some URLs in common, which led WER to put them into the same cluster, even being distinct printers. Another failure case occured when the printer description contained several other abbreviated specifications besides its simple description. For instance, the printer "Lexmark x363dn Multifunction Printer" has as one of its descriptions the string "Lexmark 13b0501 x363dn mfp mono fb adf enet usb 1200dpi 128mb" that details other printer specifications embedded in its description. Such detailed descriptions produce few or no common URL with other descriptions of the same printer, which makes WER put them into distinct clusters. In both cases, the baseline also produced erroneous results.

Authority File

We evaluated the quality of the heading of the authority file produced by WER. We compared the heading extracted from the Web with the original printer description value. We considered good headings those containing the printer description without abbreviations or additional specifications such as speed, size, and memory.

We randomly selected 108 (5%) printer descriptions from the dataset, manually evaluated their headings, and classified the results in four cases: (1) the heading is good and the original description is not, (2) the heading and the original description are both good, (3) neither the heading nor the original description is good, and (4) the heading is not good and the original description is. We obtained the following results for the four cases, respectively: 40.7%, 28.7%, 23.2%, and 7.4% . Such results demonstrate that WER obtains a better description for 40.7% of the printers, and its error rate was of 7.4%. Note that 63.9% of the input descriptions are not good to be used as a printer canonical description, and using the headings obtained by WER, only 30.6% of the descriptions remain not good.

5.2 Experiments with Publication Venue Title Resolution

In this section, we present the experiments we carried out to evaluate the WER method for the publication venue title resolution problem. It is a common problem in digital libraries, which needs to identify which bibliographic citations refer to each one of the publication venues. A publication venue title might appear written in distinct forms in citations.

To illustrate, consider some of the distinct references to the VLDB conference presented in Table 5.5. Some references, such as those in the first and in the third lines, have no similarity when compared using traditional string matching techniques such as cosine or Jaccard similarity coefficient.

VLDB
VLDB Conference
International Conference on Very Large Data Bases
International Conference on Very Large Databases
Int. Conf. on Very Large Data Bases (VLDB)

Table 5.5. Examples of distinct references to the VLDB Conference found in the test dataset.

5.2.1 Dataset

For testing, we used a real dataset of citations (bibliographic records) obtained by querying Google Scholar[2]. It consists of Computer Science publications from four American universities (Stanford, MIT, Harvard and UC Berkeley). We chose Google Scholar because it stores real data automatically crawled from Web pages. We used the names of faculty members as queries to Google Scholar and collected the bibtex entries in the answer set of each query. We selected only articles/papers that included information on publication venue. Google Scholar does not identify the type of a publication venue, neither converts it into a canonical form. The resulting dataset is composed of 16689 citation records containing 8399 distinct publication venue title strings, which we used for the experiments.

We have 8399 distinct publication venue title strings in the test collection. It is time consuming to manually determine the correct cluster of each string. Besides, we need to determine the publication venue canonical title and acronym for each cluster.

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

Inspired by French et al. [2000], we measured the results based on sample test bases. After a preliminary execution of the system, we defined two sample test bases. For the first sample, we randomly selected 110 clusters. We named this sample as *sample-at-random*. For the second sample, which we named as *sample-of-the-largest*, we chose the 50 clusters with the largest support, i.e., the largest number of non-distinct citations in the input collection. For each publication venue of each sample, we manually determined its canonical title, acronym, and all strings in the input collection that represent citations to that publication venue.

The *sample-at-random* dataset has a total of 691 distinct strings, representing 8.2% of the input strings. This sample has on average 6.3 strings per cluster, the largest cluster has 81 strings, and there are 46 single clusters. The *sample-of-the-largest* dataset has a total of 1142 distinct strings, representing 13.6% of the input strings. This sample has on average 22.8 strings per cluster, the largest cluster has 81 strings, and there are only three single clusters.

For evaluation purposes, we assumed that the goal was to place the strings of each sample in their correct cluster, the remaining strings could be clustered in any way as long as they did not appear in one of the standard clusters of the sample. This task is more difficult than simply clustering the sample strings, more clusters must be examined, and the sample strings of interest may be intermingled with other strings.

5.2.2 Results

For querying the Google search engine, we used the value of the publication venue title attribute. In the clustering step, the best combination of ranking functions used the pairs of URLs, headings, and acronyms, with the weights 0.2, 0.5, and 0.3, respectively. The parameter q in F_{url} function (the number of common URLs) was empirically set to 3. The parameter v in F_{acro} function was also empirically set to 0.7.

Table 5.6 shows the results comparing the Jaccard similarity coefficient baseline with WER using the same similarity function to compare headings. The last line of each sample in the table presents the gains of WER, which are statistically significant at a 95% confidence level for the K and the pF1 metrics, on both the *sample-at-random* and *sample-of-the-largest* sets. The results on pairwise F1 are lower for the *sample-at-random* set because it has many single clusters, which are not counted on this metric. And the results on cluster F1 are lower for the *sample-of-the-largest* set because it has many large clusters containing many variant forms, making it more difficult to obtain totally correct clusters.

We also compared WER with baselines using edit distance, Jaccard combined

Method	*sample-at-random* (%)		
	K	pF1	cF1
WER	75.4 ± 3.2	37.5 ± 3.6	27.6 ± 3.3
baseline	67.2 ± 3.5	19.3 ± 2.9	25.1 ± 3.2
Gain of WER	**12.2**	**94.3**	10.0
Method	*sample-of-the-largest* (%)		
	K	pF1	cF1
WER	74.5 ± 2.5	58.0 ± 2.9	3.0 ± 1.0
baseline	65.1 ± 2.8	35.9 ± 2.8	1.7 ± 0.7
Gain of WER	**14.4**	**61.6**	76.5

Table 5.6. Comparison of WER against the baseline for the *sample-at-random* and *sample-of-the-largest* sets from the dataset. Each cell shows the value for the K, the pairwise (pF1), and the cluster F1 (cF1) metrics.

with edit distance, and cosine similarity functions. The results were similar to the Jaccard similarity coefficient, except for the edit distance similarity which reached lower numbers.

We manually verified the clustering results and observed that short and long strings, such as "*ACM SOSP*" and "*Symposium on Operating Systems Principles*", that have no similarity among them using traditional string matching techniques, were correctly put together in the same cluster by WER and in distinct clusters by the baseline.

WER failed in cases for which (1) the query was too generic, for instance, "Artificial Intelligence" as a journal title, (2) the query contained many spelling errors or abbreviated forms, and (3) two publication venues use a same acronym or a publication venue is known by different acronyms.

Authority File

We evaluated the quality of the heading of the authority file produced by WER. We compared the heading extracted from the Web with the original publication venue title value. We considered good headings those containing the full publication venue title without abbreviations, acronyms or additional information such as local of event, date, volume, and page number. Small variations in the titles were admitted, specially those such as inclusion of the sponsor and the use of words "Annual" or "International". For instance, "*IEEE Computer Society International Conference on Computer Vision and Pattern Recognition*" and "*IEEE Conference on Computer Vision and Pattern Recognition*" are both considered good headings. But "*CVPR*" and "*Comp. Vision Patt. Recog.*" are both considered not good headings.

We randomly selected 92 (5%) publication venue titles from the two sample datasets, manually evaluated their heading, and classified the results in four cases: (1) the heading is good and the original title is not, (2) the heading and the original title are both good, (3) neither the heading nor the original title is good, and (4) the heading is not good and the original title is. We obtained the following results for the four cases, respectively: 25.0%, 57.6%, 10.9%, and 6.5% . Such results demonstrate that WER obtains a better title for 25.0% of the publication venues, and its error rate was of 6.5%.

Publication venues are usually identified by their acronyms. Then, we also evaluated the quality of the acronym obtained by WER. We considered correct the cases in which the publication venue has an acronym and it was correctly obtained or the publication venue does not have an acronym and WER did not obtain it. In all other cases, it is considered incorrect. To compare an acronym obtained by WER with the original acronym, we considered that an original acronym exists if it can be easily identified in the original title by a human being. We evaluated the same 92 publication venue titles used to evaluate headings, and classified the results in four cases: (1) the acronym was correctly obtained and the original acronym does not exist, (2) the acronym was correctly obtained and the original acronym exists, (3) the acronym was not obtained and the the original really does not exist, and (4) the acronym was incorrectly obtained or it was not obtained and the original acronym exists. We obtained the following results for the four cases, respectively: 51.1%, 14.1%, 23.9%, and 10.9%. Such results demonstrate that WER produced a correct acronym for 89.1% of the cases, and its error rate was of 10.9%.

5.3 Experiments with Author Name Resolution

In this section, we present the experiments to evaluate the WER method for the author name resolution problem. It is a common problem in digital libraries, which need to identify which bibliographic citations refer to each one of the authors. We cannot solve this problem using only the author name due to the existence of multiple authors with the same name (polysemes) or different name variations for the same author (synonyms).

To illustrate, consider the three citations presented in table 5.7. The three citations have an author called "A. Gupta". However, "A. Gupta" in citations c_1 and c_2 refers to "Arobinda Gupta", professor of the Indian Institute of Technology, and "A. Gupta" in citation c_3 refers to "Ajay Gupta", professor of the Western Michigan

University.

Citation	Description
c_1	H. S. Paul, A. Gupta, and A. Sharma, "Finding a Suitable Checkpoint and Recovery Protocol for Distributed Applications", Journal of Parallel and Distributed Computing, 2006.
c_2	S. Karmakar and A. Gupta , "Fault-tolerant Topology Adaptation by Localized Distributed Protocol Switching", IEEE International Conference on High Performance Computing, 2007.
c_3	A. Gupta, D. Nelson, and H. Wang, "Efficient Embeddings of Ternary Trees into Hypercubes", J. Parallel Distr. Comp., 2003.

Table 5.7. Example of three citations (c_1 to c_3) containing each one its author names, work title, publication venue title, and year published, respectively.

5.3.1 Dataset

For testing WER for the author name resolution problem, we performed experiments on the dataset proposed by Han et al. [2005]. Table 5.8 presents information about this dataset. It is composed of 8442 citation records, with 480 distinct authors, divided into 14 ambiguous groups. Each author has at least 2 citations. Each record contains the following attributes: ambiguous author name, coauthor names, work title, and publication venue title.

5.3.2 Results

For querying the Google search engine, we used the value of the ambiguous author name and the work title attributes. The idea is that, if two distinct queries return common URLs, probably such documents contain publications of the same author. In the clustering step, the best combination of ranking functions used the pairs of URLs only. The parameter q in F_{url} function (the number of common URLs) was empirically set to 2.

Table 5.9 shows the results comparing the Jaccard similarity coefficient baseline with WER using the K, pairwise F1 (pF1), and cluster F1 (cF1) metrics. For the baseline, each citation is represented by the string formed by concatenating the coauthor name, work title, and publication venue title attribute values. The last line in the table presents the gains of WER, which are statistically significant at a 95% confidence level for the K and pF1 metrics. Although the baseline overtook WER on cluster F1, the gain is not statistically significant, and the results are low for both methods.

Name	Number of Authors	Number of Citations
A. Gupta	26	577
A. Kumar	14	244
C. Chen	61	800
D. Johnson	15	368
J. Lee	100	1417
J. Martin	16	112
J. Robinson	12	171
J. Smith	31	927
K. Tanaka	10	280
M. Brown	13	153
M. Jones	13	259
M. Miller	12	412
S. Lee	86	1458
Y. Chen	71	1264
Total	480	8,442

Table 5.8. Information about the dataset proposed by Han et al. [2005], divided into 14 ambiguous groups. Each column lists, respectively, the name label of the ambiguous group, the number of authors each name label corresponds to, and the total number of citations in the correspondent ambiguous group.

Method	K	pF1	cF1
WER	72.6 ± 5.0	66.4 ± 9.3	3.8 ± 1.7
baseline	55.9 ± 3.6	37.3 ± 7.7	5.1 ± 2.2
Gain of WER	**28.1**	**78.0**	-25.5

Table 5.9. Comparison of WER against the baseline. Each cell shows the mean value among the 14 groups of authors for the K, the pairwise (pF1), and the cluster F1 (cF1) metrics.

We also compared WER with baselines using edit distance, Jaccard combined with edit distance, and cosine similarity functions. The results were similar to the Jaccard similarity coefficient, except for the edit distance similarity which reached lower numbers.

We manually verified the clustering results and identified that many citations of distinct authors were put together in the same cluster. This happens because many documents in the query answer sets contain citations of distinct authors. Such documents are, for example, text of articles or pages of articles in digital libraries that also contain their bibliographic references, or list of articles from a journal or conference edition. Probably, we could obtain better results if we identified the documents in the query answer sets that contained only publications of a single author, such as his/her curriculum vitae. This motivated us to implement a specific solution for author name resolution that we describe in Chapter 6.

Authority File

We evaluated the quality of the heading of the authority file produced by WER. We compared the heading extracted from the Web with the original author name. We considered good headings those ones containing an extend or full name of the author. For example, "Arobinda Gupta" is an extend name for the abbreviated name "A. Gupta".

We randomly selected 84 (1%) author name references from the dataset, manually evaluated their headings based on the DBLP author page or on the curriculum vitae of the authors, and classified the results in four cases: (1) the heading is a correct expanded name and the original name is abbreviated, (2) the heading and the original name are both expanded and correct, (3) the heading and the original name are both abbreviated and correct, and (4) the heading is incorrect. We obtained the following results for the four cases, respectively: 17.9%, 9.5%, 67.9%, and 4.7% . Such results demonstrate that WER obtained a correct expanded name for 27.4% of the authors, and its error rate was of 4.7%. The percentage of expanded names is low because author names usually appear abbreviated in citations.

Chapter 6

A Specific Solution for Author Name Resolution

The proposal for extracting information from the Web to help in the entity disambiguation process may be specialized for some applications. In this case, we can use specific knowledge about the application to apply heuristics that may improve the results. We apply this idea to develop a specific solution for author name resolution [Pereira et al., 2009b]. Several other methods have been proposed in the literature for disambiguating author names in bibliographic citations [Cota et al., 2007; Han et al., 2004, 2005; Huang et al., 2006; Kang et al., 2009; Lee et al., 2005; On et al., 2005; Song et al., 2007; Tan et al., 2006].

As we presented in the experiments on author name resolution in Section 5.3, we submitted queries to a Web search engine aiming at finding documents containing publications of authors. Then, we trusted the fact that, if two or more queries return documents in common, they may indicate that such documents contain publications of a single author. In fact, it may be true for many documents, but there are also many others that can contain publications of distinct authors, such as text of articles or pages of articles in digital libraries that also contain their bibliographic references, or list of articles from a journal or conference edition. We can take advantage of knowing such specific application properties and make use of some heuristics to determine which ones are the documents that will effectively help in the author name disambiguation process.

The specific solution is similar to the generic one in the sense that we submit queries to a Web search engine using the attribute values of the input citations, we collect and extract information from the documents in the answer sets, and we apply a clustering procedure to group citations of the same author. The main difference is

that we use specific knowledge about the application to classify the documents in order of importance to the disambiguation process. The specific solution is based on: (1) identifying single author documents, i.e., documents containing citations of a single person such as a curriculum vitae, (2) weighting documents generating a ranking of their importance in the disambiguation process, and (3) applying a specific clustering procedure using single author documents and the document ranking. Such steps are summarized as follows.

To identify a single author document, we look for an author name in the page title of a document, in its URL, and in the beginning of its text. In the latter case, we try to identify curricula vitae, where the author name appears in the beginning of the document along with words such as "curriculum vitae", "resume", "name", "address", "mail", and "phone". If an author name appears alone in one of these three places and the same author name appears in all citations found in the document, then the document is considered a single author document.

The importance of each document in the disambiguation process is quantified using the *inverse host frequency* (IHF) factor proposed by Tan et al. [2006]. IHF quantifies the relative rarity of an Internet host among a suitable corpus of Web documents, akin to the Inverse Document Frequency metric [Baeza-Yates and Ribeiro-Neto, 1999] used in information retrieval. The idea is that pages from rare Web sites have more importance in the disambiguation process than from common ones, such as pages from digital libraries.

Finally, we use a hierarchical agglomerative clustering procedure that groups citations found in single author documents with high IHF. Such documents are probably curricula vitae or home pages containing citations of a single person. For those single author documents with low IHF, we groups only those citations that are also found together in other documents. Such documents are probably pages of authors in digital libraries, and we look for more evidence to group their citations, since digital libraries may contain errors.

The WER method deals at the same time with the split citation and mixed citation problems [Lee, 2007; Lee et al., 2005]. By putting into the same cluster citations that are together in the same document identified as being of an ambiguous author, independently of spelling names of the authors in the citations, we are dealing with the split citation problem. On the other hand, by separating in distinct clusters authors whose citations do not appear together in any document, even having the same spelling names, we are dealing with the mixed citation problem.

In the next sections we discuss the main steps of the WER method in details. In order to describe the method, we consider the first author name of each citation as the

ambiguous one. The method can be generalized by applying it to a specific citation as many times as needed, each time selecting one of its coauthors as the ambiguous one. In this work, we refer to the ambiguous author name simply as the author name of the citation, and the other $n-1$ author names as its coauthor names.

6.1 Obtaining Information from the Web

In Step 1, we receive as input a list of citations containing ambiguous author names, submit queries to a Web search engine, and collect the top 10 documents from the answer set of each query. The goal is to generate a Web document sub-collection containing the publications of the ambiguous authors. In particular, the target of the queries are curricula vitae and Web pages containing publications of these authors. For this, we suggest two options for queries. To illustrate the options, we will use the following working citation example: *Using Web Information for Creating Publication Venue Authority Files*, authored by Denilson A. Pereira (which is considered the ambiguous author) and whose coauthors are Berthier Ribeiro-Neto, Nivio Ziviani and Alberto H. F. Laender. Note that the initials of the author name and stop words are removed, except when they are inside a quoted title.

- **Query option 1:** unquoted author name followed by the word "publications", followed by unquoted work title (e.g.: Denilson Pereira publications Using Web Information Creating Publication Venue Authority Files).

- **Query option 2:** unquoted author name followed by quoted work title (e.g.: Denilson Pereira "Using Web Information for Creating Publication Venue Authority Files").

Queries using quoted work title (as in query option 2) retrieve only documents that really contain the phrase formed by the work title, although they may not contain the target citation. The problem with this type of query is that input citations with misspelling errors in the work title generate queries with no result. Queries using unquoted work title (as in query options 1) may correctly retrieve citations with misspelling errors, although they also tend to retrieve documents that do not contain the target citations.

6.2 Extracting Information from Documents

In Step 2, the Web document sub-collection formed by querying a Web search engine is used as a source of additional information for disambiguating author names. We keep no relationship between the citations used in a query and the documents in its answer set, thus the queries are used only to create a Web document sub-collection to be used in a clustering procedure. In order to be sure that a citation is really contained in a document, we look for it in text of the document.

In this step, we look for citations in the documents from the Web document sub-collection, identify single author documents, and weight the importance of each document in the process of disambiguation, as follows.

6.2.1 Looking for Citations in Documents

To identify which documents contain each citation c_i in the input list of citations C, we look for them in the Web document sub-collection D. We implemented a local search engine using CLucene [CLucene, 2008] to index and query D. For each citation c_i, we use its list of author names $A_i = \{a_{1i}, a_{2i}, ..., a_{ni}\}$, its work title t_i, and its publication venue title v_i to submit queries to the local search engine. Three distinct strategies were used to create queries depending on the size (the number of words without stop words) of the work title t_i.

If t_i comprises at least five words, the query is formed by the quoted title t_i followed by an AND clause that includes the first four author names, or all author names if $|A_i| < 4$. For each author name a_{ji}, an OR clause is formed by the first name and the last surname. Considering the citation example in Section 6.1, the query generated would be: "Using Web Information for Creating Publication Venue Authority Files" AND (Denilson OR Pereira) AND (Berthier OR Ribeiro-Neto) AND (Nivio OR Ziviani) AND (Alberto OR Laender). If the query returns no document, a sequence of other similar queries is produced until one of them returns at least one document or no more queries can be produced. In these other queries, the aim is to permit errors of one word in the work title t_i, i.e., the searched work title may have a missing word, or one additional word, or one different word, or a displacement of one word. We do this by generating queries without one of the words in work title t_i and using the *slop factor* of Lucene [Gospodnetić and Hatcher, 2005].

If t_i comprises three or four words, the query is formed by the quoted title t_i followed by the first four author names, similarly to the previously described query, except that it does not permit one word error.

If t_i comprises only one or two words, the query is formed by the quoted title t_i followed by the first four author names, similarly to the previously described query, also not permitting one word error, followed by an AND statement using the first four words from the publication venue title v_i. Short titles may mislead the results, specially generic titles such as *Genetic Programming*, then we add the publication venue title to this type of query.

From the result sets of the queries, we store, for each document $d_i \in D$, the list of citations it contains. This information will be used to create the clusters of authors.

6.2.2 Identifying Single Author Documents

Simply identifying the citations in each document is not enough to disambiguate authors. It is important to identify whether a document contains citations of a single person, which we call *single author document*. In the Web document sub-collection, we can find several types of non single author documents, for example, the text of an article with its bibliographic references, the page of an article from a digital library, such as The ACM Digital Library or CiteSeer, that also contains its bibliographic references, a page with the list of articles from a journal or conference edition, and pages of a collection of articles for a specific purpose.

We compare author names using a function that receives as input two strings, corresponding to two author names, compares them, and returns true if the two names are compatibles. The function compares each of the individual fragments that compose the names (including initials), ignoring the order in which these fragments appear in the input strings. If all fragments of one of the input strings match, the names are considered compatibles. For example, "Pereira D. A." and "D. Pereira" are compatibles, and both are compatible with "Denilson A. Pereira" and with "David A. Pereira", but the last two names are not compatible. This function is similar to that presented by Cota et al. [2007], but we do not use an edit distance function. In this work, we refer to this function as *CompareAuthors*.

To identify single author documents, we look for an author name in the page title (the text between the HTML tags <title> and </title>) of a document, in its URL, and in the beginning of its text. If an author name appears alone in one of these three places, the same author name appears in all citations found in the document, and the document does not contain the text of an article, then the document is a single author document. Formally, let S_j be the set of all citations found in a document d_j and let N_j be the set of all author names in citations S_j. We identify d_j as a single author document and $a_k \in N_j$ as its author if *(a)* a_k is the only one author name in

the page title, or in the URL, or in the beginning of the text of d_j, in this order, *(b)* every citation $c_i \in S_j$ contains author a_k, and *(c)* document d_j does not contain two of the words "Abstract", "Introduction", "References", and "Citations" as a single word in a line of its text. Comparison of author names uses the *CompareAuthors* function. To verify whether a_k is the only one author name in the page title, or in the URL, or in the beginning of the text of document d_j, we proceed as follows:

- **Page Title**: if d_j is an HTML page then extract the text t_j between tags <title> and </title>. For each author name $a_k \in N_j$, compare, using *CompareAuthors*, strings a_k and t_j. If this comparison returns true for only one a_k, then a_k is the only one author name in the page title of d_j;

- **URL**: For each author name $a_k \in N_j$, verify whether one of the fragments that composes a_k is a substring of the URL of d_j. If this verification returns true for only one a_k, then a_k is the only one author name in the URL of d_j;

- **Text**: Extract the text t_j of the first 20 lines of d_j, limited to 1,000 characters. Look, in text t_j, for words such as "curriculum vitae", "resume", "name", "address", "mail", and "phone". If one of these words is found, then reduce text t_j to 100 characters before and after the found word. For each author name $a_k \in N_j$, compares, using *CompareAuthors*, strings a_k and t_j. If this comparison returns true for only one a_k, then a_k is the only one author name in the beginning of the text of document d_j.

6.2.3 Weighting Documents

In the process of disambiguation, we consider citations in the same document as being of the same author. The problem is that a specific citation may be found in more than one document. Then, it is necessary to weight the importance of each document in the process of disambiguation. For this objective, we use the Inverse Host Frequency (IHF), proposed by Tan et al. [2006]. IHF quantifies the relative rarity of an Internet host among a suitable corpus of Web documents, akin to the Inverse Document Frequency measure [Baeza-Yates and Ribeiro-Neto, 1999] used in information retrieval. If a hostname h has frequency $f(h)$, then its *inverse host frequency* is computed as:

$$IHF(h) = \log_2 \frac{max_h f(h) + 1}{f(h) + 1} + 1$$

The IHF idea is that pages from rare Web sites have more importance in the process of disambiguation than common Web sites, such as pages from digital libraries.

We quantify the importance of a document d_i in a Web document sub-collection D as the IHF value of the host of d_i, i.e., $IHF(d_i) = IHF(h(d_i))$. The Web corpus to compute IHF are all documents in the Web document sub-collection D that contain at least one citation.

6.3 Clustering Citations

Using information extracted in Step 2, we cluster the input citations in such way that each cluster should represent a distinct author (person). We perform a hierarchical agglomerative clustering (HAC) procedure, grouping citations in the same document together in a bottom-up fashion, given by Algorithm 3.

We start putting each citation into a single cluster (lines 1 to 5). After that, it fuses clusters considering the citations found in the same document. First, it takes only single author documents and sort them in descending order by IHF (lines 6 to 7). Using this order, for each document whose IHF is higher than a threshold (minIHF, line 10) and the author name contained in the document is compatible with the author name of some cluster that contains at least one citation found in that document, then we fuses all clusters that contain citations found in that document (lines 9 to 12). The idea is that the minIHF separates personal sites from sites of digital libraries. Thus, if the author of the document is one of the ambiguous authors, then all citations in the document are of the same author and are grouped together. Compatibility between names are checked by the *CompareAuthors* function. To select the author name of the fused clusters, we take the name with more fragments of size greater than one (not considering initials) among the fused clusters.

A single author document may be a document from a coauthor of an ambiguous author. Based on a general heuristic that considers that very rarely two authors with compatible names that share a coauthor in common would be two different people in the real world [Cota et al., 2007; Laender et al., 2008b], we consider coauthor's document as a weaker evidence to fuse clusters. In this case, we fuse clusters with citations in the coauthor's document if it finds at least one more document from a distinct host that also contains citations from the clusters to be fused (lines 14 to 15). To fuse two clusters, it is necessary to find two distinct pairs of citations, each pair formed by citations in distinct clusters, in the same document. If a cluster has only one citation, then it is enough only one pair of citations. Similarly, documents with IHF less than or equal to minIHF need another evidence to have clusters fused. Documents from digital libraries are good sources to disambiguate author names but they may contain errors,

Algorithm 3 WER - Clustering function

Require: Citations $C = \{c_1, c_2, ..., c_n\}$;

Require: Web document sub-collection $D = \{d_1, d_2, ..., d_m\}$ with its information extracted in Step 2;

Ensure: $CL = \{(\{cc_1\}, p_1), (\{cc_2\}, p_2), ..., (\{cc_p\}, p_p)\}$ where CL is the set of generated clusters, cc_i is the set of citations in cluster cl_i, and p_i is the author name of cl_i;

1: **for each** citation c_i **do**
2: create new cluster cl_i;
3: $cc_i \leftarrow c_i$;
4: $p_i \leftarrow$ author name of c_i;
5: **end for**
6: $S \leftarrow$ single author documents in D;
7: sort documents in S in descending order by IHF;
8: **for each** $d_j \in S$ in descending order **do**
9: $A \leftarrow$ clusters that contain at least one citation found in d_j;
10: **if** $IHF(d_j) > minIHF$ **and** author name of d_j is compatible with the author name of some cluster in A **then**
11: fuse all clusters in A;
12: select an author name for the fused clusters;
13: **else**
14: fuse clusters in A which have compatible author names and two pairs of distinct citations found in a document $d_k \neq d_j$ whose $host(d_k) \neq host(d_j)$;
15: select an author name for each fused set of clusters;
16: **end if**
17: **end for**
18: $R \leftarrow$ **not** single author documents in D;
19: sort documents in R in descending order by IHF;
20: **for each** $d_j \in R$ in descending order **do**
21: $A \leftarrow$ single clusters that contain citation found in d_j;
22: fuse clusters in A which have compatible author names;
23: select author name for each fused set of clusters;
24: **end for**

then we look for evidences in more than one site to fuse clusters.

The last part of the algorithm (lines 18 to 24) uses non single author documents as a weak evidence to fuse single clusters. Most of these documents contain bibliographic references of an article, and we use a heuristic that considers that an article usually contains citations to more than one article of the same person, specially self-citations, i.e., citations to articles from one of the coauthors. But being this a weak evidence, we use it to fuse only single clusters with compatible author names.

6.4 Experimental Evaluation

To evaluate the specific solution for author name resolution, we used the dataset described in Section 5.3.1 and compared the results against the generic solution and three other baselines, as we describe in what follows.

To compute IHF, we used all documents from the query answer sets of the 14 author groups that contained at least one citation. We obtained the threshold ($minIHF$) of Algorithm 3 using all these documents, and used the same fixed and normalized value of 0.6 for each author group.

6.4.1 Results for the WER Method

Table 6.1 and Table 6.2 show the results for the WER method for the 14 groups of ambiguous authors. The results vary by group of authors but, on the average, they are similar. A disadvantage of queries using quoted work titles (query option 2) is that a simple misspelling error in the work title does not retrieve the correct citation, although this type of query tends to be better than those using unquoted work titles (query option 1) if there are few erroneous work titles. Note that a citation not retrieved by a quoted work title may be retrieved in a document returned by a query of another citation of the same author, since in its Step 2 we can find citations with errors of one word in their work titles.

For the 8,442 citations in the test dataset, queries using unquoted work titles (query option 1) retrieved 41,151 distinct URLs. Queries using quoted work titles (query option 2) retrieved 20% less distinct URLs, what means less processing effort to extract information from their documents.

We also evaluated the accuracy of the strategy for the identification of single author documents. We randomly selected a sample of 100 documents that contain at least two citations and manually checked if they are single author documents. We

Name	K	pF1	cF1
A. Gupta	0.90	0.88	0.21
A. Kumar	0.86	0.88	0.20
C. Chen	0.71	0.47	0.18
D. Johnson	0.87	0.93	0.13
J. Lee	0.69	0.62	0.09
J. Martin	0.89	0.90	0.29
J. Robinson	0.84	0.86	0.24
J. Smith	0.72	0.62	0.06
K. Tanaka	0.87	0.93	0.04
M. Brown	0.76	0.73	0.16
M. Jones	0.78	0.81	0.03
M. Miller	0.79	0.79	0.07
S. Lee	0.76	0.69	0.20
Y. Chen	0.72	0.58	0.09
Mean	**0.80**	**0.76**	**0.14**
Std Dev	0.07	0.15	0.08

Table 6.1. Results for the WER method using query option 1 (unquoted author name + "publications" + unquoted work title). "Std Dev" - Standard Deviation.

Name	K	pF1	cF1
A. Gupta	0.82	0.78	0.13
A. Kumar	0.82	0.85	0.16
C. Chen	0.74	0.56	0.16
D. Johnson	0.81	0.80	0.19
J. Lee	0.71	0.63	0.18
J. Martin	0.88	0.89	0.29
J. Robinson	0.94	0.96	0.53
J. Smith	0.79	0.77	0.10
K. Tanaka	0.82	0.86	0.14
M. Brown	0.82	0.82	0.13
M. Jones	0.81	0.84	0.08
M. Miller	0.68	0.61	0.13
S. Lee	0.78	0.81	0.23
Y. Chen	0.75	0.65	0.13
Mean	**0.80**	**0.77**	**0.18**
Std Dev	0.07	0.12	0.11

Table 6.2. Results for the WER method using query option 2 (unquoted author name + quoted work title). "Std Dev" - Standard Deviation.

correctly identified 90% of these documents (with a confidence interval of ± 5.9% at 95% confidence).

6.4.2 Baselines

WER was compared with three baselines: the support vector machines based method (SVM) [Han et al., 2004], the k-way spectral clustering based method (KWAY) [Han et al., 2005], and the Web-based method using hierarchical agglomerative clustering (Web-HAC) [Tan et al., 2006]. KWAY and SVM are classical unsupervised and supervised learning methods, respectively, and Web-HAC, like WER, is an unsupervised method that uses the Web as an external source. Note that the comparison is not with supervised and unsupervised methods per se, but with the strategies adopted by the original works [Han et al., 2004, 2005; Tan et al., 2006]. KWAY and SVM use only basic citation metadata, whereas Web-HAC and WER also use data from the Web. In addition, KWAY and Web-HAC use privileged information about the number of clusters.

In SVM, each author name (individual person) is associated with an author class and the classifier for that class is trained. Each citation is represented by a feature vector with its attribute values as features and the frequency of its attribute values as the feature weight. In this work, each citation is represented by a feature vector, with each author and coauthor name, and each term in the work and publication venue titles corresponding to a feature, and TF-IDF (Term Frequency - Inverse Document Frequency) being the feature weight. We run SVM 10 times, each time using 50% of the data, selected at random, for training and the others 50% for test. For the experiments, we used an RBF kernel function and the LibSVM package [LibSVM, 2009]. We also used the LibSVM Grid program to find the best parameter values (cost and gamma) for every training step, i.e., before applying the method to each test data.

KWAY does not use any training data, although it considers that the number of correct clusters is previously known. This disambiguation method uses a K-way spectral clustering, a graph model where each citation is represented by a vertex of an undirected graph, and the edge weight between two vertices represents the similarity between the corresponding citations. The disambiguation problem consists in splitting the graph so that citations that are more similar to each other belong to the same cluster. In [Han et al., 2005], each citation is represented by a feature vector where each feature corresponds to an element of a given instance of one of its attributes. In this work, the vectors were constructed with the same features and feature weights used by the SVM method. We used the SpectralLIB package [SpectralLIB, 2009] in its

implementation.

Web-HAC is a hierarchical agglomerative clustering method that computes the pairwise similarity of citations using the cosine similarity to derive a number of clusters previously known. For each citation, it submits a query to a Web search engine using the corresponding work quoted title to obtain a set of relevant URLs. Each citation is then represented by a feature vector, whose features are the relevant URLs weighted by their IHFs. In the implementation of Web-HAC, we used hostnames to calculate IHF and the single link scheme for hierarchical agglomerative clustering. Queries were submitted using the Google Search API, and the top 10 URLs of each answer set were taken.

6.4.3 Comparison with Baselines

Table 6.3 shows the comparison of the specific solution (WER specific) against the generic solution (WER generic) and the unsupervised methods Web-HAC and KWAY. We use the query option 1 (unquoted author name + "publications" + unquoted work title). As we can see, WER (specific) outperforms the two baselines Web-HAC and KWAY under all metrics, and is statistically tied with WER (generic) under K and pF1 metrics. The generic solution outperforms KWAY under all metrics and it is statistically tied with Web-HAC.

Method	K (%)	pF1 (%)	cF1 (%)
WER (specific)	0.80 ± 0.04	0.76 ± 0.08	0.14 ± 0.05
WER (generic)	0.72 ± 0.05	0.66 ± 0.09	0.04 ± 0.02
Web-HAC	0.63 ± 0.06	0.46 ± 0.14	0.05 ± 0.03
KWAY	0.50 ± 0.04	0.36 ± 0.05	0.01 ± 0.01

Table 6.3. Comparison of the specific solution (WER specific) against the generic solution (WER generic) and the unsupervised methods Web-HAC and KWAY. Each cell shows the mean value among the 14 groups of authors for the K, pairwise (pF1), and cluster F1 (cF1) metrics, and its 95% confidence interval.

As the generic solution, Web-HAC does not analyze the content of the documents, thus a non single author document with high IHF, for instance, an article with its bibliographic references, can be used as evidence to fuse clusters of distinct authors. Analyzing the content of the documents, the specific solution outperforms Web-HAC. Note that WER does not use the privileged information about the number of clusters to be generated. Even using this information, Web-HAC does not reach the correct number of clusters because there were not enough similarities among citations to fuse clusters. Analyzing the results of Web-HAC, we also note the following problems: (i)

since it uses only quoted titles to formulated the queries, a simple misspelling error produces incorrect results or no results at all, which makes the corresponding feature vectors meaningless and (ii) generic work titles such as *Parallel Processing* deviate the result, retrieving pages that do not contain the citation (because of this problem, WER includes the author name in its queries, which adds more context to retrieve citations, and after that it looks for citations permitting small errors). Like Web-HAC, WER (specific) also uses a hierarchical agglomerative clustering strategy, however, there is no threshold to stop fusing clusters.

Table 6.4 shows the comparison of the specific solution (WER specific) against the generic solution (WER generic) and the supervised learning based method SVM. The WER method uses the query option 1 (unquoted author name + "publications" + unquoted work title). For a fair comparison, we run WER (specific and generic) 10 times, each time using the same 50% of the data used by SVM for testing. The presented results are an average of these 10 runs. As we can see, the specific solution outperforms the generic solution and both are statistically tied with SVM. However, WER (specific and generic) does not use any training data whereas SVM does, which is per se a major advantage from a practical viewpoint. Note, however, that SVM uses only basic citation metadata as in [Han et al., 2004], whereas WER uses additional Web information that provides stronger evidence for the disambiguation task.

Method	K (%)	pF1 (%)	cF1 (%)
WER (specific)	0.82 ± 0.04	0.80 ± 0.07	0.29 ± 0.07
WER (generic)	0.71 ± 0.05	0.62 ± 0.08	0.17 ± 0.04
SVM	0.77 ± 0.04	0.66 ± 0.08	0.20 ± 0.05

Table 6.4. Comparison of the specific solution (WER specific) against the generic solution (WER generic) and the supervised learning based method (SVM). Each cell shows the mean value among the 14 groups of authors for the K, pairwise (pF1), and cluster F1 (cF1) metrics, and its 95% confidence interval.

6.4.4 Improvements

So far, we have presented the essence of the WER method. In this section, we present additional strategies that can be used to improve its effectiveness.

6.4.4.1 Another query option

Besides the two query options presented in Section 6.1, we also evaluated a third query option to obtain information from the Web, which is described as follows.

Query option 3: unquoted author name followed by three unquoted work titles. To find three work titles to compose a query, we create clusters of authors with compatible names and having at least one common coauthor with compatible names, exactly as done in [Cota et al., 2007]. This is a simple and practical strategy which generates pure, but fragmented clusters. We use these clusters only to generate queries, after which they are not used for any other purpose. For each author cluster, we generate n queries, if $n > 3$, where n is the number of citations in the cluster, using a sliding and circular window. Each query contains the unquoted cluster author name followed by three unquoted work titles, as follows, considering a the author name. $Query_1$: $a\ t_1\ t_2\ t_3$, $query_2$: $a\ t_2\ t_3\ t_4$, ..., $query_{n-2}$: $a\ t_{n-2}\ t_{n-1}\ t_n$, $query_{n-1}$: $a\ t_{n-1}\ t_n\ t_1$, and $query_n$: $a\ t_n\ t_1\ t_2$. If $n \leq 3$, it generates only one query containing the unquoted cluster author name followed by n unquoted work titles.

Table 6.5 shows the results for the WER method using query option 3. We can see that the results are slightly better than those of the other query options. This kind of query tends to retrieve only documents that contain the three work titles in the same page, although the efficacy of the result depends on the quality of the generated clusters. Query option 3 retrieved 14.4% less distinct URLs than query option 2 and 31.5% less than query option 1, what means less processing effort to extract information from the documents.

Name	ACP	AAP	K	pF1	cF1
A. Gupta	0.89	0.90	0.89	0.88	0.19
A. Kumar	0.98	0.76	0.86	0.87	0.21
C. Chen	0.93	0.60	0.75	0.57	0.16
D. Johnson	0.86	0.74	0.80	0.82	0.06
J. Lee	0.87	0.54	0.69	0.61	0.11
J. Martin	0.97	0.84	0.90	0.90	0.29
J. Robinson	1.00	0.81	0.90	0.92	0.28
J. Smith	0.77	0.81	0.79	0.76	0.09
K. Tanaka	0.99	0.75	0.87	0.93	0.04
M. Brown	0.97	0.69	0.82	0.80	0.18
M. Jones	0.99	0.65	0.80	0.84	0.09
M. Miller	0.99	0.63	0.79	0.79	0.10
S. Lee	0.87	0.79	0.83	0.87	0.19
Y. Chen	0.84	0.65	0.74	0.73	0.08
Mean	**0.92**	**0.73**	**0.82**	**0.81**	**0.15**
Std Dev	0.07	0.10	0.06	0.11	0.08

Table 6.5. Results for the WER method using query option 3 (unquoted author name + three unquoted work titles). "Std Dev" - Standard Deviation.

We also tested the combination of the three query options, i.e, we used the union

of the document sets returned by the three query options, but this combination did not improve the results.

6.4.4.2 Combining additional information

Many works have employed basic citation metadata such as coauthor names, work titles and publication venue titles as features to disambiguate author names. Fusion of clusters in hierarchical clustering methods can be performed in phases, and in each phase distinct evidence can be used to fuse clusters [Cota et al., 2007; Laender et al., 2008b]. For this process to work well, each phase needs to deliver very pure clusters (i.e., clusters with few mixed citations) to the next phase.

As we can see by the ACP and AAP metrics, respectively, in Table 6.5, the WER method produces very pure but very fragmented (i.e., citations from the same author spread in different clusters) sets of clusters. The fragmention happens mainly due to the fact that citations not found in the Web or found only in a document with low IHF factor are put into single clusters by WER. We can take advantage of this fact by using WER in a first phase of a hierarchical clustering process and applying other pieces of evidence in a second phase to continue fusing clusters.

We experimented by using coauthor information to fuse clusters after applying the WER method. Table 6.6 shows the results. We can see that the fragmentation has improved, increasing K, although the purity has decreased. The strategy to fuse clusters using coauthor information works as follows. In a hierarchical manner, the existing clusters with compatible author names are pairwise compared. Two clusters are fused if they have at least 10% of the total of their coauthor names compatible, restricted to a minimum equal to 1 and to maximum equal to the number of coathors in the smaller cluster. This process continues until no more clusters can be fused.

6.4.5 Analysis of Failure Cases

In this section, we show some cases for which the method has failed, discuss the reasons for these failures, and illustrate them with some examples. A major case of failure of the WER method is due to citations not found on the Web (about 7% of the citations in the test dataset), since WER generates single clusters with these citations. The main reasons for this are (i) misspelling errors in more than one word in work titles and (ii) existence of incorrect citations in the test dataset.

As described in Section 6.2, the WER method is able to find citations with at most one incorrect word in the work title. The test dataset we are using was generated in 2005 by Han et al. [2005] by collecting data from DBLP. As DBLP has been

Name	ACP	AAP	K	pF1	cF1
A. Gupta	0.88	0.96	0.92	0.90	0.36
A. Kumar	0.98	0.83	0.90	0.91	0.36
C. Chen	0.84	0.72	0.78	0.64	0.24
D. Johnson	0.83	0.81	0.82	0.83	0.16
J. Lee	0.68	0.74	0.71	0.59	0.22
J. Martin	0.97	0.93	0.95	0.93	0.71
J. Robinson	1.00	0.90	0.95	0.96	0.69
J. Smith	0.75	0.88	0.82	0.78	0.12
K. Tanaka	0.95	0.85	0.90	0.94	0.18
M. Brown	0.94	0.79	0.87	0.88	0.29
M. Jones	0.98	0.81	0.89	0.92	0.17
M. Miller	0.99	0.84	0.91	0.90	0.32
S. Lee	0.79	0.85	0.82	0.85	0.30
Y. Chen	0.77	0.76	0.76	0.75	0.17
Mean	**0.88**	**0.83**	**0.86**	**0.84**	**0.31**
Std Dev	0.11	0.07	0.07	0.11	0.18

Table 6.6. Results using coauthor information to fuse clusters after applying the WER method. "Std Dev" - Standard Deviation.

continually updated, some citations can no more be found in DBLP pages and it seems they no longer exist in any other site on the Web (we checked several cases manually). For instance, the citation "Akhil Kumar, Zigzag Path-based Matching Algorithms for High Availability and Load Balancing, manuscript" is not listed in Akhil Kumar's DBLP page (http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/k/Kumar:Akhil.html). Furthermore, ambiguous author names of some citations in the test dataset are incorrect. For instance, the citation "Cowley AW Jr, Tonellato PJ., The Physiome: Tool for Physiological Genomics, Proc. of On Designing the Physiome Project, 1997" is incorrectly attributed to C. Chen. As WER uses ambiguous author names to query a Web search engine and look for citations in the answer set, if an author name is incorrect its citation is not found.

Another case of failure of the WER method happens due to the fact that some citations are found only in pages of digital libraries, for instance, the citation "Edward Rothburg and Anoop Gupta, An Efficient Block-oriented Approach to Parallel Sparse Cholesky Factorization, Supercomputing, 1993". In cases like this, if there is no more than one page from different digital libraries containing a same citation, WER generates fragmented clusters. Furthermore, digital libraries contain errors that may cause incorrect fusion of clusters by WER. For instance, the citation "J. R. Smith, S.-F. Chang, Quadtree Segmentation for Texture-Based Image Query, ACM Multimedia, 1994" is authored by John

R. Smith according to the URL http://www.ctr.columbia.edu/∼jrsmith/pubs-to-2003.htm, but in DBLP this same citation is listed in the page of Jonathan M. Smith according to the URL http://www.informatik.uni-trier.de/∼ley/db/indices/a-tree/s/Smith:Jonathan_M=.html.

Finally, we present an example of failure on the identification of single author documents. The document in http://www.cs.ucsb.edu/∼teo/publications/OTHERSOCJR.html whose author name is "Teofilo F. Gonzalez", coauthor of "D. Johnson", does not have his name appearing in the page title, URL, or in the beginning of the text of the document. Thus, WER was not able to identify it as a single author document.

Chapter 7

Conclusions and Future Work

In this chapter, we summarize our accomplishments and present final conclusions. We also suggest future directions to improve the general method and to develop specific solutions to be applied in digital libraries.

7.1 Conclusions

We have presented a novel method that uses information extracted from the Web for entity resolution and creation of authority files. We proposed a sequence of steps to submit queries to a Web search engine, extract information from documents in the answer sets, and create clusters of records based on the extracted infomation. We take advantage of sophisticated matching procedures and of the large repositories implemented by the modern Web search engines. The method is generic and can be applied to different types of entity. We implemented a configurable framework containing basic operations for generic entity resolution and creation of authority files that can be extended for domain-specific applications. It is usefull for simulations, permitting to model different strategies for entity resolution and assess their results.

We evaluated our method using real datasets collected from the Web. These datasets reflect typical entity resolution problems faced by Web catalog services and digital libraries that collect data from Web sites. The results indicate large gains in the quality of entity resolution when applied to three distinct datasets, achieving gains in the pairwise F1 metric up to 125% when compared to the selected baselines. In the creation of authority files, the method is able of expanding abbreviated forms and obtaining canonical names of good quality, and it can also produce acronyms automatically. We also presented a specialized solution for author name resolution that obtains better results when compared to unsupervised baseline methods, and it

is statistically tied with a supervised learning based method, which requires human labeling and expensive training time.

The Web is a valuable source of information to disambiguate references to primary entities. The challenge is to find the right documents and extract information from them. We proposed to submit queries to a Web search engine to find documents and some heuristics to extract inferred attributes from them. However, the experiments show that there is no specific combination of Web-inferred attributes that is good for all datasets. The set of URLs and the heading are good evidences, specially when combined between them and with other attributes. The acronym is valuable to improve the results but they are not usually found in all datasets. The set of titles and the set of texts are not good evidence to be used as we proposed, but they were used indirectly to obtain the heading.

We proposed to solve the problems of entity resolution and authority file at the same time. Both problems have in common the task of disambiguating entity references, and after this step, in the authority file problem, an authority name is chosen to represent a canonical name for each entity. The heading Web-inferred attribute is used to generate the canonical name of each entity. Such attribute, in general, represents a better name than the original one of the primary entities. We used it to disambiguate entity references and to generate canonical names at the same time.

We also evaluated the influences of the clustering procedure in the results. We implemented two of them: KNN and HAC with single link. In all experiments, the best results were obtained using KNN. Both clustering procedures suffer from a chaining effect in which two clusters may be forced together due to the fact that a single record in a cluster is close to a single record in the other cluster, even though the others may be very distant. This suggests that other clustering strategies should be investigated to be more conclusive.

The use of Web-inferred attributes such as URLs and acronyms are interesting because they provide additional evidence to disambiguate entity references besides their labels. They are useful to separate in distinct clusters references that have very similar labels but represent distinct entities, and to put into the same cluster references containing totally distinct labels. This does not work well in methods that use only string matching techniques such as edit distance or cosine to compare records. Although our method works well for such cases, it also fails in other ones. We analyzed some cases of failure in the methods and observed that some queries can bring documents that contain no information about a primary entity, specially for generic or very ambiguous strings, for instance, the string "Computer" as a journal title. Besides, it does not work well for references to primary entities containing very specific strings such as those

that contain several attribute values embedded in a single attribute. For example, a printer description that also contains other data such as speed, memory, and size. The challenge is to identify whether a document or a Web-inferred attribute value is good or not. The method also fails when the information contained in a record can not be found by querying a Web search engine.

Although generic solutions may be applied to many different datasets, in some cases, specific knowledge about an application can be used to create specific solutions that obtain better results. In this work, we proposed a specific solution for author name resolution. We proposed and evaluated some heuristics to identify documents containing citations of a single author and a clustering procedure that groups citations in the same document together. The framework was implemented in such way that specific solutions may be created by extending its code.

7.2 Future Work

Following, we suggest how to continue this work improving the generic method. We also suggest to develop an application to disambiguate bibliographic citations and take statistics about publications of an institution.

7.2.1 Improving the Generic Solution

There are many interesting directions we can follow to improve the generic solution. We suggest the following:

- to investigate other forms to use the Web-inferred attributes set of titles and set of texts. In the experiments, we did not obtain good results using them, but they contain valuable information that must be better analyzed. An option would be to verify which references to primary entities each text contains and correlate them;
- to study the use of supervised methods to combine weights and ranking functions. Supervised methods such as SVM [Vapnik, 1995] or Random Forests [Breiman, 2001] may be used to combine weights and ranking functions using domain-specific and Web-inferred attributes in the clustering procedure, similar to the works of Bilenko et al. [2003] and Cohen et al. [2003b]. Or to combine them using genetic programming such as in [de Carvalho et al., 2008];
- to study strategies to verify whether a document in the result set of a query is valuable or not to the disambiguation task, similar to the specific solution for

author name resolution. For example, to disambiguate replicated hotels, a page of a travel agency is not a good document because it contains several distinct hotels. However, to disambiguate products that belong to the same manufacturer, a page of a manufacturer containing its products is a good document;

- to evaluate the impact of the Web search engine in the results. We only used the Google Search Engine in all experiments. It is important to experiment other Web search engines to evaluate how much Google influenced our results;

- to implement the solution using the paradigm of cloud computing. The idea is to offer a service that provides data about each one of the steps of the framework, permitting a user to develop specific solutions for her applications. For instance, she can ask for the set of URLs or an acronym of a reference to a primary entity.

7.2.2 Disambiguating Bibliographic Citations

From the experience of developing specific solutions for creating publication venue authority files [Pereira et al., 2008] and for disambiguating author names [Pereira et al., 2009b], we intend to improve such solutions and to develop new ones in order to disambiguate bibliographic citations. This is an important problem faced by digital libraries. The intention is to develop better strategies to assess the research quality in Brazilian institutions. In [Laender et al., 2008a], they identify the profile of the scientific publications of the main Computer Science departments in Brazil and worldwide, generating statistical data about them, for example, venue titles and venue types (journal, conference, workshop) in which each institution publishes more articles. Their study is based on data from DBLP [Ley, 2002]. In order to increase the coverage of this dataset, adding other data sources, it is important to reliably identify individuals, institutions, and publication venues associated with each publication. Our work may be usefull in this disambiguation task.

In order to develop an application to identify the publication profile of an academic institution, we need to execute the following steps:

1. To collect the researchers' publications from the Web. We can use a Web search engine to collect documents containing bibliographic citations and then to identify single author documents as we did in [Pereira et al., 2009b].

2. To extract the citations of a document and identify each field of a citation. We can use tools such as in [da Silva et al., 2007] and [Cortez et al., 2007].

3. To identify replicated citations. We can use our solution for entity resolution.

4. To disambiguate entities such as authors and publication venues. We can also use our solution for entity resolution.

We can improve the solution to disambiguate author names extracting from the Web documents information such as e-mail, address, affiliation, and author's full names from single author documents. This information is normally available in the curricula vitae of authors. We can also combine strategies to disambiguate author names and publication venue titles at the same time. Moreover, we can create and maintain a large publication venue authority file in computer science, developing strategies to insert new entries into it, keeping its quality.

Bibliography

Auld, L. (1982). Authority control: An eight-year review. *Library Resources & Technical Services*, 26:319--330.

Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern information retrieval*. Addison-Wesley-Longman.

Baeza-Yates, R. A. and Navarro, G. (1998). Fast approximate string matching in a dictionary. In *Proceedings of the 2nd South American Symposium on String Processing and Information Retrieval*, pages 14--22, Santa Cruz de La Sierra, Bolivia.

Bekkerman, R. and McCallum, A. (2005). Disambiguating web appearances of people in a social network. In *Proceedings of the 14th World Wide Web Conference*, pages 463--470, Chiba, Japan.

Benjelloun, O., Garcia-Molina, H., Kawai, H., Larson, T., Menestrina, D., and Thavisomboon, S. (2007). D-swoosh: A family of algorithms for generic, distributed entity resolution. In *Proceedings of the 27th International Conference on Distributed Computing Systems*, pages 37–46, Toronto, Canada. IEEE Computer Society.

Benjelloun, O., Garcia-Molina, H., Menestrina, D., Su, Q., Whang, S. E., and Widom, J. (2008). Swoosh: A generic approach to entity resolution. *The VLDB Journal*. Published online.

Berkhin, P. (2002). Survey of clustering data mining techniques. Technical report, Accrue Software, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.18.3739.

Bhattacharya, I. and Getoor, L. (2006). A latent dirichlet model for unsupervised entity resolution. In *Proceedings of the 6th SIAM Conference on Data Mining*, Bethesda, USA. SIAM.

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

Bilenko, M., Kamath, B., and Mooney, R. J. (2006). Adaptive blocking: Learning to scale up record linkage. In *Proceedings of the 6th IEEE International Conference on Data Mining*, pages 87–96, Hong Kong. IEEE Computer Society.

Bilenko, M. and Mooney, R. (2003). Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 39--48, Washington, USA. ACM, New York, NY, USA.

Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., and Fienberg, S. (2003). Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5):16--23.

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 92--100, Madison, USA. ACM, New York, NY, USA.

Bollegala, D., Honma, T., Matsuo, Y., and Ishizuka, M. (2008a). Mining for personal name aliases on the web. In *Proceedings of the 17th World Wide Web Conference*, pages 1107--1108, Beijing, China.

Bollegala, D., Matsuo, Y., and Ishizuka, M. (2008b). Disambiguating personal names on the web using automatically extracted key phrases. In *Proceedings of the 17th European Conference on Artificial Intelligence*, pages 553--557, Riva del Garda, Italy.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5--32.

Carvalho, J. C. P. and da Silva, A. S. (2003). Finding similar identities among objects from multiple web sources. In *Proceedings of the 5th ACM International Workshop on Web Information and Data Management*, pages 90--93, New Orleans, USA. ACM.

Chakrabarti, S. (2003). *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufmann.

Chaudhuri, S., Ganti, V., and Motwani, R. (2005). Robust identification of fuzzy duplicates. In *Proceedings of the 21st International Conference on Data Engineering*, pages 865--876, Washington, USA. IEEE Computer Society.

CLucene (2008). CLucene search engine. http://clucene.wiki.sourceforge.net. Accessed in November, 2008.

Cohen, W. W. (1998). Integration of heterogeneous databases without common domains using queries based on textual similarity. In *Proceedings of the 1998 ACM*

SIGMOD International Conference on Management of Data, pages 201--212, Seattle, USA. ACM, New York, NY, USA.

Cohen, W. W., Ravikumar, P., and Fienberg, S. E. (2003a). A comparison of string distance metrics for matching names and records. In *Proceedings of the 9th ACM SIGKDD Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, pages 13--18, Washington, USA. ACM, New York, NY, USA.

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

Cortez, E., da Silva, A. S., Gonçalves, M. A., Mesquita, F., and de Moura, E. S. (2007). FLUX-CiM: Flexible unsupervised extraction of citation metadata. In *Proceedings of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 215--224, Vancouver, Canada.

Cota, R. G., Gonçalves, M. A., and Laender, A. H. F. (2007). A heuristic hierarchical clustering method for author name disambiguation in digital libraries. In *Proceedings of the 22nd Brazilian Symposium on Databases*, pages 20--34, João Pessoa, Brazil.

Croft, W. B., Metzler, D., and Strohman, T. (2009). *Search Engines: Information Retrieval in Practice*. Addison Wesley.

da Silva, A. S., Barbosa, D., Cavalcanti, J. M. B., and Sevalho, M. A. S. (2007). Labeling data extracted from the web. In *OTM Conferences*, volume 4803 of *Lecture Notes in Computer Science*, pages 1099–1116. Springer.

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171--176.

Davis, P. T., Elson, D. K., and Klavans, J. L. (2003). Methods for precise named entity matching in digital collections. In *Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 125--127, Houston, USA.

de Carvalho, M. G., Laender, A. H. F., Gonçalves, M. A., and da Silva, A. S. (2008). Replica identification using genetic programming. In *Proceedings of the 23rd ACM symposium on Applied computing*, pages 1801--1806, Fortaleza, Brazil. ACM.

DiLauro, T., Choudhury, G. S., Patton, M., Warner, J. W., and Brown, E. W. (2001). Automated name authority control and enhanced searching in the Levy Collection. *D-Lib Magazine*, 7(4).

Dong, X., Halevy, A., and Madhavan, J. (2005). Reference reconciliation in complex information spaces. In *Proceedings of the 25th ACM SIGMOD International Conference on Management of Data*, pages 85--96, Baltimore, USA. ACM, New York, NY, USA.

Elmacioglu, E., Kan, M.-Y., Lee, D., and Zhang, Y. (2007). Web based linkage. In *Proceedings of the 9th Annual ACM International Workshop on Web Information and Data Management*, pages 121–128, Lisbon, Portugal. ACM.

Elmagarmid, A. K., Ipeirotis, P. G., and Verykios, V. S. (2007). Duplicate record detection: A survey. *IEEE Transaction on Knowledge and Data Engineering*, 19(1):1--16.

Febrl (2009). Freely extensible biomedical record linkage. http://sourceforge.net/projects/febrl. Accessed in July, 2009.

Fellegi, I. P. and Sunter, A. B. (1969). A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183--1210.

French, J. C., Powell, A. L., and Schulman, E. (2000). Using clustering strategies for creating authority files. *Journal of the American Society for Information Science*, 51(8):774--786.

Giles, C. L., Bollacker, K. D., and Lawrence, S. (1998). Citeseer: an automatic citation indexing system. In *Proceedings of the 3rd ACM conference on Digital libraries*, pages 89--98, Pittsburgh, USA. ACM New York, NY, USA.

Google API (2009). Google ajax search api. http://code.google.com/apis/ajaxsearch. Accessed in August, 2009.

Gospodnetić, O. and Hatcher, E. (2005). *Lucene in Action: A Guide to the Java Search Engine*. Manning Publications Co.

Han, H., Giles, C. L., Zha, H., Li, C., and Tsioutsiouliklis, K. (2004). 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.

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

Han, J. and Kamber, M. (2000). *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers.

Hernández, M. A. and Stolfo, S. J. (1995). The merge/purge problem for large databases. *ACM SIGMOD Record*, 24(2):127--138.

Hong, Y., On, B.-W., and Lee, D. (2004). System support for name authority control problem in digital libraries: OpenDBLP approach. In *Proceedings of the 8th European Conference on Digital Libraries*, volume 3232/2004 of *Lecture Notes in Computer Science*, pages 134--144, Bath, UK. Springer Berlin / Heidelberg.

Huang, J., Ertekin, S., and Giles, C. L. (2006). Efficient name disambiguation for large-scale databases. In *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 536--544, Berlin, Germany. Springer.

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: A review. *ACM Computing Surveys*, 31(3):264--323.

Jaro, M. A. (1989). Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414--420.

Jaro, M. A. (1995). Probabilistic linkage of large public health data files. *Statistics in Medicine*, 14(5–7):491--498.

Kalashnikov, D. V., Nuray-Turan, R., and Mehrotra, S. (2008). Towards breaking the quality curse.: A web-querying approach to web people search. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 27--34, Singapore, Singapore. ACM, New York, NY, USA.

Kan, M.-Y. and Tan, Y. F. (2008). Record matching in digital library metadata. *Communications of the ACM*, 51(2):91--94.

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

Karypis, G., Han, E.-H., and Kumar, V. (1999). Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68--75.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.

Laender, A. H. F., de Lucena, C. J. P., Maldonado, J. C., de Souza e Silva, E., and Ziviani, N. (2008a). Assessing the research and education quality of the top brazilian computer science graduate programs. *ACM SIGCSE Bulletin*, 4(2):135--145.

Laender, A. H. F., Gonçalves, M. A., Cota, R. G., Ferreira, A. A., Santos, R. L. T., and Silva, A. J. C. (2008b). Keeping a digital library clean: New solutions to old problems. In *Proceedings of the 8th ACM Symposium on Document Engineering*, pages 257--262, São Paulo, Brazil.

Lapidot, I. (2002). Self-organizing-maps with BIC for speaker clustering. IDIAP research report 02-60, IDIAP Research Institute, Martigny, Switzerland.

Larkey, L. S., Ogilvie, P., Price, M. A., and Tamilio, B. (2000). Acrophile: An automated acronym extractor and server. In *Proceedings of the 5th ACM International Conference on Digital Libraries*, pages 205–214, San Antonio, USA.

Lawrence, S., Giles, C. L., and Bollacker, K. (1999a). Digital libraries and autonomous citation indexing. *IEEE Computer*, 32(6):67--71.

Lawrence, S., Giles, C. L., and Bollacker, K. D. (1999b). Autonomous citation matching. In *Proceedings of the 3rd Annual Conference on Autonomous Agents*, pages 392--393, Seattle, USA. ACM, New York, NY, USA.

Lee, D. (2007). Practical maintenance of evolving metadata for digital preservation: Algorithmic solution and system support. *International Journal on Digital Libraries*, 6(4):313--326.

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

Lee, D., On, B.-W., Kang, J., and Park, S. (2005). Effective and scalable solutions for mixed and split citation problems in digital libraries. In *Proceedings of the 2nd Int'l Workshop on Information Quality in Information Systems*, pages 69--76, Baltimore, USA.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707--710.

Ley, M. (2002). The DBLP computer science bibliography: Evolution, research issues, perspectives. In *Proceedings of the 9th International Symposium on String Processing and Information Retrieval*, volume 2476 of *Lecture Notes in Computer Science*, pages 1--10, Lisbon, Portugal. Springer.

Library of Congress (2009). Us library of congress name authority file. http://www.loc.gov/marc/authority/index.html. Accessed in July, 2009.

LibSVM (2009). LibSVM package. www.csie.ntu.edu.tw/∼cjlin/libsvm. Accessed in January, 2009.

McCallum, A., Nigam, K., and Ungar, L. H. (2000). Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 169--178, Boston, USA. ACM, New York, NY, USA.

Michell, T. M. (1996). *Machine Learning*. McGraw Hill, New York, NY.

Monge, A. and Elkan, C. (1996). The field matching problem: Algorithms and applications. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 267--270, Portland, USA.

Montgomery, D. C. and Runger, G. C. (1999). *Applied Statistics and Probability for Engineers*. John Wiley & Sons, 2nd edition.

NewCombe, H. B., Kennedy, J. M., Axford, S. J., and James, A. P. (1959). Automatic linkage of vital records. *Science*, 130(3381):954--959.

On, B.-W., Lee, D., Kang, J., and Mitra, P. (2005). Comparative study of name disambiguation problem using a scalable blocking-based framework. In *Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 344--353, Denver, USA.

Pasula, H., Marthi, B., Milch, B., Russell, S., and Shpitser, I. (2002). Identity uncertainty and citation matching. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 1401–1408, Vancouver, Canada. MIT Press.

Pereira, D. A., Ribeiro-Neto, B., Ziviani, N., and Laender, A. H. F. (2008). Using web information for creating publication venue authority files. In *Proceedings of the 8th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 295--304, Pittsburgh, USA. ACM New York, NY, USA.

Pereira, D. A., Ribeiro-Neto, B., Ziviani, N., Laender, A. H. F., and Gonçalves, M. A. (2009a). A generic web-based entity resolution framework. *Journal of the American Society for Information Science and Technology*. Submitted.

Pereira, D. A., Ribeiro-Neto, B., Ziviani, N., Laender, A. H. F., Gonçalves, M. A., and Ferreira, A. A. (2009b). Using web information for author name disambiguation. In *Proceedings of the 9th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 49--58, Austin, USA. ACM New York, NY, USA.

Sarawagi, S. and Bhamidipaty, A. (2002). Interactive deduplication using active learning. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 269--278, Edmonton, Canada. ACM New York, NY, USA.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1--47.

sik Kim, H. and Lee, D. (2007). Parallel linkage. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management*, pages 283--292, Lisbon, Portugal. ACM, New York, NY, USA.

Snyman, M. M. M. and van Rensburg, M. J. (2000). Revolutionizing name authority control. In *Proceedings of the 5th ACM International Conference on Digital Libraries*, pages 185--194, San Antonio, USA.

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

SpectralLIB (2009). SpectralLIB package. http://www.stat.washington.edu/spectral. Accessed in January, 2009.

Tan, Y. F., Kan, M.-Y., and Lee, D. (2006). Search engine driven author disambiguation. In *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 314--315, Chapel Hill, USA.

Tejada, S., Knoblock, C. A., and Minton, S. (2001). Learning object identification rules for information integration. *Information Systems*, 26(8):607--633.

Tejada, S., Knoblock, C. A., and Minton, S. (2002). Learning domain-independent string transformation weights for high accuracy object identification. In *Proceedings*

of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 350--359, Edmonton, Canada. ACM, New York, NY, USA.

Tillett, B. B. (2000). Authority control on the web. In *Proceedings of the Bicentennial Conference on Bibliographic Control for the New Millennium: Confronting the Challenges of Networked Resources and the Web*, Washington, USA.

Tillett, B. B. (2001). A virtual international authority file. In *Proceedings of the 67th IFLA Council and General Conference*, Boston, USA.

Tillett, B. B. (2004). Authority control: State of the art and new perspectives. *Cataloging & Classification Quarterly*, 38(3-4):23--41.

Treeratpituk, P. and Giles, C. L. (2009). Disambiguating authors in academic publications using random forests. In *Proceedings of the 9th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 39--48, Austin, USA. ACM New York, NY, USA.

VIAF (2008). VIAF: The virtual international authority file. http://www.oclc.org/research/projects/viaf/default.htm. Accessed in January, 2008.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Velag, New York, USA.

Warner, J. W. and Brown, E. W. (2001). Automated name authority control. In *Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 21--22, Roanoke, USA.

Wick, M., Culotta, A., Rohanimanesh, K., and McCallum, A. (2009). An entity based model for coreference resolution. In *Proceedings of the 9th SIAM International Conference on Data Mining*, pages 365–376, Sparks, USA.

Winkler, W. E. (1999). The state of record linkage and current research problems. *Statistical Research Division, U.S. Census Bureau*. Internal Revenue Service Publication R99/04, http://www.census.gov/srd/papers/pdf/rr99-04.pdf.

Witten, I. H. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, second edition.

Xia, J. (2006). Personal name identification in the practice of digital repositories. *Program: Electronic Library & Information Systems*, 40(3):256--267.