Ranking de Publicações baseado na Extração de Textos da Internet

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Henrique Przibisczki de Oliveira e aprovada pela Banca Examinadora.

Campinas, 9 de dezembro de 2009.

Ricardo de Oliveira Anido - IC/UNICAMP
(Orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecária: Maria Fabiana Bezerra Müller – CRB8 / 6162

Oliveira, Henrique Przibisczki de
OL4r Ranking de publicações baseado na extração de textos da Internet/Henrique Przibisczki de Oliveira-- Campinas, [S.P. : s.n.], 2009.

Orientador : Ricardo de Oliveira Anido.
Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1.Publicações científicas. 2.Classificações bibliográficas. 3.Bibliometria. 4.Indexação automática. 5.Recuperação da informação. 6.Referências bibliográficas. I. Anido, Ricardo de Oliveira. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Título em inglês: Ranking of publications based on extraction of texts of the Internet

Palavras-chave em inglês (Keywords): 1.Science publishing.. 2.Bibliographic classification. 3.Bibliometrics. 4.Automatic indexing. 5.Information retrieval. 6.Bibliographical references.

Área de concentração: Metodologia e Técnicas da Computação

Titulação: Mestre em Ciência da Computação

Banca examinadora: Prof. Dr. Ricardo de Oliveira Anido (IC-UNICAMP)
Prof. Dr. Altigran Soares da Silva (UFAM)
Prof. Dr. Jacques Wainer (IC-UNICAMP)

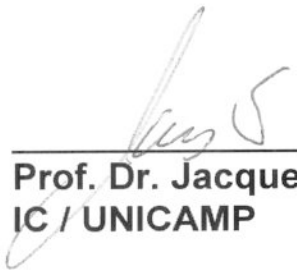
Data da defesa: 04/12/2009

Programa de Pós-Graduação: Mestrado em Ciência da Computação

TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 04 de dezembro de 2009, pela Banca examinadora composta pelos Professores Doutores:

Prof. Dr. Altigran Soares da Silva
DCC / UFAM



Prof. Dr. Jacques Wainer
IC / UNICAMP

Prof. Dr. Ricardo de Oliveira Anido
IC / UNICAMP

Instituto de Computação
Universidade Estadual de Campinas

Ranking de Publicações baseado na Extração de Textos da Internet

Henrique Przibisczki de Oliveira[1]

Dezembro de 2009

Banca Examinadora:

- Ricardo de Oliveira Anido - IC/UNICAMP (Orientador)
- Altigran Soares da Silva - UFAM
- Jacques Wainer - IC/UNICAMP
- Rogério Drummond B. P. de Mello Filho - IC/UNICAMP (Suplente)
- Roberto Marcondes Cesar Jr. - IME/USP (Suplente externo)

[1]Suporte financeiro de: Bolsa da FAPESP (processo 07/57996-0) 2008–2009.

Resumo

Vários métodos de *ranking* atuais comparam os diversos veículos de publicação em relação à qualidade ou impacto. Esta informação é muito importante para que um pesquisador selecione veículos de renome para publicar suas pesquisas, ou mesmo, instituições podem promover seus pesquisadores baseando-se na qualidade dos veículos onde publicam. Esta informação sobre os veículos pode também ser valiosa para um governo destinar recursos às instituições ou uma empresa avaliar a qualidade de um candidato a um emprego.

Existem várias métricas distintas para realizar *ranking* de veículos, mas o ponto comum entre a maioria é o uso de citações. Portanto, por mais que um veículo seja bastante prestigiado pelos pesquisadores, se ele não for indexado em uma base sua qualidade não será considerada.

Este trabalho propõe um método para *ranking* de veículos de publicação obtendo as informações não de uma base de citações existente, mas de uma outra fonte de dados: a Web. As páginas dos professores de universidades são visitadas e delas são extraídas as suas publicações. De cada publicação é extraído o veículo e dessa forma, baseado nos veículos que um pesquisador quis exibir em sua página, os mesmos são ordenados. Este método irá contemplar veículos de publicação não existentes nas atuais bases de dados criando um novo *ranking* de publicações. Vários problemas computacionais interessantes são abordados neste trabalho: busca de informação na internet, segmentação textual, extração de componentes em uma referência bibliográfica e agrupamento.

Abstract

Several current ranking methods compare different publication venues in relation to quality or impact. This information is very important for a researcher to choose renowned venues to publish his research. Institutes could promote their researchers based on the quality of places they have published. This information about the venues can also be valuable for a government to allocate resources to universities, or for companies to evaluate the quality of a candidate for a job.

There are other distinct measures to perform a ranking of venues, but the idea in common among most of them is the use of citations. Therefore, despite the fact a venue is very prestigious for its researchers, if it is not indexed in a citation database, it will not be considered, since its "quality" cannot be measured.

This work proposes to construct a ranking of publication venues obtaining the information not from a database, but from another data source: the Web. The university professor's webpages are visited to extract the publications. The venue is extracted from each publication, and thus, based on venues which a researcher wanted to show in his webpage, they are ranked. This method will include publication venues that do not exist in current databases, creating a new ranking of publications. Many interesting computational problems are discussed in this work: information search on the internet, text segmentation, extraction of components in a bibliographic citation, and clustering.

Agradecimentos

Em primeiro lugar gostaria de agradecer à minha esposa por me apoiar em todas as etapas do mestrado e pela compreensão quando o trabalho extrapolava o horário normal e avançava noite adentro e fins de semana.

Agradeço também ao professor Ricardo Anido pelo auxílio nas atividades acadêmicas, iniciações científicas e projetos desde minha graduação.

Também não posso esquecer de agradecer aos professores(as) e estagiários(as) da academia que frequento por ajudar a manter minha saúde e a relaxar na prática de natação e musculação.

Agradeço ao apoio financeiro recebido pela FAPESP, pois sem ele não poderia desenvolver este trabalho.

Por fim agradeço a todos os familiares e amigos que fiz, dentro ou fora da Unicamp, que direta ou indiretamente me apoiaram na conclusão do mestrado.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Publicações são importantes no meio acadêmico para garantir a difusão do conhecimento gerado. Existe uma grande variedade de veículos de publicação, como periódicos (*journals*), conferências, *sites* web, entre outros, e uma grande diversidade de qualidade entre eles. Ordenar (fazer um *ranking*) de acordo com algum critério, por exemplo em relação à qualidade, influência ou abrangência, é importante por várias razões: um pesquisador pode utilizar um *ranking* para selecionar um veículo de renome para sua publicação, uma instituição pode promover seus pesquisadores baseado na qualidade dos veículos onde publicam, o governo pode destinar suas verbas para instituições que publicam em veículos de grande impacto, uma empresa pode utilizar um *ranking* para avaliar as publicações de um candidato, etc. Há várias métricas possíveis para ordenar os veículos, e essas métricas geram *rankings* distintos. Estas diferenças de ordenação fazem com que os pesquisadores e/ou instituições estejam constantemente criando e avaliando métricas de modo a obter resultados mais confiáveis ou fiéis com a realidade. Um novo método de *ranking* de veículos de publicação pode dar a devida atenção a várias publicações antes não prestigiadas. Entende-se por *veículo de publicação* o local onde um trabalho é publicado, por exemplo, *Symposium on Principles of Programming Languages (POPL)* e *ACM Transactions on Programming Languages and Systems (TOPLAS)* são dois importantes nomes de veículos na área da Ciência da Computação.

Atualmente, a grande parte dos *rankings* utilizados pelos pesquisadores são os baseados em número de citações. A citação é uma referência feita em um documento que indica a origem da informação apresentada. Para que seja viável a avaliação e comparação entre os diferentes periódicos, as citações são indexadas em bases de dados que podem ser públicas (DBLP - "DataBase systems and Logic Programming" [30]) ou privadas (SCI - "Science Citation Index" [8]). Estas bases armazenam as citações, ou seja, uma estrutura que mostra a relação entre um documento citado e o citante, sendo que algumas bases armazenam também os documentos (textos) citados e os que realizam a citação [48].

O aspecto recorrente nestas análises é a forte influência de periódicos na medição da qualidade do material publicado e de seus autores [33], já que a grande maioria das bases existentes não indexam conferências. No entanto, o aumento da popularidade das conferências é significativo, especialmente na área de Ciência da Computação [38]. O estudo da crescente importância das conferências para a computação torna-se relevante, pois as atuais métricas podem não dar a devida consideração a esses veículos [42].

Procurando dar a devida atenção a uma maior variedade de veículos de publicação, o objetivo deste trabalho é propor um novo *ranking* de veículos. Nesse novo *ranking*, os veículos são ordenados baseando-se em sua "reputação". A "reputação", neste trabalho, procura capturar o prestígio atribuído a um determinado veículo pelos pesquisadores, analisando as publicações que os mesmos decidiram exibir em suas páginas de Internet. Parte-se da hipótese de que se o pesquisador colocou em seu *curriculum* Web uma publicação que ele realizou, ele atribuiu uma "boa reputação" ao veículo de publicação, pois se esse pesquisador considerasse que essa publicação fosse irrelevante ou desprestigiada, frente a suas outras publicações, então ele provavelmente a omitiria.

Para construção do *ranking* de veículos utiliza-se um "*ranking* padrão" escolhido entre instituições já existentes, como por exemplo, NRC [12], Times [16], SJTU [3], USnews [13] e USAtoday [5]. Os veículos extraídos das páginas dos professores são pontuados de modo que seja preservado a ordem das instituições no *ranking* escolhido. Exemplo: se o *ranking* do USnews coloca a instituição "Harvard" em primeiro e "Princeton" em segundo então deve-se pontuar os veículos de publicação de modo que ao somar os pontos dos veículos onde os pesquisadores de "Harvard" publicam obtenha-se um valor maior que a soma dos pontos dos veículos onde os pesquisadores de "Princeton" publicam.

Este trabalho concentra-se na área de Ciência da Computação das instituições, dessa forma o entendimento sobre os dados é maior, possibilitando o melhor refinamento das ferramentas. No entanto após terminado o processo para uma determinada área o mesmo pode ser feito de modo análogo para outras áreas cabendo certos ajustes.

Diferentemente de outros métodos, a extração diretamente do currículo obtém dados referentes às publicações desconsideradas em outros *rankings*. Assim pode-se fazer uma nova análise sobre a qualidade dos veículos, como a computação da relação de tipos determinados, como conferências e periódicos.

Os resultados obtidos podem ser utilizados pela comunidade acadêmica para melhor escolha do local de publicação, para comparar a produção de autores, grupos de autores, departamentos, bem como avaliar a posição de suas instituições de pesquisa nos cenários internacionais, com a vantagem de que a metodologia é aplicável em qualquer área de conhecimento.

Neste trabalho, vários problemas computacionais interessantes são abordados. Os robôs de busca precisam analisar páginas HTMLs, avançar por *links* e decidir quando sal-

var uma página contendo publicações. Nesta página com inúmeras referências é necessário identificar os pontos onde cada uma inicia e termina, pois pressupõe que a ferramenta que obtém o nome do veículo irá trabalhar sobre um texto que representa uma única referência bibliográfica. Para cada texto de referência é necessário criar métodos de separação dos componentes (autores, título, etc.) para identificar o nome do veículo de publicação. O foco é sobre a obtenção do nome do veículo, mas encontrar no texto os segmentos que correspondem a outros componentes exclui regiões da referência e auxilia no isolamento do veículo de publicação. Após obter todos os nomes de veículos será necessário agrupá-los para identificar quais representam o mesmo veículo, computando sua frequência e calculando o *ranking*.

Como resultado, este trabalho criou não somente um *ranking* de veículos de publicação, mas também contabilizou os veículos que mais ocorreram e os mais abrangentes. Um outro resultado foi a classificação dos veículos, identificando os periódicos e as conferências. Com esses dados analisou-se a variação destes valores ao longo do anos, mostrando que o número médio de conferências por pesquisador vem aumentando com o passar do tempo. De modo geral, este trabalho confirmou o aumento da influência das conferências.

Capítulo 2

Conceitos e Revisão Bibliográfica

Neste capítulo são apresentados os principais conceitos e definições necessários para formar a base do projeto, além de uma revisão bibliográfica sobre o assunto.

2.1 Citações

A citação é um reconhecimento que um documento recebe de outro, uma indicação de que aquele documento é a fonte de uma informação. A citação serve para indicar a fonte de alguma informação importante que foi utilizada no trabalho e também funciona como uma forma de homenagem aos autores que originaram as idéias referenciadas [48].

Análise de Citações

A análise de citações é o nome dado à técnica de verificação da influência e relevância da pesquisa publicada baseada nas citações relacionadas com o trabalho analisado. A contagem do número de vezes que uma publicação é citada em textos científicos é um método de avaliação representando as ligações cognitivas entre as idéias [48]. O primeiro uso de citações na avaliação de veículos de publicação foi realizada por Eugene Garfield entre meados de 1950 e início de 1960, em um projeto realizado na literatura médica das forças armadas americanas. Com base em suas pesquisas e inúmeros projetos realizados, Garfield criou a base de indexação SCI (*Science Citation Index*) que é atualmente uma das mais detalhadas bases de indexação de periódicos científicos [8].

Um problema associado à contagem do número de citações é o fato de ela não representar o contexto em que ocorreu a citação. Por exemplo, se um artigo é citado porque seus experimentos foram ratificados, então ele recebeu uma boa qualificação, de outra forma, se este artigo é citado porque ele cometeu erros, então sua qualidade foi posta em dúvida. Em ambos os casos conta-se uma citação para o periódico, o que representaria

algo favorável [35].

Veículos Citados (cited) e Citantes (citing)

No contexto de análise das referências, quando um artigo no veículo A referencia um artigo no veículo B é dito que B é um veículo citado (*cited*) e A é um veículo citante (*citing*). Uma abordagem análoga é dizer que o veículo B é a fonte do conhecimento (*source*) e que o veículo A é o armazenador de conhecimento (*storer*) [33].

Fator de Impacto

O Fator de Impacto é um índice bibliométrico baseado no número de vezes que artigos de um determinado periódico são citados, e serve de mecanismo para a classificação, avaliação, categorização e comparação entre periódicos. O valor para um determinado periódico pode ser obtido dividindo-se o número de citações feitas no ano corrente para publicações (daquele periódico) no período de dois anos atrás, pelo número de artigos publicados nestes dois anos [15].
Por exemplo, o cálculo do fator de impacto de um periódico X em 2007 é:

- A = número de citações para artigos publicados por X entre 2005 e 2006 feitas por artigos publicados em 2007.
- B = número de artigos publicados entre 2005 e 2006.
- Fator de Impacto de 2007 = A/B.

A idéia do Fator de Impacto foi inicialmente proposta em 1955 na revista *Science* por Eugene Garfield e foi a referência primordial para o *Science Citation Index* [4]. Na década de 60, Eugene Garfield e Irving Sher criaram o Fator de Impacto para periódicos para auxiliar na escolha de periódicos da SCI. O índice de citações para autores foi reordenado para indexar citações para periódicos, o que mostrou a existência de grandes grupos de periódicos altamente citados e que precisam ser contemplados pela SCI. A ordem neste *ranking* era dependente unicamente do total de publicações e da contagem de citações, o que fazia com que alguns periódicos importantes não fossem reconhecidos. Dessa forma foi criado o Fator de Impacto que era um método simples de comparação entre periódicos não dependente diretamente do número e frequência das citações.

No começo o Fator de Impacto era usado tanto para analisar o impacto de periódicos como de autores. Isto causava problemas pois os periódicos possuem um grande número de artigos publicados e citações envolvidas, mas os autores produzem uma quantia muito menor. Atualmente o mais importante uso do Fator de Impacto é na avaliação acadêmica,

sendo utilizado para se obter uma estimativa do prestígio de um periódico no qual os pesquisadores têm publicado [15].

O Fator de Impacto e o *ranking* dos periódicos podem ser influenciados por muitos aspectos; um bom exemplo são os artigos de revisão ("review articles"). Este tipo de artigo é normalmente bastante citado por pesquisadores, pois serve de substituto para a literatura recente. O sistema JCR [10] marca um artigo como "review" se ele contiver no título as palavras "review" ou "overview", se ele for publicado na seção de "review" do periódico, ou se ele possui mais que 100 referências bibliográficas. Dessa forma, o JCR fornecerá tanto o número de artigos de revisão em um periódico, como também sua média de citações por artigo, permitindo ao pesquisador identificar o porque do alto Fator de Impacto de determinado periódico.

Mesmo sendo um método bastante utilizado para avaliar um periódico, o Fator de Impacto pode não ser válido para outros tipos de análises, pois como é necessário que as citações estejam indexadas em uma base de dados, muitos veículos podem não estar presentes. Apesar de sua relevância na análise de um periódico, o Fator de Impacto não permite uma análise centrada em um único artigo ou um único autor [35].

Outros problemas são apontados por Kleijnen *e*t al. [27], como por exemplo o fato de que o Fator de Impacto da forma como foi concebido privilegia, com um alto valor, periódicos mais gerais (que englobam mais subcategorias de uma disciplina), em detrimento de periódicos especialistas. Da mesmo forma como Garfield escreveu, artigos que reúnem as informações recentes da literatura ("survey") possuirão mais citações do que artigos técnicos. O Fator de Impacto também sofre muita variação dentre as subcategorias de uma disciplina, o exemplo dado por Kleijnen mostra que em 1996 o Fator de Impacto para a subcategoria "Sistemas de Informação" era 1.777, enquanto que para a subcategoria "Inteligência Artificial" era 2.654. A janela de tempo de dois anos usada no cálculo do impacto não é o melhor valor para todos os campos. De acordo com Kleijnen, enquanto em Economia um valor bom para a janela de tempo é de três a quatro anos, para a área de Sistemas de Informação seria necessário uma janela muito maior.

Análise de Meia-Vida (half-life)

O índice de meia-vida de um periódico é o número de anos, contados para trás a partir do atual, que perfazem 50% do número total de citações recebidas pelo periódico no ano analisado [7]. Por exemplo, pretende-se calcular o índice de meia-vida para um periódico X em 2007, e as informações sobre o número de citações recebidas estão na Tabela 2.1.

No ano de 2007, o periódico X recebeu 80 citações. Portanto para atingir o total de 40 citações (50%) deve-se considerar quatro anos: de 2007 até 2004. Dessa forma, o índice de meia-vida para o periódico X vale 4.

O índice de meia-vida é usado para medir o tempo médio que os artigos publicados no

Tabela 2.1: Número de citações recebidas por X em 2007

Ano do Artigo Citado	Número de Citações
até 2003	40
2004	20
2005	10
2006	5
2007	5

periódico analisado continuam relevantes. Por isto esse índice é também conhecido por se um medidor da obsolescência dos artigos.

Índice H

Em 2005, Jorge E. Hirsch propôs o índice H que fornece uma estimativa da importância, significância e impacto das contribuições dos pesquisadores [25]. Por ser um físico, seu trabalho foi concentrado nesta área, mas o índice pode ser usado para qualquer outra disciplina científica. O cálculo do índice H é simples e considera o número de artigos publicados e o número de citações recebidas por cada um deles. Um pesquisador recebe o valor H se H de seus N_p artigos foram citados no mínimo H vezes e os outros $(N_p - H)$ artigos não foram citados mais que H vezes. Por exemplo, se um pesquisador tem três artigos e eles foram citados 1, 2 e 3 vezes respectivamente, então o índice H será 2 porque 2 artigos receberam no mínimo 2 citações e $(3 - 2)$ artigos não receberam mais de 2 citações.

Um limitante inferior para o total de citações recebidas pelos artigos de um pesquisador pode ser obtido por H^2. Logicamente esse limitante subestima o número de citações recebidas, pois desconsidera os artigos mais citados e os que foram citados menos que H vezes. O número total de citações ($N_{c,tot}$) se relaciona com H da seguinte forma: $N_{c,tot} = aH^2$, onde a é uma constante que vale entre 3 e 5 (valores determinados experimentalmente)

O índice H é uma medida fácil de calcular e une a análise do número de artigos publicados e o número de citações recebidas por estes artigos, mas não possui as desvantagens que estes métodos isolados têm. Quando se analisa somente o número de publicações não se considera a importância e impacto dos artigos, e por outro lado, quando se utiliza somente o número de citações pode-se prejudicar a análise com artigos que obtiveram um alto número de citações somente em um curto período tempo e servem para inflar a computação do total, sendo discrepante com o número de citações recebidas pelos outros artigos publicados pelo mesmo autor.

Por outro lado, a métrica proposta por Hirsch recebeu críticas como as feitas por

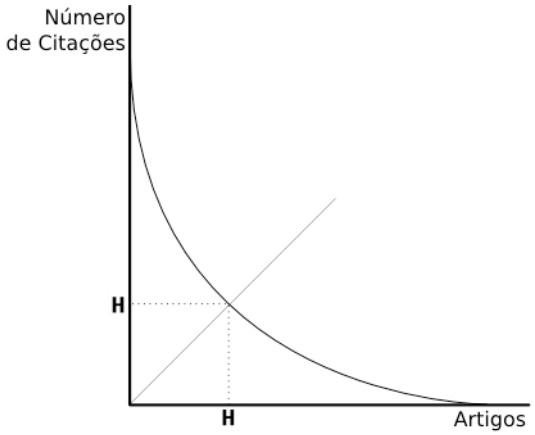


Figura 2.1: Esquema (extraído de [25]) de uma curva do número de citações para cada artigo, ordenado em ordem decrescente de citações. A interseção da linha de 45º com a curva resulta no índice H. O número total de citações é a área sob a curva.

Michael Nielsen [34]. Ele explicou como pode ser feito uma boa aproximação do índice H usando uma função simples baseada no número total de citações, dessa forma o índice H não conteria muita informação além de uma análise de citações padrão. Suponha que T é o número total de citações, então para muitos físicos (que foi a área em que Hirsch focou inicialmente seu trabalho) a seguinte função é uma boa aproximação: $H \approx \frac{\sqrt{T}}{2}$. Esta relação segue a partir da equação de Hirsch para estimar o número total de citações ($N_{c,tot} = aH^2$) e como a pode variar entre 3 e 5, então Nielsen calculou que a precisão do índice H possui um erro de mais ou menos 15%.

Esta aproximação não é adequada para dois tipos de pesquisadores: os que publicaram pouquíssimos artigos, simplesmente porque a distribuição de citações pelos artigos não é usual, e os pesquisadores que possuem um trabalho muitíssimo mais citado que qualquer outro trabalho, neste caso a função de aproximação superestima o índice H. Nesta última condição, a relação para aproximação do índice H pode ser modificada da seguinte forma: $H \approx b\sqrt{T'}$, onde T' é o número total de citações menos as mais citadas (retirar as duas mais citadas é um bom valor), e b é uma constante que precisa ser determinada experimentalmente.

Nielsen conclui que o índice H não é irrelevante, mas não é muito prático na maioria dos casos, já que não fornece mais informações além do obtido analisando o número de citações.

2.2 *Ranking*

O *ranking* é uma listagem de instituições ou veículos de publicação que fornece a colocação ou ordem entre as entidades em relação à qualidade ou prestígio. A instituição ou veículo em primeiro lugar é considerado melhor (de acordo com algum critério) que as outras entidades avaliadas. Alguns importantes *rankings* de instituições são: NRC [12], Times [16], SJTU [3], USnews [13] e USAtoday [5], que usam critérios objetivos e subjetivos para avaliar os programas das instituições. De forma análoga, os *rankings* de veículos de publicação utilizam alguma métrica quantitativa para ordenar os veículos. Essa métrica pode ser, por exemplo, o Fator de Impacto, o índice H, índice de meia-vida, ou mesmo a contagem de citações.

Uma das mais famosas ferramentas para *ranking* de veículos é o JCR [10], que usa informações estatísticas baseadas em referências de citação de artigos. Essa ferramenta auxilia na medição da influência e impacto de periódicos e fornece uma grande variabilidade de métricas e campos de pesquisa.

Além de *rankings* de autores e instituições, existem diversos outros tipos, tais como de autores (ISI *Highly Cited* [46]), países e áreas de pesquisa. Quando um *ranking* se utiliza de publicações para realizar uma ordenação, tipicamente, ele atribui uma pontuação para cada veículo, e a ordem dos autores ou instituições (ou algum outro conjunto) é obtida a partir da soma das pontuações dos locais onde se publicou [42]. O último *ranking* que utilizou publicações para ordenar instituições na área de Ciência da Computação foi finalizado em 1996 por Geist *et* al. [42]. Eles selecionaram 17 periódicos publicados pela ACM ("Association for Computing Machinery") ou IEEE ("Institute of Electrical and Electronics Engineers") e cada um desses 17 periódicos recebia um ponto para cada vez que era citado por algum outro periódico entre Janeiro de 1990 a Maio 1995. O *ranking* que utiliza publicações tem a vantagem de poder ser feito unicamente sobre uma base de dados com uma metodologia objetiva. Por outro lado, os *rankings* de instituições/escolas (que não utilizam unicamente publicações) necessitam de um processo científico e social complexo, muitas vezes envolvendo questionários e visitas presenciais nos locais envolvidos.

Uma das desvantagens apontadas por Jien Ren e Richard N. Taylor nos *rankings* que utilizam publicações para realizar uma ordenação é que em alguns casos a obtenção de parte das informações (ou em sua totalidade) é feita manualmente, resultando em um número muito menor de artigos, periódicos e faixas de tempo considerados. Isto implica diretamente na redução da pontuação dos veículos, o que altera o *ranking*. O fato de muitas vezes as informações serem obtidas manualmente pode explicar o porque de se considerar exclusivamente periódicos, em detrimento de outros tipos de veículos igualmente importantes como as conferências.

Uma outra limitação do *ranking* que utiliza publicações é que ele é restrito a somente

um campo científico. Um novo *ranking* precisa ser construído para se avaliar uma nova área, repetindo-se os mesmo procedimentos feitos anteriormente. Uma mesma métrica pode ser aplicada em várias áreas e todo processamento feito uma única vez, mas isto é equivalente a processar cada área separadamente, pois os *rankings* obtidos não podem ser combinados. Cada área da ciência possui um padrão de tempo e número de citações, dessa forma, o melhor veículo de um certo campo científico pode estar com uma pontuação muitas vezes abaixo de um veículo de médio prestígio em outro campo da ciência.

Uma última desvantagem é que o critério adotado para qualificar as publicações não pode ser alterado sem ter que repetir todo o processamento feito.

2.3 Correlação entre *Rankings*

Métricas distintas de ordenação geram *rankings* diferentes. Mesmo existindo diferenças entre duas ordenações algumas propriedades podem estar presentes em ambas, como por exemplo, alguns elementos de um dos *rankings* podem aparecer entre os primeiros do outro *ranking*, mas com alguma variação de ordem. As Tabelas 2.2 e 2.3 ilustram esse comportamento.

Tabela 2.2: *Ranking* das primeiras 10 universidades de acordo com o USnews [13]

Rank	Universidade
1	Massachusetts Institute of Technology
1	Stanford University
1	University of California–Berkeley
4	Carnegie Mellon University
5	University of Illinois–Urbana-Champaign
6	Cornell University
6	Princeton University
6	University of Washington
9	Georgia Institute of Technology
9	University of Texas–Austin

Esse exemplo mostrou uma propriedade entre os dois *rankings*: alguns elementos são considerados bem qualificados em ambas as ordenações. Portanto, quando dois *rankings* são comparados, a simples afirmação que eles não são exatamente iguais não é suficiente. O importante, para uma análise científica, é verificar o quão parecidas são as duas ordenações. Desse modo surge o conceito do cálculo da correlação entre *rankings*.

Uma famosa métrica para calcular a correlação entre *rankings* foi proposta por Kendall em 1955 [2]. O coeficiente de correlação de Kendall mede a similaridade entre dois *rankings* que possuem o mesmo conjunto de elementos. Utiliza-se para o cálculo do coeficiente o

Tabela 2.3: *Ranking* das primeiras 10 universidades de acordo com o Times [16]

Rank	Universidade
1	Massachusetts Institute of Technology
2	University of California, Berkeley
3	Stanford University
4	California Institute of Technology
5	University of Cambridge
6	Carnegie Mellon University
7	Imperial College London
8	Georgia Institute of Technology
9	University of Tokyo
10	University of Toronto

número de pares de elementos que estão invertidos entre as duas ordenações. Portanto, cada *ranking* é representado por um conjunto de pares de elementos que denotam a ordem entre eles. Para um conjunto de $N = 4$ elementos, por exemplo, tem-se a seguinte representação:

$$\Phi = \{a, b, c, d\}$$

Para esse conjunto de elementos pode-se ter a ordem $\phi_1 = [a, c, b, d]$, o que resulta no *ranking* $\rho = [1, 3, 2, 4]$. O conjunto de N elementos pode ser decomposto em $\frac{1}{2}N(N-1)$ pares ordenados. O par ordenado indica que o primeiro item do par está na frente, na ordenação, do segundo item do par. Para o exemplo utilizado, a ordem ϕ_1 gera o seguinte conjunto de seis pares ordenados:

$$\Phi_1 = \{[a, c], [a, b], [a, d], [c, b], [c, d], [b, d]\}$$

O *ranking* ϕ_1 só poderá ser comparado com *rankings* que ordenam os mesmos elementos do conjunto Φ, e o método de Kendall usará a soma do número de pares ordenados diferentes entre os dois conjuntos. O valor da diferença entre eles é chamado de *distância*, que é denotada por $d_\Delta(\Phi_1, \Phi_2)$, onde Φ_x é o conjunto de pares ordenados oriundos da ordenação ϕ_x.

O coeficiente de correlação de Kendall é obtido através da normalização da distância na faixa de valores entre -1 e +1. O valor -1 denota que as ordenações são totalmente diferentes, ou seja, uma é o inverso da outra. A máxima proximidade é obtida com o valor +1 que denota que as ordenações são iguais. Sabendo que o número máximo de pares de um conjunto é:

$$\tfrac{1}{2}N(N-1),$$

então o maior valor de distância possível é:

$$N(N-1)$$

O coeficiente de correlação de Kendall é definido como a diferença entre o número de pares corretos e o número de pares diferentes: $\tau = P(iguais) - P(diferentes)$, portanto pode-se expandir a fórmula deste modo:

$$\begin{aligned}
\tau = &\quad P(iguais) - P(diferentes) \\
= &\quad \frac{N(N-1) - d_\Delta(\Phi_1, \Phi_2)}{N(N-1)} \quad - \frac{d_\Delta(\Phi_1, \Phi_2)}{N(N-1)} \\
= &\quad \frac{N(N-1) - d_\Delta(\Phi_1, \Phi_2) - d_\Delta(\Phi_1, \Phi_2)}{N(N-1)} \\
= &\quad \frac{N(N-1) - 2d_\Delta(\Phi_1, \Phi_2)}{N(N-1)} \\
= &\quad 1 - 2\frac{d_\Delta(\Phi_1, \Phi_2)}{N(N-1)}
\end{aligned}$$

Para exemplificar o cálculo, suponha a ordem ϕ_1 mostrada acima sendo comparada com a ordem $\phi_2 = [a, c, d, b]$ que representa o *ranking* $\rho = [1, 3, 4, 2]$. Os conjuntos de pares ordenados serão:

$$\begin{aligned}
\Phi_1 &= \{[a,c], [a,b], [a,d], [c,b], [c,d], [b,d]\} \\
\Phi_2 &= \{[a,c], [a,d], [a,b], [c,d], [c,b], [d,b]\}
\end{aligned}$$

O conjunto dos pares diferentes entre Φ_1 e Φ_2 é $\{[b,d], [d,b]\}$, portanto $d_\Delta(\Phi_1, \Phi_2) = 2$. Pela fórmula dada acima, o coeficiente de correlação de Kendall é:

$$\begin{aligned}
\tau = &\quad 1 - 2\frac{d_\Delta(\Phi_1, \Phi_2)}{N(N-1)} \\
= &\quad 1 - \frac{2 \times 2}{12} \\
= &\quad 1 - \tfrac{1}{3} \\
\approx &\quad 0,67
\end{aligned}$$

Neste exemplo, o valor $0,67$ representa uma boa correlação entre as duas ordenações. Isto pode ser visto pelos elementos a e c (50% dos elementos) que em ambos *rankings* estão na mesma posição.

2.4 Busca de Informação Bibliográfica na Internet

Os artigos científicos estão amplamente disponibilizados na Web, seja através de bases de dados, sites de periódicos publicados eletronicamente (e-journals) ou em páginas pessoais de seus autores. Contudo, estes dados não estão indexados, com exceção dos pertencentes a uma base indexadora como, por exemplo, o CiteSeer [21] e o SCISEARCH [23]. Estas duas bases de dados citadas são conhecidas por ACI (*Autonomous Citation Indexing*), que são sistemas que localizam automaticamente artigos na Internet e extraem as informações contidas indexando-as.

Qualquer que seja o ponto de partida para obtenção das informações das publicações na Web, o processo de busca precisa ser automatizado. O agente de software (ou robô de busca) a ser usado para obter os dados das publicações na Internet possui muitas características análogas com os agentes utilizados para indexar as citações pela Web [17, 24, 29].

Na maioria dos casos, o que é recuperado da Internet são documentos PDF, Postscript, DOC, HTML ou TXT, e sobre estes dados é necessária a análise e recuperação das informações sobre os veículos de publicação.

Um projeto recente que utiliza informações advindas da internet é o MESUR [26]. Este projeto foi fundado por Andrew W. Mellon e constitui um esforço para analisar o impacto utilizando métricas baseadas em uso. As informações de uso são os dados referentes aos acessos feitos (à *links* ou arquivos) na internet. A base conta com um bilhão de eventos de uso, além de toda informação associada aos dados bibliográficos e de citação de importantes instituições, agregadores e editoras. Os dados sobre eventos de uso mostram os aspectos de todas as fases de uma produção científica, que dificilmente é obtida somente com a estatística de citações. Um pesquisador inicia seu ciclo na produção de um trabalho com o desenvolvimento de uma idéia, resultando eventualmente em uma publicação e posterior citação. Portanto as citações ocorrem somente no final do ciclo, mas todo o processo envolvido na criação do artigo, tais como, busca por informações relacionadas, *download* de artigos, envio por e-mail para outros autores, leitura e uma cópia para posterior leitura, está desconsiderado. Esses eventos associados à criação do artigo são eventos de uso e refletem toda a dinâmica do processo científico

As atividades, do projeto MESUR, foram as seguintes:

- *Criação de um Conjunto de Dados de Referência*: O projeto MESUR processa uma vasta quantidade de arquivos de *logs* de uso associados às informações bibliográficas e de citação de 14 importantes agregadores, editoras e instituições.

- *Pesquisa de Tendência dos Dados*: MESUR examinou amostras dos dados e verificou se esse conjunto de informações poderia ser usado para representar as características globais de uso.

- *Geração de Mapas de Ciência*: Os mapas de ciência são a base do conjunto de dados, ajudando a dar significado aos dados de uso, indicando a validade dos dados coletados e auxiliando na análise dos efeitos específicos de cada domínio.

- *Definição e Validação de uma Métrica Baseada em Uso*: Definiu-se uma métrica sobre o conjunto dados para calcular um impacto baseado em uso, que avalia a confiança e validade das informações.

MESUR gerou dois mapas de ciência baseado em acesso de periódicos: um baseado em citações e o outro em eventos de uso. A rede de periódicos baseada em citações conecta os indivíduos baseando-se na relação de citação de um periódico para outro. Na Figura 2.2 é ilustrado esse tipo de rede. Cada linha da lista representa uma conexão entre um par de periódicos, e o número de citações indica a força da conexão.

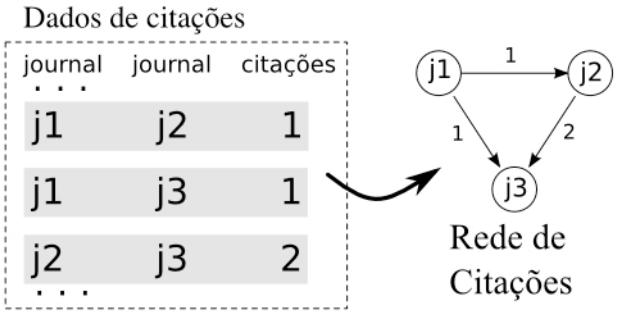


Figura 2.2: Rede de periódicos baseada em citação.

A rede de periódicos baseada em eventos de uso é criada de outra forma. Os dados de uso não fornecem a informação de uma conexão periódico a periódico, mas fornece uma lista sequêncial no tempo de eventos de uso (no nível de artigos), de onde pode ser derivado conexões entre os periódicos. MESUR realiza a conexão entre periódicos utilizando uma abordagem utilizada em serviços de internet (como a *Amazon.com*). A idéia é definir que o grau que dois itens estão conectados é uma função da frequência que os usuários realizam conjuntamente uma compra, um download, ou um acesso. Esta relação é conhecida como coocorrência de uso e é usada para criar a rede de conexão entre periódicos. MESUR se baseia em acessos por sessão, pois as informações dos usuários na base de dados foi ocultada para permitir o anonimato dos pesquisadores. O grau da conexão entre dois periódicos na rede baseada em uso é uma função da frequência com que eles foram conjuntamente acessados na sessão de usuário anônima. A Figura 2.3 ilustra essa rede.

Definido uma forma de relacionar dois periódicos, vários tipos de métricas podem ser utilizadas para analisar o impacto, e dessa forma essa métricas serão baseadas em eventos de uso. Alguns tipos de análise de impacto são:

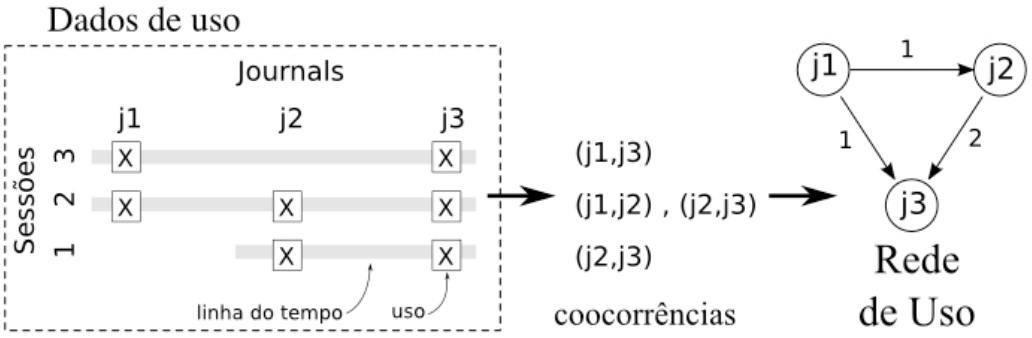


Figura 2.3: Rede de periódicos baseada em eventos de uso.

- **Popularidade**: Número de conexões de entrada e/ou saída nos periódicos.
- **Dispersão**: Distribuição de conexões de entrada e/ou saída nos periódicos.
- **Amigo-do-amigo**: Média entre os tamanhos dos caminhos mais curtos entre os periódicos.
- **Poder**: Número de vezes que um periódico aparece no caminho mais curtos.
- **Prestígio**: Valor do prestígio do periódico que se relaciona com outro periódico.

O projeto MESUR apresentou uma nova forma de analisar o ciclo de produção científica criando uma métrica baseada em eventos de uso. Essa nova forma de avaliação ainda não foi totalmente aceita, pois ainda falta um entendimento maior sobre a base de dados sobre eventos de uso e de como aplicar métricas sobre este conjunto. Na Figura 2.4 é exibido o *ranking* baseado em uso comparado com o fator de impacto (do ano de 2005), mostrando que essa abordagem difere completamente de uma das mais utilizadas formas de avaliação de periódicos.

Uso		FI (2005)		
Rank	**Valor**	**Rank**	**Valor**	**Journal**
1	15,830	6	30,927	SCIENCE
2	15,167	11	29,273	NATURE
3	12,798	89	10,231	PNAS
4	10,131	5989	0,402	LNCS
5	8,409	235	5,854	J BIOL CHEM

Figura 2.4: Ranking baseado em eventos de uso em comparação com o fator de impacto (FI) de 2005.

2.5 Segmentação Textual ou de Contexto

A segmentação textual é o processo de determinação das posições no texto onde ocorrem mudanças de tópicos, de sentenças, de parágrafos, de conteúdo, entre outras, onde estas escolhas são particulares à aplicação [37, 40].

Quando se tenta recuperar certas informações desejadas em um documento textual em linguagem natural o processo de extração de informação será tão mais difícil quanto mais complexa a estrutura do conteúdo e a variabilidade de formas de representação [44].

Nesse processo, uma melhor exatidão é obtida se ao invés da informação ser procurada em todo texto de entrada, ela for procurada somente no "trecho mais relevante" do texto. Um exemplo seria procurar o nome do periódico somente no segmento de texto resultante da referência de entrada descartando-se o nome dos autores e o título do trabalho [41].

Basicamente existem três maneiras de se segmentar o texto, a saber, a baseada em regras, a análise numérica e estatística, e a baseada em aprendizado de máquinas [37]. O processo baseado em regras usa expressões regulares para identificar no texto marcadores e com base nos marcadores usa novas expressões para identificar os segmentos, trabalhando muito bem quando o conjunto de regras é relativamente pequeno e os segmentos desejados são livres de ambiguidade e de contexto [37]. Um exemplo de uso é na identificação de datas no texto, onde a primeira regra usada marca todos os números e depois aplica-se outra regra para encontrar os números separados por barras no formato dia, mês e ano.

Na análise numérica e estatística é proposto modelos probabilísticos para determinar os segmentos no texto. O cálculo da probabilidade utiliza a contagem de símbolos previamente marcados (palavras, números, separadores, etc.) e suas posições no texto [37, 47]. Um exemplo desta técnica é na separação de um texto em segmento dos autores e segmento do título, onde o trecho com maior presença do separador vírgula e de abreviações caracteriza o segmento dos autores.

Por último, a segmentação baseada em aprendizado de máquina usa uma base de conhecimento para inferir sobre os possíveis segmentos. Os dados processados podem então servir para retroalimentar a base de dados. Um exemplo seria tentar identificar se um nome de veículo é uma conferência, para isso seria fornecida uma base de dados com diversos nomes rotulados em conferência e não conferência que servirá para o programa ser treinado e aprender como diferenciar os dois grupos (provavelmente analisando a frequência de palavra chaves).

2.6 Casamento Aproximado de Textos

O casamento aproximado de textos tem por objetivo verificar se dois trechos textuais são iguais, mas permitindo que existam alguns erros entre as sequências [32]. Basicamente

deseja-se encontrar um padrão textual dentro de um texto, onde tanto o padrão como o texto alvo podem estar corrompidos. Entende-se que um texto está corrompido se ele apresenta erros de grafia (omissão, adição, substituição ou transposição de caracteres), ou ainda se ele apresenta termos abreviados, que mesmo não se tratando de um erro semântico, dificulta o casamento do padrão. Quando se abrevia uma palavra no texto ocorre uma omissão de caracteres e a adição (na maioria dos casos) do caractere "ponto", analisando dessa forma, um texto com abreviações está corrompido.

O problema do casamento aproximado de textos é, de forma geral, procurar um padrão textual em um texto onde aquele padrão ocorre e permitir um número limitado de erros na verificação da igualdade. Dependendo do problema abordado usa-se diferentes modelos de contagem de erros para definir o quão diferentes são dois textos. Para expressar a diferença entre eles define-se a ideia de "distância" entre os dois textos, onde um valor pequeno indica que os textos são semelhantes e um valor alto indica textos diferentes. O modelo de contagem de erro adotado dá o valor da distância entre os segmentos comparados, enquanto um modelo pode indicar que dois trechos são diferentes, outro pode considerá-los iguais.

No contexto deste trabalho, um bom exemplo é a comparação de dois nomes de veículos de publicação. Normalmente alguns termos dentro do nome do veículo são abreviados, como por exemplo a palavra *journal* ou a palavra *conference*. Essa condição ocorre por exemplo no nome "Journal of the ACM", que aparece em muitas referências como "J. of the ACM", ou ainda como "J. ACM".

O estudo sobre o problema de procurar padrões em textos que possuem algum tipo de erro na grafia é bastante antigo: os primeiros estudos datam da década de 20 [32], e desde a década de 60 o casamento aproximado de textos é uma dos melhores métodos para resolver este problema. Estudos da época mostram que 80% dos erros encontrados são corrigidos realizando uma única adição, remoção, substituição, ou transposição de caractere. O modelo clássico que resolve este problema é chamado de "Distância de Edição". A Distância de Edição analisa o menor número de operações de adição, remoção, substituição e transposição necessárias para fazer com que os trechos de texto comparados fiquem iguais. Portanto se dois textos têm um valor distância igual a zero eles são iguais, porque nenhuma operação foi necessária para torná-los idênticos. Um exemplo do cálculo da distância de edição é o seguinte: as palavras "artigo" e "aritg" possuem distância de edição igual a 2, porque é necessário adicionar a letra "o" e transpor o "t" para imediatamente antes do "i" na segunda palavra para que ela fique exatamente igual à primeira. Se para cada operação realizada soma-se um na distância então o método chama-se Distância de Edição Simples ou somente Distância de Edição, mas se são dados pesos para cada tipo de operação então o método é chamado de Distância de Edição Geral. No exemplo anterior, se a operação de transposição tivesse peso 3 então a distância entre

as palavras "artigo" e "aritg" teria valor igual a 4.

Algumas das funções para cálculo de distância mais comumente usadas estão descritas abaixo:

- **Distância de Levenshtein**: Permite a adição, remoção e substituição de caracteres. Todas as operações têm custo 1. Esta função pode ser descrita como "o menor número de operações de adição, remoção e substituição de caracteres que fazem os dois textos serem iguais. A distância é simétrica e atende a expressão $0 \leq d(x, y) \leq max(|x|, |y|)$.

- **Distância de Hamming**: Permite somente substituição de caracteres com custo 1. A distância é simétrica e finita quando $|x| = |y|$ e nesse caso atende a expressão $0 \leq d(x, y) \leq |x|$.

- **Distância de Episódio**: Permite somente adição de caracteres com custo 1. A distância não é simétrica e pode ser que não seja possível converter x em y, então $d(x, y)$ é $|y| - |x|$ ou infinito.

- **Distância da Maior Subsequência Comum**: Permite somente adição e remoção de caracteres, todos com custo 1. O nome desta distância está associado ao fato de que dado a maior subsequência comum entre os dois textos, o número de caracteres que não fazem parte desta subsequência resulta no valor da distância, então $0 \leq d(x, y) \leq |x| + |y|$.

Contudo, o modelo clássico de solução do problema de casamento aproximado de textos não resolve os problemas de recuperação textual atuais, pois a coletânea de textos onde deseja-se procurar uma informação é muita extensa. Além disso, os textos apresentam uma grande heterogeneidade (de formatos e de idioma por exemplo), o que resulta em uma alta variabilidade de tipos de erro. Exemplos deste aspecto são os artigos digitalizados, onde o programa OCR (*Optical Character Recognition*) erra na identificação dos caracteres de 7% a 16%, e um outro aspecto são as já mencionadas abreviações em nomes de veículos. Atualmente, um dos modelos adotados é o casamento aproximado de textos que trabalha no nível das palavras e não dos caracteres. As mesmas operações de adição, remoção, substituição e transposição são aplicadas em relação às palavras do texto, o que possibilita uma melhor precisão na busca de frases por exemplo. Desse modelo seguem alguns outros mais avançados que não tratam erros sintáticos entre frases, mas verificam a semântica. Dessa forma, dois textos são iguais se as ideias são as mesmas, não importando se algumas palavras estão adicionadas, omitidas, substituídas ou trocadas de posição.

Alguns dos algoritmos para resolver o problema de casamento aproximado de textos são: os que usam programação dinâmica, os baseados em autômatos, os que usam a computação paralela dos *bits*, os que usam filtragem, e os que utilizam uma função *kernel*.

Algoritmos com Programação Dinâmica

Algoritmos com programação dinâmica computam dinamicamente uma matriz de distância de edição entre dois textos X e Y, onde cada elemento $E_{i,j}$ da matriz representa o menor número de operações necessárias para tornar o segmento $X_{1..i}$ em $Y_{1..j}$. A Figura 2.5 ilustra o cálculo da distância entre os textos "artigo" e "aritg". Esta computação tem a seguinte propriedade:

$$
\begin{aligned}
&E_{i,0} = i \\
&E_{0,j} = j \\
&E_{i,j} = \begin{cases} E_{i-1,j-1} & \text{se } X_i = Y_j \\ 1 + \min(E_{i-1,j}, E_{i,j-1}, E_{i-1,j-1}) & \text{caso contrário} \end{cases}
\end{aligned}
$$

Quando o caractere X_i é diferente de Y_i são permitidos três tipos de operações: eliminar o caractere X_i e converter da melhor forma $X_{1..i-1}$ em $Y_{1..j}$; inserir Y_j no final de $X_{1..i}$ e converter da melhor forma $X_{1..i}$ em $Y_{1..j-1}$; ou colocar o mesmo caractere de Y_j em X_i e converter da melhor forma $X_{1..i-1}$ em $Y_{1..j-1}$. Para qualquer operação o custo é igual a 1 mais o custo do resto do processo.

O último elemento da matriz exibe a distância de edição entre X e Y e este processamento foi da ordem de $O(|X||Y|)$.

		a	r	t	i	g	o
	0	1	2	3	4	5	6
a	1	0	1	2	3	4	5
r	2	1	0	1	2	3	4
i	3	2	1	1	1	2	3
t	4	3	2	1	2	2	3
g	5	4	3	2	2	2	3

Figura 2.5: Algoritmo de programação dinâmica para computar a distância entre "artigo" e "aritg"

Algoritmos Baseados em Autômatos

Algoritmos baseados em autômatos utilizam o modelo de autômatos finitos não determinísticos e consideram um fator de erro k para construir a distância de edição. Na Figura 2.6 encontra-se um autômato que tolera até 2 erros ($k = 2$). Cada linha denota o número de erros vistos, por exemplo, a primeira linha não existem erros, na segunda ocorreu um erro, e assim por diante. Cada coluna representa um casamento entre o prefixo do padrão e do texto (considerando o erro), sendo o tamanho do prefixo dado pelo número da coluna. As setas horizontais representam que houve o casamento de mais um caractere

entre o padrão e o texto, aumentando o tamanho do prefixo entre os dois. As outras setas representam sempre a adição de um erro (setas diagonais e verticais): a inserção de um caractere no padrão é dado pela seta vertical (avança-se no texto e não no padrão), a substituição de um caractere do padrão é dado pela seta diagonal não tracejada (como o caractere é substituído ocorre um avanço no tamanho do prefixo), e a eliminação de um caractere do padrão é dado pela seta diagonal tracejada (aumenta-se o prefixo, pois um caractere deixa de existir, avançando somente no padrão). Os símbolos Σ e ε são as funções de transição de estado.

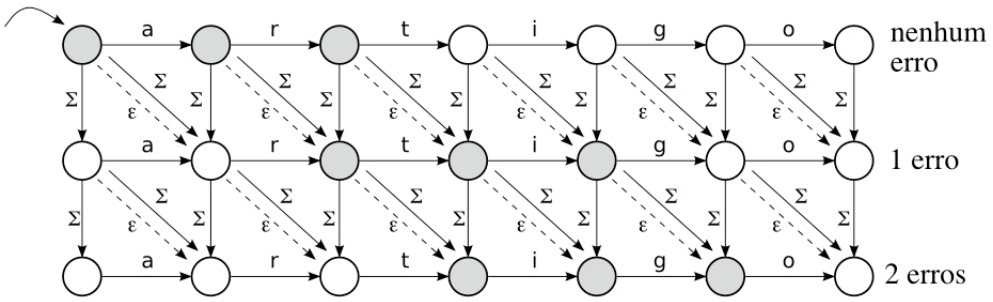


Figura 2.6: Autômato finito não determinístico para verificar o casamento com o texto padrão "artigo" considerando no máximo 2 erros. Os estados em cinza são os possíveis estados percorridos na verificação da palavra "aritg", neste caso as duas palavras são diferentes dado a tolerância de 2 erros.

Se na comparação dos textos chega-se no último estado de uma linha e a entrada terminou, então os textos são iguais (a menos dos erros tolerados). Esta abordagem é $O(n)$ no pior caso, onde n é o tamanho do padrão.

Algoritmos que Utilizam Computação Paralela em *Bits*

Algoritmos que exploram a capacidade de paralelismo fornecida pelo computador são outro tipo de abordagem para resolver o problema de casamento aproximado de textos. A idéia é "torna paralelo" um outro algoritmo usando *bits*. O algoritmo a ser paralelizado pode ser a matriz de distâncias, o autômato não determinístico ou outro método, desde que ele possa ser adaptado para esse paradigma. Esta técnica usa o paralelismo inerente às operações de *bits* nas palavras do computador. Dessa forma o número de operações que um algoritmo executa pode ser diminuído por um fator w, onde w é o número de *bits* de uma palavra (dependendo da arquitetura a palavra tem 32 ou 64 *bits*). O algoritmo precisa ser modelado de modo que uma palavra possa representar um estado ou um conjunto de valores, e a mudança de estado ou cálculo deve ser feito utilizando operações de "complemento", "and", "or", "xor" e "shift" em *bits*.

Algoritmos de Filtragem

A penúltima categoria de algoritmos são os que realizam uma filtragem do texto, rapidamente descartando partes que não irão casar. O filtro é baseado no fato que é mais fácil determinar que parte do texto não casa, do que determinar que partes são iguais. Se, por exemplo, as palavras "art" e "igo" não podem ser encontradas em um texto alvo, então a palavra "artigo" não poderá ser encontrada no texto com um erro menor que a distância de edição de cada segmento dela. O algoritmo de filtragem tem vantagens na procura de pequenos pedaços do texto padrão se o algoritmo de busca exata for mais rápido que o de uma busca aproximada. Posteriormente à eliminação das partes do texto que não poderão casar com o padrão, aplica-se um outro algoritmo para então verificar o casamento aproximado. Como a entrada foi reduzida o processamento do segundo algoritmo será mais rápido.

Algoritmos com Função *Kernel*

A última categoria de algoritmos usa uma função *kernel* para calcular a proximidade entre dois textos [31]. A ideia é comparar o texto padrão com o texto de entrada pelas suas subsequências. Os textos terão uma maior similaridade quanto maior o número de subsequências iguais. Os caracteres que compõem uma subsequência não precisam estar todos contíguos no texto alvo para que seja acusado que ela foi encontrada, neste caso usa-se um peso para determinar o grau de contiguidade. A subsequência "sem", por exemplo, está presente na palavra "**sem**pre" e na palavra "**s**ab**em**", mas com pesos diferentes.

Neste algoritmo, cada texto é mapeado em um conjunto de características formando um "espaço de características". Cada subsequência existente fornece uma dimensão neste espaço e o valor desta coordenada depende de quão frequente e compacto é aquela subsequência no texto. Para subsequências não contíguas utiliza-se um fator de decaimento $\lambda \in (0, 1)$ que é usado para "pesar" ou medir a presença de uma característica no texto. Para comparar dois textos usa-se uma função *kernel* que é uma função que calcula o produto interno entre os dois espaços de características.

A função kernel definida por Huma Lodhi *et* al. [31] é dada por:

$$
\begin{aligned}
&K'_0(s,t) = 1, \text{ para qualquer } s \text{ e } t;\\
&K'_i(s,t) = 0, \text{ se } min(|s|,|t|) < i;\\
&K_i(s,t) = 0, \text{ se } min(|s|,|t|) < i;\\
&K'_i(sx,t) = \lambda K'_i(s,t) + \textstyle\sum_{j:t_j=x} K'_{i-1}(s, t[1:j-1])\lambda^{|t|-j+2};\\
&K_n(sx,t) = K_n(s,t) + \textstyle\sum_{j:t_j=x} K'_{n-1}(s, t[1:j-1])\lambda^{2};
\end{aligned}
$$

A Figura 2.7 ilustra um exemplo do uso da função *kernel*. Nela é considerado as palavras "cat" e "car" e um tamanho de subsequência igual a 2. Isto gera um espaço de

características de tamanho 5. Os valores de cada característica foi calculado de acordo com a função definida acima. Dessa forma para encontrar a semelhança entre as palavras "cat" e "car" realiza-se um produto interno dos dois vetores de características, portanto $K_2(cat, car) = \lambda^4$, pois somente a coordenada referente a subsequência "ca" possui um valor diferente de zero para ambas as palavras. Este resultado não está normalizado, para isso deve-se obter $K_2(cat, cat) = K_2(car, car) = 2\lambda^4 + \lambda^6$ e normalizar dividindo λ^4 por $2\lambda^4 + \lambda^6$. De modo geral, a semelhança normalizada de dois textos s e t é dado por:

$$K_2^*(s,t) = \frac{K_2(s,t)}{\sqrt{K_2(s,s)K_2(t,t)}};$$

Os experimentos realizados por Huma Lodhi *et al.* mostram que este algoritmo pode ser uma alternativa eficiente para casamento aproximado de textos. O problema encontra-se no tempo necessário para computar a função *kernel* e na definição do tamanho da subsequência e o valor de λ, que são parâmetros a serem definidos de acordo com a aplicação e características dos textos comparados.

	ca	ct	at	cr	ar
$\phi(cat)$	λ^2	λ^3	λ^2	0	0
$\phi(cat)$	λ^2	0	0	λ^3	λ^2

Figura 2.7: Dado subsequências de tamanho 2, as palavras "cat" e "car" possuem vetores de características definidos por $\phi(cat)$ e $\phi(car)$. O cálculo do valor da coordenada utiliza a função *kernel* definida

2.7 Extração de Componentes em uma Referência Bibliográfica

A informação representada por uma referência bibliográfica pode ser dividida em vários componentes, tais como, autor, título, veículo, ano de publicação, acrônimo do veículo, volume, páginas, entre outros. Métodos de extração de componentes em referências bibliográficas são ferramentas que conseguem isolar um ou mais componentes dentro de uma referência em formato texto. Supõe-se que este texto não está estruturado de modo a permitir um processo trivial de extração, na verdade ele está estruturado de acordo com um formato para leitura humana. Como a forma de se apresentar uma referência bibliográfica pode variar muito, o método de extração de componentes deve acompanhar esta complexidade.

A bibliografia relacionada com a extração de componentes de uma referência é relativamente recente. Em 2006, Fuchun Peng e Andrew McCallum publicaram um artigo que

trata da extração de componentes em citações [39]. Para essa tarefa eles usam campos condicionais aleatórios (CRF), que são grafos não-direcionados onde são realizados treinos para maximizar uma probabilidade condicional. A função de probabilidade condicional usa uma função de características que pode medir qualquer aspecto de uma transição de estados. Cada estado é uma associação das partes da citação com o tipo de informação que ele está representando, por exemplo, definir que um termo é parte do nome de um autor e o termo seguinte é parte do título. Esta técnica permite utilizar características arbitrárias e realiza inferências na junção de sequências para formar completamente o componente. Eles conseguiram um desempenho superior na época em relação aos outros métodos de extração de informação.

Uma outra abordagem foi a proposta por Jun Xu e Yalou Huang em 2007 [49]. Eles utilizaram uma máquina de suporte vetorial (SVM) para extrair do texto um acrônimo e propor sua expansão. Primeiramente é identificado o acrônimo usando regras heurísticas, depois usando os textos ao redor do acrônimo procura-se determinar qual a mais provável expansão para ele. Na determinação do nome expandido usa-se o SVM que precisa ser previamente treinado com um conjunto dos dados. Nos experimentos foi obtida uma precisão acima de 89%.

Em 2007, Altigran e outros autores propuseram o método FLUX-CiM [18] que realiza a extração de componentes de citações de modo flexível e não-supervisionado. Este método utiliza uma base de conhecimento (*knowledge-base*) criada automaticamente a partir de uma amostragem dos dados e seu aprendizado não requer uma fase de treinamento. FLUX-CiM inicia seu processo quebrando o texto da referência em blocos. Cada bloco é uma sequência de caracteres sem qualquer p-delimitador (o p-delimitador é qualquer caractere que não seja letra nem número). Usando a base de conhecimento, o método tenta associar cada bloco com um tipo de campo bibliográfico, por exemplo, o bloco com o termo "proceedings" claramente não é parte do campo título, e o bloco com o termo "Halloween" é uma típica ocorrência de um título. Com certeza muito blocos terão ambiguidades. No passo seguinte os blocos que não foram associados a nenhum campo são analisados e associados de acordo com sua vizinhança, por exemplo, um bloco não associado entre dois blocos associados com o campo autor receberá também esta associação ao campo autor. Finalmente os blocos contíguos e associados ao mesmo campo são unidos para constituir aquele campo. Os experimentos realizados mostraram uma precisão média superior a 90%, sendo que 82% das citações tiveram seus campos extraídos perfeitamente. Em 2009, um outro artigo dos mesmos autores apresenta novamente o método FLUX-CiM com algumas melhorias e novos experimentos [19]. Agora a base de conhecimento é realimentada com os resultados da extração o que melhorou a qualidade dos resultados. Na parte de experimentos, FLUX-CiM foi comparado com o método CRF [39] e demonstrou-se que sem qualquer intervenção do usuário para criar o

conjunto de treino, FLUX-CiM tem uma qualidade melhor de extração do que CRF.

2.8 Algoritmos de Agrupamento

Algoritmos de agrupamento (ou *Cluster*) são procedimentos que organizam um conjunto de informações dentro de grupos baseando-se na similaridade entre os membros [1, 50]. Portanto, uma informação (normalmente descrita como um vetor de características ou um ponto no espaço) dentro de um grupo é mais similar aos outros membros do mesmo grupo e é dissimilar aos elementos dos outros agrupamentos. A Figura 2.8 mostra um exemplo de agrupamento. Nesse exemplo, cada informação é representada por dois valores, o que possibilita que os dados sejam mapeados no plano. Nesta figura é clara a divisão dos dados em dois grupos.

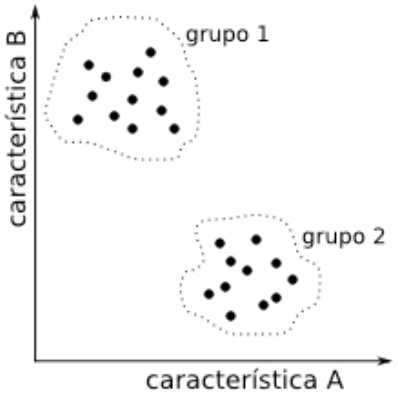


Figura 2.8: Dados mapeados no espaço 2D e a formação dos agrupamentos.

Existem duas importantes classes de agrupamento: os métodos supervisionados e os métodos não supervisionados. No método supervisionado existe um conjunto de rótulos (ou seja, um descritor de um padrão), pelos quais os dados serão classificados e agrupados. Neste tipo de abordagem é necessário conhecer de antemão os tipos de grupos que irão se formar. O que ocorre em muitos casos é a presença de alguns dados que são dissimilares a qualquer um dos rótulos, então um novo rótulo deve ser criado e o processo refeito. Normalmente usa-se um conjunto de rótulos para analisar as classes de dados presentes e criar os rótulos necessários para o agrupamento, como se fosse uma etapa de treinamento. No método não supervisionado não existem rótulos, ou seja, padrões predefinidos para realizar o agrupamento. Os dados são agrupados de acordo com alguma regra de similaridade e o número de grupos distintos emerge do processamento.

Algoritmos de agrupamento são largamente utilizados para analisar ou visualizar padrões em um conjunto de dados. De acordo com o tipo de aplicação, os padrões ob-

servados podem ser agrupados, podem ser utilizados para uma tomada de decisão ou aprendizagem de um agente de *software*, ou ainda serão úteis para uma visualização geral sobre uma extensa base de dados, já que o número de classes tende a ser bem menor que o número de dados. As mais diversas áreas da computação utilizam técnicas de agrupamento, tais como, aprendizagem de máquina, recuperação de documentos, segmentação de imagens, e classificação de padrões. Para muitos tipos de problemas, o entendimento sobre os dados é limitado, impossibilitando uma análise direta ou por modelos estatísticos. Nestas condições, os algoritmos de agrupamento são uma alternativa adequada para explorar a estrutura e relação entre os dados.

Abaixo serão descritos os principais algoritmos de agrupamento de acordo com Rui Xu e Donald Wunsch II [50], além do conceito de medida de distância e similaridade que é utilizado por muitos deles:

Medidas de Distância e Similaridade

Uma abordagem natural quando se compara dois objetos é analisar o quão parecido eles são, ou o quão diferentes. Desse raciocínio originou a idéia de funções que possam medir a dissimilaridade (distância) ou similaridade entre dois objetos. Todo dado é descrito por um conjunto de características, normalmente representado como um vetor multidimensional. O valor de uma característica pode ser qualitativo ou quantitativo, determinando o mecanismo de medida de similaridade/distância. A função de distância ou dissimilaridade para um conjunto de dados X é definida para satisfazer as seguintes condições :

(1) Simetria: $D(x_i, x_j) = D(x_j, x_i)$.

(2) Positividade: $D(x_i, x_j) \geq 0$ para todo x_i e x_j.

(3) Desigualdade Triangular: $D(x_i, x_j) \leq D(x_i, x_k) + D(x_k, x_j)$ para todo x_i,x_j e x_k.

(4) Reflexividade: $D(x_i, x_j) = 0$ se e somente se $x_i = x_j$.

Na verdade somente as condições (1) e (2) precisam ser satisfeitas, mas se as condições (3) e (4) forem também atendidas então esta função é chamada de "métrica". No outro extremo, a função de similaridade é definida para satisfazer as seguintes condições:

(1) Simetria: $S(x_i, x_j) = S(x_j, x_i)$.

(2) Positividade: $0 \leq S(x_i, x_j) \leq 1$ para todo x_i e x_j.

(3) $S(x_i, x_j)S(x_j, x_k) \leq [S(x_i, x_j) + S(x_j, x_k)]S(x_i, x_k)$.

(4) $S(x_i, x_j) = 1$ se e somente se $x_i = x_j$.

Novamente, a função de similaridade precisa satisfazer somente as condições (1) e (2), mas se as condições (3) e (4) forem também atendidas então esta função é chamada de "métrica de similaridade". A escolha de qual função usar ou se as duas são usadas depende da aplicação, pois de acordo com os tipos de dados uma métrica pode ser mais fácil de desenvolver do que outra. Normalmente, funções de distância são usadas para medir características contínuas, enquanto que as funções de similaridade são mais usadas quando as variáveis são analisadas qualitativamente. Abaixo esta relacionado dois tipos de medidas tipicamente utilizadas para características contínuas:

- Distância Euclidiana: $D_{ij} = \left(\sum_{l=1}^{d} |x_{il} - x_{jl}|^2\right)^{\frac{1}{2}}$

- Cosseno de Similaridade: $S_{ij} = cos\alpha = \frac{x_i^T x_j}{||x_i||||x_j||}$

Para um conjunto de dados de entrada de tamanho N, pode-se criar uma matriz de proximidade NxN onde cada elemento i, j da matriz armazena o valor da função de distância ou de similaridade entre o i-ésimo e o j-ésimo objeto. Esta é uma das formas de se apresentar a proximidade entre os objetos, mas cada tipo de algoritmo que utiliza as métricas de distância/similaridade faz um uso próprio das funções.

Agrupamento Hierárquico

Nos algoritmos de agrupamento hierárquico as informações são organizadas de acordo com a matriz de proximidade construída. Esta estrutura é projetada na forma de uma árvore, com nó raiz e nós folhas. O nó raiz representa todo o conjunto de dados e os nós folhas representam os objetos individualmente. Já os nós intermediários representam a proximidade entre um par objeto-objeto, grupo-grupo, ou objeto-grupo. Um resultado de agrupamento pode ser obtido cortando a árvore em diferentes níveis. Existem dois tipos de estratégia que operam sobre uma árvore do agrupamento hierárquico:

- **Agrupamento Aglomerativo**: Inicialmente tem-se N grupos, cada um com exatamente um objeto. Então realiza-se sucessivas operações mesclando os objetos até que todos estejam no mesmo grupo.

- **Agrupamento Divisivo**: Inicialmente todos os objetos estão em um mesmo grupo. Então realiza-se sucessivas divisões até que todos os grupos tenham um objeto.

Na prática, o método divisivo não é muito utilizado, pois em um grupo com N objetos existem $2^{N-1} - 1$ possibilidades de divisão, tornando o processo muito dispendioso. Para métodos aglomerativos o processo pode ser descrito da seguinte forma:

Passo 1: Iniciar com N grupo, cada um com um único objeto. Calcular a matriz de proximidade para estes N grupos.

Passo 2: Encontre a distância mínima através da equação:

$$D(C_i, C_j) = minD(C_m, C_l) \text{ , para } 1 \leq m, l \leq N \text{ e } m \neq l,$$

onde D é uma função de distância (dissimilaridade) para criar um novo grupo combinando C_i e C_j.

Passo 3: Atualizar a matriz de proximidade, pois os grupos foram combinados.

Passo 4: Repetir os passos 2 e 3 até todos os objetos estarem no mesmo grupo.

Na união de dois grupos pode-se usar a técnica de ligação simples ou ligação completa. A ligação simples calcula a distância entre dois grupos baseando-se nos dois objetos mais próximos (um de cada grupo), sendo também conhecida por método do vizinho mais próximo. De maneira oposta, a ligação completa calcula a distância usando os dois objetos, de grupos diferentes, mais distantes.

O algoritmo de agrupamento hierárquico é considerado sem robustez e sensível a ruídos. Após um objeto ser incluído em um grupo essa associação não é mais revista, impossibilitando a correção de uma falha de classificação anterior.

Agrupamento Baseado em Erro Quadrático

Este tipo de agrupamento utiliza a função de erro quadrático como critério de proximidade. Nos problemas abordados dessa maneira tem-se um conjunto de objetos x_j, com $j = 1.....N$, e o intuito é organizá-los em K grupos $C = \{C_1, ..., C_k\}$. O objetivo é minimizar a soma dos erros quadráticos entre os objetos dentro de um mesmo grupo. Um dos algoritmos mais conhecidos para agrupamento baseado em erro quadrático é o "K-means", que pode ser descrito da seguinte forma:

Passo 1: Inicializar K partições aleatoriamente ou se baseando em algum conhecimento prévio sobre os dados. Calcular para os grupos a matriz de protótipos $M = [m_1, ..., m_k]$. A matriz de protótipos armazena um representante de cada grupo que é, na maioria das vezes, o centróide dos objetos.

Passo 2: Associar cada objeto ao grupo C_w mais próximo:

$$x_j \in C_w \text{ se } ||x_j - m_w|| < ||x_j - m_i||, \text{ para } j = 1, ..., N, \; i \neq w, \text{ e } i = 1, ..., K.$$

Passo 3: Recalcular a matriz de protótipos baseado nas novas partições.

Passo 4: Repetir os passos 2 e 3 até que não ocorram mudanças nos grupos.

Alguns problemas do *K-means* são: sensível a ruídos, não garante a convergência para um ótimo global, e não existe um método geral para se definir as partições iniciais, nem o número de grupos.

Agrupamento Baseado em Mistura de Densidades

Esse tipo de algoritmo usa uma abordagem probabilística, assumindo que os dados são gerados por uma distribuição de probabilidades. Se a função de distribuição dos dados é conhecida, então agrupá-los é equivalente a encontrar os parâmetros do modelo probabilístico. Supõe-se uma probabilidade a priori (probabilidade de "mistura") $P(C_i)$ para os grupos $C_i, i = 1, .., K$, e a densidade de probabilidade condicional $p(x|C_i, \theta_i)$ (densidade de componente), onde θ_i é um vetor de parâmetros cujos valores não são conhecidos. A probabilidade de mistura de densidades é expressa por:

$$p(x|\theta) = \sum_{i=1}^{K} p(x|C_i, \theta_i)P(C_i)$$

onde $\theta_i = (\theta_1, ..., \theta_K)$, e $\sum_{i=1}^{K} P(C_i) = 1$. Assim que o valor dos parâmetros do vetor θ é definido, a probabilidade a posteriori pode ser calcula com o teorema de Bayes. As maiores críticas a este algoritmo é sua sensibilidade à variação dos parâmetros iniciais, sua possibilidade de convergência para ótimos locais e uma velocidade de convergência muita baixa.

Agrupamento Baseado em Teoria dos Grafos

Neste tipo de agrupamento os dados são associados aos nós de um grafo e as arestas representam a proximidade entre os objetos. Se a matriz de distâncias é definida por

$$D_{ij} = \begin{cases} 1 & \text{se } D(x_i, x_j) < d_0 \\ 0 & \text{caso contrário} \end{cases}$$

onde d_0 é um valor de limiar, o grafo é chamado de "grafo de limiar não ponderado". Após a modelagem através de um grafo pode-se aplicar um outro algoritmo para agrupar, como por exemplo o agrupamento hierárquico. Neste caso a técnica de ligação simples pode ser feita procurando subgrafos conectados maximalmente (componentes) e a ligação completa pode ser feita procurando a máxima *clique* no grafo.

Agrupamento Baseado em Técnicas de Busca Combinatorial

Nesta abordagem procura-se o ótimo global (ou uma aproximação para ele) resolvendo um problema de otimização combinatorial, que costuma ser NP-difícil. O agrupamento pode ser visto como um tipo de problema combinatorial: dado um conjunto de dados $x_j, j = 1, ..., N$, o algoritmo de agrupamento organiza-os em K grupos $\{C_1, ..., C_K\}$ que otimizam alguma função objetivo. A seguinte fórmula define os K grupos para N dados:

$$P(N, K) = \frac{1}{K!} \sum_{m=1}^{K} (-1)^{k-m} C_K^m m^N$$

Mesmo para valores pequenos de N e K, a computação já se mostra lenta. Portanto são utilizados outras técnicas para buscar o ótimo global, tais como, busca Tabu, algoritmos genéticos e outros algoritmos baseados em computação natural.

Agrupamento *Fuzzy*

Nos métodos anteriores, os objetos estão associados a um único grupo. No agrupamento *fuzzy* essa restrição é relaxada, e o objeto pode pertencer a mais de um grupo, mas para cada um é definido um grau de participação. Isto ajuda a resolver o problema de grupos com problema de separação e ambiguidades. Um dos mais populares algoritmos para agrupamento *fuzzy* é o FCM, que encontra c grupos *fuzzy* para um conjunto de objetos $x_j, j = 1, .., N$ minimizando a seguinte função custo:

$$J(U, M) = \sum_{i=1}^{c} \sum_{j=1}^{N} (u_{i,j})^m D_{ij}$$

onde

$U = [u_{i,j}]_{c \times N}$	a matriz U armazena a informação dos grupos *fuzzy* e $u_{i,j} \in [0, 1]$ é o grau de participação do j-ésimo objeto no i-ésimo grupo;
$M = [m_1, ..., m_c]$	é a matriz de protótipos (centróides) dos grupos;
$m \in [1, \infty)$	é um parâmetro da lógica *fuzzy* e normalmente vale 2;
$D_{ij} = D(x_j, m_i)$	é a medida de distância entre x_j e m_i.

FCM utiliza normalmente a distância Euclidiana e o seu algoritmo pode ser descrito da seguinte forma:

Passo 1: Selecionar valores apropriados para m, c e um pequeno valor ε. Inicializar a matriz de protótipos M aleatoriamente. Iniciar uma variável de contagem $t = 0$.

Passo 2: Calcular (em $t = 0$) ou atualizar (em $t > 0$) a matriz U fazendo,

$$u_{ij}^{(t+1)} = \frac{1}{\sum_{l=1}^{c}(\frac{D_{lj}}{D_{ij}})^{\frac{1}{(1-m)}}}, \text{ para } i = 1, ..., c \text{ e } j = 1, ..., N.$$

Passo 3: Atualiza a matriz de protótipos fazendo,

$$m_i^{(t+1)} = \frac{\sum_{j=1}^{N}(u_{ij}^{t+1})^m x_j}{\sum_{j=1}^{N}(u_{ij}^{t+1})^m}, \text{ para } i = 1, .., c.$$

Passo 4: Repetir os passos 2 e 3 até $||M^{(t+1)} - M^t|| < \varepsilon$.

Existem numerosas variações de FCM e de outros algoritmos de agrupamento *fuzzy*. Um aspecto negativo do algoritmo, e que também ocorrem em outras abordagens de agrupamento, é a sensibilidade a ruídos e a dificuldade de determinar os grupos iniciais.

Agrupamento Baseado em Redes Neurais

Um dos métodos dominantes para agrupamento baseado em rede neural é o SOFM, que é uma rede neural competitiva, onde neurônios ativos reforçam as ligações com seus vizinhos em determinadas regiões e suprime as atividades de outros neurônios. O objetivo desse método é utilizar uma malha em duas dimensões e representar nela os protótipos dos padrões de entrada. Os neurônios são as unidades da malha e são conectados aos seus vizinhos ao redor. Desses neurônios serão definidos alguns para serem os protótipos e eles farão com que a malha se molde para se ajustar à entrada. Um padrão de entrada será mais próximo de um dos protótipos da malha fazendo que os neurônios adjacentes àquele protótipo se aproximem e os de outros protótipos se afastem. Seu algoritmo pode ser descrito da seguinte forma:

Passo 1: Definir a topologia de SOFM e inicializar os protótipos $m_i, i = 1, ..., K$ aleatoriamente.

Passo 2: Apresentar um padrão x para a rede. Escolher o nó J que é mais próximo de x: $J = arg_j min(||x - m_j||)$.

Passo 3: Atualizar os protótipos

$$m_i(t+1) = m_i(t) + h_{ci}(t)[x - m_i(t)],$$

onde t representa o instante atual, $t+1$ representa o valor para a próxima iteração, e $h_{ci}(t)$ é uma função de vizinhança definida por

$$h_{ci}(t) = \alpha(t) exp(\frac{|||r_c - r_i||^2}{2\sigma^2(t)}),$$

onde $\alpha(t)$ (taxa de aprendizado) e $\sigma(t)$ são duas funções monotonicamente decrescentes, r representa a posição dos neurônios, i é o neurônio protótipo, e c são os neurônios ligados aquele protótipo i. Uma alternativa para a função $h_{ci}(t)$ é:

$$h_{ci}(t) = \begin{cases} \alpha(t) & \text{se o nó c faz parte da vizinhança do protótipo vencedor J} \\ 0 & \text{caso contrário} \end{cases}$$

Passo 4: Repetir os passos 2 e 3 até a posição dos neurônios não se alterar além de um certo ε.

Um problema recorrente e presente nesse tipo de algoritmo é a necessidade de se definir os protótipos e o tamanho da malha, isto é, o número de agrupamentos, mesmo isto sendo uma informação desconhecida. O SOFM também pode ser integrado a outros algoritmos, como o *k-means* e o agrupamento hierárquico, para deixá-los mais rápidos e efetivos.

Agrupamento Baseado em *Kernel*

Neste tipo de agrupamento os padrões são mapeados em um espaço de características complexo e não separável linearmente, porém eles podem ser separados através de uma transformação não linear. O objetivo do algoritmo é encontrar K centros para minimizar a distância entre cada padrão $x_j, j = 1, .., N$ (mapeados por uma função ϕ) e seus centros mais próximos:

$$\begin{gathered} ||\phi(x) - m_l||^2 \\ = ||\phi(x) - \textstyle\sum_{j=1}^{N} \tau_{lj}\phi(x_j)||^2 \\ = k(x,x) - 2\textstyle\sum_{j=1}^{N} \tau_{lj}k(x,x_j) + \sum_{i,j=1}^{N} \tau_{li}\tau_{lj}k(x_i,x_j), \end{gathered}$$

onde m_l é o centro do grupo l e $k(x, x_j) = \phi(x).\phi(x_j)$ é a função *kernel*. O algoritmo pode ser formulado da seguinte forma:

Passo 1: Inicializar os centros $m_l, l = 1, ..., K$ com o primeiro padrão i observado.

Passo 2: Olhar um novo padrão x_{i+1} e calcular $C_{(i+1)h}$ da seguinte forma:

$$C_{jl} = \begin{cases} 1 & \text{se } x_j \text{ está no grupo } l \\ 0 & \text{caso contrário} \end{cases}$$

Passo 3: Atualizar os centros m_h cujo $C_{(i+1)h}$ é 1

$$m'_h = m_h + \xi(\phi(x_{i+1}) - m_h),$$

onde $\xi = \frac{C_{(i+1)h}}{\sum_{j=1}^{i+1} C_{jh}}$.

Passo 4: Adapte os coeficientes τ_{hj} para cada $\phi(x_j)$:

$$\tau'_{hj} = \begin{cases} \tau_{hj}(1-\xi) & \text{para } j \neq i+1 \\ \xi & \text{para } j = i+1 \end{cases}$$

Passo 5: Repetir os passos 2 a 4 até convergir.

Assim como todas as abordagens mostradas, o agrupamento baseado em *kernel* possui seus problemas. Um deles é a complexidade na computação do cálculo da não linearidade para grandes conjuntos de dados.

Capítulo 3

Ferramentas de Extração e Ordenação dos Veículos

Neste capítulo são descritas as características e funções das ferramentas utilizadas para extrair os veículos de publicação e ordená-los. No capítulo seguinte serão mostrados os resultados obtidos.

A Figura 3.1 ilustra o processo usado para a extração e identificação de um veículo de publicação, mostrando como os dados fluem de uma ferramenta para outra.

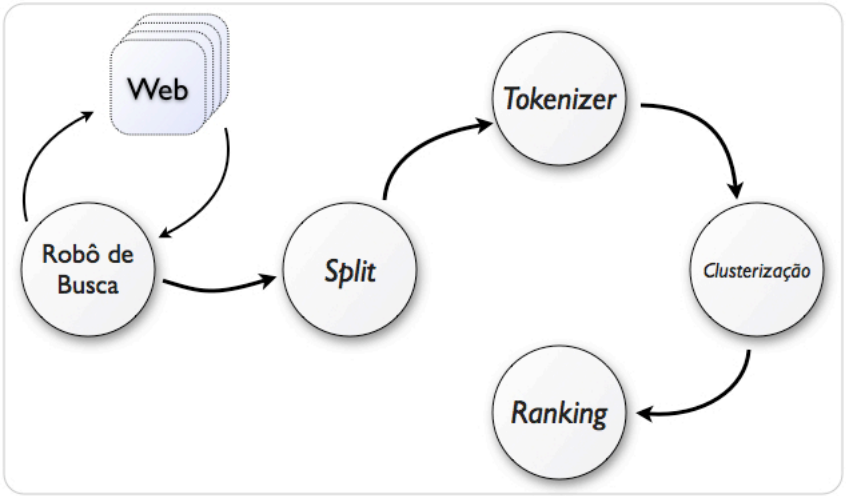


Figura 3.1: Fluxo de trabalho para a construção do *ranking*

Primeiramente o robô de busca acessa as páginas Web dos professores de uma universidade e obtém arquivos HTML, PDF ou PS contendo publicações. Como cada arquivo possui várias referências, a ferramenta *Split* separa cada uma delas para o processamento individual. Em seguida a ferramenta *Tokenizer* extrai de cada referência seu veículo

de publicação. Todos os veículos extraídos são agrupados pelo clusterizador e então é realizado o *ranking* das publicações baseado em um *ranking* já existente de universidades.

3.1 Robô de Busca na Web

O robô de busca na Web é uma ferramenta, escrita na linguagem Python, responsável por obter da página dos professores de uma universidade suas listas de publicações. Existem três pontos identificados como iniciais para o robô de busca começar o seu trabalho, a saber:

- Página inicial da universidade: por exemplo, *http://www.ic.unicamp.br* que é a página inicial da faculdade de Ciência da Computação da Universidade de Campinas, para que o robô avance através dos *links* e encontre os *curricula*.
- Página com a lista de pesquisadores: por exemplo, iniciar o robô na página com a lista de docentes da faculdade de Ciência da Computação da UNICAMP, neste caso o robô avança sobre cada *link* nos nomes dos docentes e dentro deles tenta encontrar o *curriculum*.
- Página do *Curriculum*: passar diretamente à página do *curriculum* de cada docente.

O primeiro ponto de partida torna muito simples a tarefa de coletar a entrada da ferramenta, mas tem a desvantagem de aumentar a complexidade do algoritmo, pois buscar as páginas dos professores torna-se uma tarefa muito árdua pela multiplicidade de *links* existentes na página principal de uma faculdade. A última opção foi descartada pois o trabalho de recuperação torna-se manual, inviabilizando o processamento sobre um grande número de universidades, já que é demandado muito tempo sobre cada professor, além de descaracterizar o robô de busca. Portanto a segunda opção foi a escolhida. Neste caso, a entrada da ferramenta é uma lista de docentes de uma faculdade, e esta é a alternativa que melhor alia um algoritmo não muito complexo com um tempo relativamente rápido para preparar a entrada.

Para se preparar a entrada do robô cria-se um arquivo em que a primeira linha é o endereço de uma página Web cujo conteúdo é uma lista de professores de uma faculdade, e as linhas seguintes do arquivo são os nomes dos docentes tal como aparecem na página. Para coletar estes dados, simplesmente procurava-se a lista de docentes na página da instituição e a copiava para o arquivo, colocando em primeiro lugar o endereço no qual ela foi obtida. Vale ressaltar que uma premissa para esta ferramenta é que o nome do professor contém o *link* para sua página pessoal. Para a maioria das universidades utilizadas isto foi verdade, mas para um grupo de aproximadamente 14% do total, o nome do professor não continha um *link*. Para estes casos sempre existia uma alternativa,

pois o *link* para a página pessoal do docente se apresentava em textos chaves, tais como "personal homepage", "website", entre outros, assim bastou fazer uma modificação no robô para ele trabalhar adequadamente para estes casos específicos.

De posse desse arquivo dispara-se o robô. Ele acessa a página Web fornecida e percorre os links verificando quais deles coincidem com algum dos nomes passados. Para cada nome que coincidir, o robô acessará o *link* e armazenará a página principal do docente. Então a página principal é analisada e todos os *links* que partem dela são armazenados desde que seu nome, ou o nome da página alvo, contenha uma das palavras chaves. Esta lista de palavras chaves é composta por termos como "publication", "papers","vitae", entre outras, e o nome não precisa ser exatamente igual, podendo conter caracteres antes ou depois da palavra chave, ou apresentá-la com variações de maiúsculo e minúsculo, pois a busca é feita utilizando-se expressões regulares o que permite informar a variabilidade da forma de escrita através de regras. Exemplo: um professor, chamado por exemplo Richard Smith, possui na sua página pessoal um *link* para suas publicações e outro para seu *curriculum* dentre outros *links* diversos. Neste caso o robô irá salvar a página principal (ex. *rsmith.html*), a página com as publicações, pois o nome do *link* ou o nome da página alvo possui a palavra chave (ex. *pubs.html*), e a página com o *curriculum* pelos mesmo critérios adotados para salvar a página de publicação (ex. *vitae.pdf*). Como no exemplo, o robô não salva apenas páginas HTML, ele também armazena arquivos PDF e PS.

Todas as páginas armazenadas de um professor ficam sob um diretório com o nome da universidade. Dentro deste diretório é comum existir inúmeras páginas salvas para um mesmo professor, portanto o próximo passo é analisar quais destes arquivos são relevantes, ou seja, contém publicações para serem extraídas. O processo é bastante simples, procura-se a frequência do sobrenome do professor no conteúdo dos arquivos HTML (no caso de PDF ou PS usa-se uma ferramenta externa para converter este tipo de arquivo em texto). O arquivo com maior frequência do sobrenome (dentre todos aqueles de um mesmo professor) é selecionado, pois espera-se que ele contenha as publicações. Quando é selecionado um arquivo texto que veio de uma conversão, na verdade seleciona-se o PDF ou PS original. Estes arquivos escolhidos são então copiados para outro diretório para separá-los.

Neste ponto do processamento temos um diretório por universidade e, dentro de cada um, vários arquivos HTML, PDF e/ou PS, mas somente um por professor. O próximo passo é separar as publicações que estão dentro de cada arquivo, pois o *Tokenizer* trabalha sobre um texto que descreve uma única publicação. Portanto na próxima seção será descrita a ferramenta *Split*, que separa uma a uma as publicações encontradas dentro de um mesmo arquivo.

3.1.1 Teste do Robô de busca

Para realizar os testes de busca do *curriculum* dos pesquisadores, recolheu-se um conjunto amostral de 341 elementos (dentre aproximadamente 3000 arquivos obtidos). Este valor foi escolhido para garantir um nível de confiança de 95% com um intervalo de confiança de 5%. Analisou-se a porcentagem de erros de busca, ou seja, a informação sobre as publicações do pesquisador estava em sua página pessoal, mas o arquivo obtido não continham estas informações. No total, 25% dos professores não tiveram suas publicações obtidas em suas páginas pessoais.

3.2 Ferramenta de Separação de Publicações

Como visto na seção anterior, nem todos os arquivos de uma universidade são HTML podendo existir arquivos do tipo PDF ou PS. O primeiro passo é a conversão destes arquivos para o formato texto. Para realizar a conversão foram utilizadas ferramentas do Sistema Operacional e sobre os arquivos textos gerados usou-se um programa, criado em Python, para marcar o início de cada publicação. Em alguns casos, as ferramentas de conversão cometiam alguns erros o que exigia um processo semi-automático para marcar o início das publicações.

Logo após todos os arquivos PDF/PS da universidade terem sido convertidos, o programa *Split*, que foi também implementado em linguagem Python, irá iniciar seu trabalho e separar as publicações. Esta ferramenta conseguirá realizar esta tarefa com base nas *tags* HTML e nas marcações inseridas no texto convertido. Caso o arquivo contenha as marcas então o trabalho é bastante simples, resumindo-se a encontrar uma marca que indica o início de uma publicação e capturar o texto até antes da marca da próxima publicação. Estas buscas são feitas até todas as marcas terem sido processadas.

Caso o arquivo seja HTML então a ferramenta *Split* irá se basear nas *tags* para delimitar o texto de cada publicação. O primeiro passo é encontrar todas as ocorrências do sobrenome do autor. Vale lembrar que o *Tokenizer* também adota esta premissa de que o sobrenome do autor é conhecido. Para cada ponto do texto HTML onde o sobrenome do autor foi encontrado procura-se ao redor por *tags* que caracterizem uma divisão lógica de idéias, como por exemplo separar as publicações usando *tags* de tabelas (“<td>” e “</td>”) ou um separador “<p>”. O programa *Split* possui uma lista contendo vários tipos de *tags* que caracterizam divisões de conteúdo, e somente estas são utilizadas como separadoras de publicações. Alguns exemplos destas *t*ags são: “<li>”,“<p>”,“ ”, “<tr>”,“<td>”, “<dt>”, e “<ptitle>”. O texto da publicação é capturado utilizando-se as *tags* separadoras mais próximas do sobrenome do autor. Exemplo:

```
<a name="Articles in conference proceedings"> </a>
```

```
<h2> Articles in conference proceedings </h2>
<ul class="index">
<li> D. Burkett and D. Klein, "Two languages are better than
one (for syntactic parsing)," in <em>Proc. 2008 Conf. on
Empirical Methods in Natural Language Processing (EMNLP
'08)</em>, ACL Anthology, Stroudsburg, PA: Association for
Computational Linguistics, 2008, pp. 877-886.
</li><p>
```

Neste exemplo as *tags* "
" e "
" delimitam o texto da publicação e são usadas para separá-lo.

Um problema encontrado foi quanto ao uso da *tag* "
", pois ela pode caracterizar tanto uma divisão entre duas publicações distintas, como um pulo de linha dentro de uma mesma publicação. A ferramenta *Split* somente usa este separador para dividir as publicações se não foi encontrada nenhuma outra *tag* separadora. Esta abordagem funciona muito bem para a maioria dos casos, mas para algumas páginas a *tag* "
" é usada para separar duas publicações distintas, e existem outras *tags* separadoras próximas, servindo, por exemplo, para agrupar artigos em diferentes temas. Nesse caso ocorre uma falha, e *Split* extrai um texto muito maior, contendo duas ou mais publicações. Ocorre que durante o processamento da universidade são exibidos dados sobre o andamento da execução, tais como, nome do autor, quantas publicações foram extraídas, tamanho da maior publicação, e média do tamanho das publicações. Caso ocorra esta situação, para aquele autor serão extraídas poucas publicações e cada uma terá um tamanho anormal. Conforme estes dados vão aparecendo os nomes dos autores que apresentaram problemas são anotados manualmente. No final o processo é refeito, ou seja, *Split* é novamente executado para aquela universidade, mas agora é passado todos os nomes anotados para que o programa force a divisão entre publicações através do "
". Este procedimento mostrou-se eficaz na extração do texto da referência.

No momento que *Split* extrai uma publicação, a ferramenta *Tokenizer* é chamada para extrair o veículo da mesma. Portanto a saída do *Split* já é o resultado do processamento sobre cada referência. É dado abaixo um exemplo de como uma linha do arquivo de saída é estruturada:

Uma linha tem o seguinte formato:
`Venue ; Acro ; Year ; C/J/O? ; Author ; University ; FullRef` , onde

- **FullRef** é o texto completo da referência. Exemplo: Azriel Rosenfeld , John L. Pfaltz, Sequential Operations in Digital Picture Processing, Journal of the ACM (JACM), v.13 n.4, p.471-494, Oct. 1966.

- **University** é o nome da universidade de origem desta publicação. Exemplo: virginia (nome único e padronizado para representar "The University of Virginia").
- **Author** é o nome do autor da página de onde foi extraído a publicação. Exemplo: John L. Pfaltz
- **C/J/O?** é um código usado para marcar se uma publicação já foi identificada previamente como uma conferência (C), um periódico (*journal*) (J), ou se sabe que é outro tipo (O).
- **Year** é o ano extraído da publicação. Exemplo: 1966.
- **Acro** é o acrônimo extraído da publicação. Exemplo: jacm.
- **Venue** é o nome do veículo de publicação extraído. Exemplo: journal of the acm.

A seguir o resultado dos testes com a ferramenta *Split*, e na próxima seção será descrito o funcionamento do *Tokenizer*, que é responsável por encontrar no texto da publicação o nome do veículo, acrônimo e ano de publicação.

3.2.1 Teste do *Split*

Recolheu-se um conjunto amostral de 372 elementos para analisar os erros na segmentação das referências bibliográficas. Este valor foi escolhido para garantir um nível de confiança de 95% com um intervalo de confiança de 5%. Foi analisado a porcentagem de erro de segmentação (referências não segmentadas ou segmentadas parcialmente) e porcentagem de excesso, ou seja, trechos que não fazem parte de qualquer referência bibliográfica, mas que foram segmentados. No total, ocorreram 3% de erros e 7% de trechos de textos não relacionados com as referências. Esses 7% de trechos inválidos irão gerar nomes de veículos inválidos na fase de extração, e podem contribuir em outros erros (como no agrupamento).

3.3 Ferramenta de Extração do Veículo de Publicação

Esta ferramenta, chamada de *Tokenizer*, foi implementada em linguagem Python como uma biblioteca e pode ser adicionada facilmente a outros programas. Ela precisa de dois parâmetros de entrada: a referência em modo texto e uma parte do nome do pesquisador de cujo *curriculum* foi extraída esta referência. Exemplos:

- Referência: Azriel Rosenfeld , John L. Pfaltz, Sequential Operations in Digital Picture Processing, Journal of the ACM (JACM), v.13 n.4, p.471-494, Oct. 1966

- Nome: Pfaltz

Neste exemplo, a referência foi extraída do *curriculum* de John L. Pfaltz (University of Virginia), então a parte do nome usada é "Pfaltz", pois o sobrenome é uma chave que identifica o autor em suas publicações.

A maior dificuldade encontrada foi com relação à diversidade no modo de se escrever uma referência. Abaixo são apresentadas três formatos comuns de referências:

- **1**: David Maltz, Jibin Zhan, Geoffrey Xie, Hui Zhang, Gisli Hjalmtysson, Albert Greenberg, and Jennifer Rexford, Structure preserving anonymization of router configuration data. Proc. Internet Measurement Conference, October 2004.
- **2**: Extracting Information from Resolution Proof Trees. Luckham, D. C. and Nilsson, N. Artificial Intelligence, 2(1):27-54, 1971.
- **3**: Azriel Rosenfeld , John L. Pfaltz, Sequential Operations in Digital Picture Processing, Journal of the ACM (JACM), v.13 n.4, p. 471-494, Oct. 1966

Percebe-se uma alta variabilidade de formas de separação e estrutura de escrita dos nomes dos autores. No caso do nome, os três exemplos exibiram duas estruturas distintas. Nos casos 1 e 3 o sobrenome esteve como último termo da estrutura (por exemplo, o sobrenome *M*altz em "David Maltz"). Já no caso 2 os sobrenomes estavam sempre à frente do(s) primeiro(s) nome(s) (por exemplo, o sobrenome *L*uckham em "Luckham, D. C."). Uma outra variação é com relação à separação entre o penúltimo e último autor. Nos casos 1 e 2 os dois últimos autores estão separados entre si pela palavra *and*, mas no caso 3 somente a vírgula é usada para esta mesma finalidade. Existe também uma variação de ordem entre o título e autores da referência. Para as referências dos casos 1 e 3 os autores vêm em primeiro, e para a referência do caso 2 o título é o primeiro a ser escrito. Uma última análise quanto à variabilidade de formas de escrita das referências é a forma de separação entre o nome do veículo de publicação e os termos anteriores. O símbolo que marca a divisão entre o nome do veículo e os autores ou título é o ponto (caso 1) ou a vírgula (casos 2 e 3). Quando a vírgula é usada para esse fim surge um problema de ambiguidade, pois a vírgula também pode ser usada para separar autores e como estrutura linguística dentro do título.

Nas próximas subseções são descritas as premissas adotadas, o processo de identificação do veículo e em seguida é dado um exemplo do funcionamento do *Tokenizer*. Por fim são reportados os testes realizados.

3.3.1 Premissas

Para o seu correto funcionamento o programa *Tokenizer* adota algumas premissas, a saber:

- Todos os nomes de autores, dentro de uma mesma referência, estão escritos no mesmo formato. Isto foi adotado para facilitar a implementação, pois a informação do formato é usada para identificar os autores na referência. Não foi encontrado um contra-exemplo que mostre que dentro de uma referência os autores foram escritos em mais de um formato, mas mesmo assim isto pode ser possível;

- O nome de um autor deve ter uma quantidade pequena de nomes não abreviados (máximo três nomes), exemplo, "José Alberto Ribeiro" é um nome válido, mas "José Alberto Ribeiro da Silva" não é, isto porque nomes grandes são normalmente abreviados e para o algoritmo ficaria difícil distinguir um nome grande de um título pequeno. Esta abordagem é adequada para nomes americanos e ingleses, mas para nomes latinos podem ocorrer erros devidos ao tamanho dos nomes;

- Uma referência inicia pelos nomes dos autores ou pelo título e logo em seguida existe o nome do veículo. Por ser muito comum que as referências iniciem pelos autores ou pelo título, esta abordagem é bastante interessante, mas podem ocorrer casos em que uma outra disposição seja empregada, precisando ser tratada separadamente;

- As referências devem estar escritas em português ou inglês. Para o processamento do texto são identificadas as palavras presentes usando um dicionário, e alguns conectivos como "e" e "*and*" são usados para separar autores e título. Como o foco é nas instituições brasileiras, americanas e inglesas isto não é um empecilho.

3.3.2 Processamento

O processo usado para a extração do veículo de publicação utiliza análises numéricas e estatísticas, além de regras e métodos heurísticos, tal como visto na Seção 2.5. No texto de uma referência são identificadas várias expressões, como números, palavras, volume, edição, vírgulas, pontos, etc., e, analisando a posição e quantidade dessas expressões, pode-se segmentar trechos no texto. Após segmentar uma parte da referência (um autor, autores, título, etc) usa-se essa informação para as próximas segmentações. Este processo é análogo ao usado pelo FLUX-CiM (Seção 2.7), onde os blocos rotulados são utilizados para descobrir o tipo de campo bibliográfico dos blocos não rotulados. Um exemplo seria na segmentação do título numa referência dado que os autores já foram segmentados, neste caso se os autores estão distantes do início do texto então supõe-se que do início do texto até o início do segmento dos autores está compreendido o segmento do título. Logicamente esta é uma abordagem heurística e é baseada na premissa de que as referências iniciem com os autores ou com o título.

Na referência passada como parâmetro, o processamento inicia marcando-se as expressões regulares. Um exemplo de uma expressão pode ser o volume de um periódico

que é escrito na forma "volume #", onde "#" é um número qualquer. Mas esta não é a única forma regular para descrever um volume; outras variações são: "v.#", "vol.#" ou "vol #". Estas variações estão implementadas. Algumas expressões marcadas são: volume, edição, acrônimo entre parênteses, página, número ordinal, números, abreviações de uma letra, separadores (vírgula, ponto, ponto e vírgula e aspas), palavras, expressões que definem uma tese ou um relatório técnico e endereços Web.

Em seguida, nas palavras encontradas, usa-se dicionários (inglês Americano, inglês Britânico e Português) para identificá-las e tentar encontrar o idioma da referência. Se uma palavra não foi encontrada no dicionário então ela é candidata a ser classificada como uma abreviação. Isto somente ocorrerá se ela for início de qualquer outra palavra no idioma da referência e for seguida por um ponto no texto. Um exemplo dessa situação é a palavra "*trans*" que não é encontrada no dicionário em inglês, mas como ela inicia outras palavras como "*transactions*" ela é marcada como uma abreviação caso exista um ponto a sua frente.

O próximo passo é encontrar o nome do autor. Usando o sobrenome passado como parâmetro, realiza-se uma busca para encontrá-lo, e no ponto do texto onde ele é identificado faz-se uma varredura para a esquerda e para a direita com o intuito de identificar todo o nome do autor e de que forma ele está estruturado. No momento da varredura para a esquerda, a partir do sobrenome, se um separador é encontrado então significa que o(s) primeiro(s) nome(s) estão à frente do sobrenome, mas se outras palavras são encontradas então o sobrenome está escrito por último dentro do nome do autor. Exemplos destes dois tipos de estruturas são:

- **. Luckham, D. C.** - partindo de "Luckham" encontra-se um separador à esquerda então os primeiros nomes estão à direita.
- **, John L. Pfaltz,** - partindo de "Pfaltz" encontra-se uma palavra à esquerda então o nome está escrito em ordem.

Identificado o primeiro autor e o formato do nome faz-se novas varreduras para a esquerda e direita tentando encontrar outros autores. O critério de parada da busca é quando se encontra várias palavras seguidas, sem nenhum separador, o que caracterizaria um título ou nome de veículo, ou quando se encontra o último autor identificado pela palavra "*e*" ou "*and*".

Segundo características observadas, se a região dos autores está deslocada do início do texto da referência então as palavras que vêm antes desta região irão compor o título.

Do lado direito da referência procura-se o primeiro termo relacionado com o nome do veículo ou com a publicação, mas que com certeza não faz parte explicitamente do nome do veículo, são exemplos disto o volume, a edição, o ano e a cidade de publicação. Desta

forma é delimitada uma região entre o término dos autores e o início destes marcadores. Se o título estava no começo então esta região será o nome do veículo de publicação, mas em muitos outros casos essa região conterá o título seguido do nome do veículo cabendo ao algoritmo encontrar o ponto de divisão dos dois.

Algumas heurísticas para dividir a região encontrada entre título e nome de veículo são: verificar se existe um único separador que separa dois blocos de palavras; verificar se existe um certo número de palavras entre aspas seguido de um separador, caracterizando o título; e uso de palavras chave para identificar o local do veículo, tais como *proceedings*.

Caso estes processos falhem então tentam-se outros mais suscetíveis a erros como separar usando a vírgula, mesmo existindo mais de uma na região, o que indicaria que o título ou o nome do veículo contém vírgulas resultando numa possível divisão errada.

O nome do veículo encontrado é então limpo dos marcadores e devolvido pelo programa juntamente com o título encontrado e a lista de autores.

3.3.3 Exemplo de Segmentação do Nome do Veículo

A seguir um exemplo de segmentação do nome do veículo:

- Referência de entrada:

  Azriel Rosenfeld , John L. Pfaltz, Sequential Operations in Digital Picture Processing, Journal of the ACM (JACM), v.13 n.4, p.471-494, Oct. 1966

- Nome chave do autor:

  Pfaltz

- Marcando o volume:

  Azriel Rosenfeld , John L. Pfaltz, Sequential Operations in Digital Picture Processing, Journal of the ACM (JACM), **VO{v.13}** *n.4, p.471-494, Oct. 1966*

- Marcando a edição:

  Azriel Rosenfeld , John L. Pfaltz, Sequential Operations in Digital Picture Processing, Journal of the ACM (JACM), **VO{v.13} IS{n.4}** *, p.471-494, Oct. 1966*

- Marcando o acrônimo:

  Azriel Rosenfeld , John L. Pfaltz, Sequential Operations in Digital Picture Processing, Journal of the ACM **AC{(JACM)}** *,* **VO{v.13} IS{n.4}** *, p.471-494, Oct. 1966*

- Marcando as páginas:

  Azriel Rosenfeld , John L. Pfaltz, Sequential Operations in Digital Picture Processing, Journal of the ACM **AC{(JACM)}** *,* **VO{v.13} IS{n.4}** *,* **PP{p. 471-494}** *, Oct. 1966*

- Marcando os números:

  Azriel Rosenfeld , John L. Pfaltz, Sequential Operations in Digital Picture Processing, Journal of the ACM **AC{(JACM)}** *,* **VO{v.13} IS{n.4}** *,* **PP{p. 471-494}** *, Oct.* **NU{1966}**

- Marcando as abreviações (uma letra):

  Azriel Rosenfeld , John **AB{L.}** *Pfaltz, Sequential Operations in Digital Picture Processing, Journal of the ACM* **AC{(JACM)}** *,* **VO{v.13} IS{n.4}** *,* **PP{p. 471-494}** *, Oct.* **NU{1966}**

- Marcando os separadores:

  Azriel Rosenfeld **SP{,}** *John* **AB{L.}** *Pfaltz* **SP{,}** *Sequential Operations in Digital Picture Processing* **SP{,}** *Journal of the ACM* **AC{(JACM)} SP{,} VO{v.13} IS{n.4} SP{,} PP{p. 471-494} SP{,}** *Oct* **SP{.} NU{1966}**

- Marcando as palavras:

  UW{Azriel} UW{Rosenfeld} SP{,} EW{John} AB{L.} UW{Pfaltz} SP{,} EW{Sequential} EW{Operations} EW{in} EW{Digital} EW{Picture} EW{Processing} SP{,} EW{Journal} EW{of} EW{the} UW{ACM} AC{(JACM)} SP{,} VO{v.13} IS{n.4} SP{,} PP{p. 471-494} SP{,} UW{Oct} SP{.} NU{1966}

- Marcando novas abreviações (caso de *Oct* - abreviação de *October*):

  UW{Azriel} UW{Rosenfeld} SP{,} EW{John} AB{L.} UW{Pfaltz} SP{,} EW{Sequential} EW{Operations} EW{in} EW{Digital} EW{Picture} EW{Processing} SP{,} EW{Journal} EW{of} EW{the} UW{ACM} AC{(JACM)} SP{,} VO{v.13} IS{n.4} SP{,} PP{p. 471-494} SP{,} AB{Oct.} NU{1966}

- Encontrando o autor inicial:

UW{ACM} AC{(JACM)} SP{,} VO{v.13} IS{n.4} SP{,} PP{p. 471-494}
SP{,} AB{Oct.} NU{1966}

- Encontrando o autor inicial (varrendo as adjacências para compor o nome completo):

UW{Azriel} UW{Rosenfeld} SP{,} **EW{John} AB{L.} UW{Pfaltz}**
SP{,} EW{Sequential} EW{Operations} EW{in} EW{Digital}
EW{Picture} EW{Processing} SP{,} EW{Journal} EW{of} EW{the}
UW{ACM} AC{(JACM)} SP{,} VO{v.13} IS{n.4} SP{,} PP{p. 471-494}
SP{,} AB{Oct.} NU{1966}

- Encontrando os demais autores (varrendo a partir do autor principal):

UW{Azriel} UW{Rosenfeld} SP{,} EW{John} AB{L.} UW{Pfaltz}
SP{,} EW{Sequential} EW{Operations} EW{in} EW{Digital}
EW{Picture} EW{Processing} SP{,} EW{Journal} EW{of} EW{the}
UW{ACM} AC{(JACM)} SP{,} VO{v.13} IS{n.4} SP{,} PP{p. 471-494}
SP{,} AB{Oct.} NU{1966}

- Isolando o volume, edição, etc., do lado direito da referência:

UW{Azriel} UW{Rosenfeld} SP{,} EW{John} AB{L.} UW{Pfaltz}
SP{,} EW{Sequential} EW{Operations} EW{in} EW{Digital}
EW{Picture} EW{Processing} SP{,} EW{Journal} EW{of} EW{the}
UW{ACM} AC{(JACM)} **SP{,} VO{v.13} IS{n.4} SP{,}**
PP{p. 471-494}SP{,} AB{Oct.} NU{1966}

- O texto interno compreende o título e o veículo:

UW{Azriel} UW{Rosenfeld} SP{,} EW{John} AB{L.} UW{Pfaltz}
SP{,} EW{Sequential} EW{Operations} EW{in} EW{Digital}
EW{Picture} EW{Processing} SP{,} EW{Journal} EW{of} EW{the}
UW{ACM} AC{(JACM)} SP{,} VO{v.13} IS{n.4} SP{,} PP{p. 471-494}
SP{,} AB{Oct.} NU{1966}

- Encontrando o ponto de divisão (neste caso um separador dentre dois blocos de texto):

UW{Azriel} UW{Rosenfeld} SP{,} EW{John} AB{L.} UW{Pfaltz} **SP{,}**
EW{Sequential} EW{Operations} EW{in} EW{Digital} EW{Picture}
EW{Processing} SP{,} **EW{Journal} EW{of} EW{the} UW{ACM}**

AC{(JACM)} SP{,} NO{v.13} IS{n.4} SP{,} PP{p.471-494} SP{,} AB{Oct.} NU{1966}

- Título e Veículo foram segmentados:

UW{Azriel} UW{Rosenfeld} SP{,} EW{John} AB{L.} UW{Pfaltz} SP{,}

EW{Sequential} EW{Operations} EW{in}EW{Digital} EW{Picture}

EW{Processing} SP{,} EW{Journal} EW{of} EW{the} UW{ACM}

AC{(JACM)} SP{,} NO{v.13} IS{n.4} SP{,} PP{p.471-494} SP{,} AB{Oct.} NU{1966}

Para esta referência de entrada o *Tokenizer* consegue segmentar corretamente o veículo, o título, o acrônimo e os autores.

3.3.4 Teste do *Tokenizer*

Os primeiros testes foram realizados com referências extraídas manualmente da Web. Para isto, era escolhida aleatoriamente uma página de um professor, recolhia-se as referências manualmente e executava-se o *Tokenizer*. Os erros encontrados eram corrigidos e novas regras eram adicionadas para contemplar os casos em que a segmentação do nome do veículo não era correta. No apêndice A consta uma tabela com os testes realizados para 92 referências extraídas da página do professor Zohar Manna do departamento de Ciência da Computação da Universidade de Stanford.

Neste teste foi obtido 86,96% de acerto na segmentação do veículo de publicação, e esta taxa de acerto é dividida em 79,35% de acertos exatos, onde o nome do veículo foi extraído corretamente, e 7,61% de acertos parciais, onde o nome do veículo foi segmentado, mas algum pequeno texto faltou ou foi acrescentado ao nome do veículo. Nesta planilha, as linhas marcadas com "E" são os erros, as marcadas com "P" são os acertos parciais e as marcadas com "A" são os acertos exatos.

Com o robô de busca criado realizaram-se testes mais robustos utilizando os dados que fazem parte do objetivo deste trabalho. Seguindo o procedimento de obtenção, foram acessadas as páginas Web dos departamentos de Ciência da Computação das universidades de Berkeley, Princeton e Stanford e obtida a lista de professores. Executou-se o robô de busca e em seguida utilizou-se a ferramenta *Split* para dividir as referências dentro de uma página e executar o *Tokenizer* para cada uma.

Os primeiros dados obtidos serviram para refinar não somente o *Tokenizer*, mas também o algoritmo de divisão do *Split*. Ao observar os dados notou-se a existência de *tags* HTML não previstas e que eram usadas para separar publicações, um exemplo seria a *tag* "<dt>". No *Tokenizer* foi incluída a identificação de nomes de cidades. Isto

foi possível utilizando uma lista de cidades dos Estados Unidos [6], Inglaterra [11] e Brasil [9], e tomando o cuidado de somente marcar uma palavra como nome de cidade se ela estiver entre pontos (“.”), pois é comum que existam palavras pertencentes ao título, nome de veículo, ou autores que sejam também nomes de cidades.

Outras pequenas modificações foram feitas para melhor adequar o algoritmo *Tokenizer* à realidade e solucionar alguns problemas. Estas modificações foram pontuais e se tratavam de alterações de expressões regulares, erros na colocação de alguma marca e modificação na heurística de encontrar o nome do veículo baseado em alguma característica observada nos testes.

As modificações foram feitas de modo a não deixar o *Tokenizer* sobretreinado para essas três universidades, pois isto o deixaria inflexível para outros conjuntos de dados. Com os dados resultantes após todos os refinamentos (aproximadamente 136000 veículos), foi recolhida uma amostra de 372 elementos. Esse tamanho do espaço de amostras foi escolhido para garantir um nível de confiança de 95% com um intervalo de confiança de 5% para uma análise sobre os dados [14]. Em seguida foi criado um pequeno programa para visualizar as amostras e marcá-las caso a extração do veículo fosse feita incorretamente. Foi considerado como uma extração incorreta se no texto da referência existia um nome de veículo e o *Tokenizer* não conseguiu extrair.

O resultado da análise constatou que 29,8% das amostras apresentaram algum tipo de erro de extração. Este valor adveio da grande variabilidade do modo como as publicações são apresentadas, escritas e formatadas. No enquanto, para o professor Zohar Manna, a taxa de acerto se aproximou de 87% como visto acima. A taxa de acerto elevada em relação ao obtido com as amostras deveu-se ao fato da ferramenta *Tokenizer* estar adaptada ao modo como as referências estão estruturadas na página do professor Manna. Se o modo de apresentação de uma publicação seguisse um padrão internacional seria mais simples a criação de ferramentas e a consequente extração das informações.

3.4 *Ranking* de Veículos Baseado em Reputação

O próximo passo foi a escolha do *ranking* de universidades a ser utilizado para se ordenar os veículos de publicação. Dentre os *rankings* estudados (NRC [12], Times [16], SJTU [3], USnews [13] e USAtoday [5]) foi escolhido o USnews para ser o *ranking* padrão, pois ele possui um *ranking* de universidades baseado nos programas de Ciência da Computação e possui *rankings* para outros campos da ciência. Caso fosse optado por aplicar as ferramentas em outra área, poderia-se obter o *ranking* apropriado que utiliza a mesma metodologia de avaliação das universidades do *ranking* de Ciência da Computação.

Este *ranking* possui mais de 100 universidades que foram ordenadas baseando-se em dois tipos de dados: a opinião de especialistas sobre a qualidade do programa e indicadores

estatísticos que mediam a qualidade dos estudantes, pesquisadores e professores. Estes dados foram coletados até o outono de 2008 e tiveram seus resultados publicados em 2009. Neste *ranking*, a posição de uma universidade reflete o número de universidades acima dela, dessa forma, se as três primeiras universidades possuem igual qualidade, então todas elas recebem a posição "1", já a quarta universidade que tem qualidade menor que essas três receberá a posição "4" e não "2".

Para que o processamento e recuperação das informações não fossem muito demorados, mas ao mesmo tempo se utilizasse um volume de dados suficiente para se ter uma amostra representativa optou-se por trabalhar com as primeiras 60 universidades. Seguindo o plano de ação, foram coletadas listas de professores dos departamentos de Ciência da Computação destas 60 universidades e para cada uma foi executado o robô de busca, a ferramenta *Split* e o *Tokenizer*. Ao final do processo obteve-se um arquivo de dados para cada universidade contendo a lista de veículos extraídos: uma linha para cada veículo no formato apresentado na Seção 3.2.

Nesta etapa era necessário o agrupamento dos dados para se reunir as publicações iguais. Como as ferramentas *Tokenizer* e *Split* e o robô de busca demandaram muito tempo, optou-se por usar uma ferramenta de agrupamento já pronta, desenvolvida por Urubatan Pacheco [36] para sua dissertação de mestrado. Além de agrupar os dados, ela tenta encontrar um nome padrão para representar cada grupo. Esse nome pode ser eleito dentre os pertencentes ao grupo ou pode ser obtido através de uma lista dos principais veículos de publicação conhecidos.

Esta ferramenta de agrupamento utiliza cosseno de similaridade para medir a diferença entre dois vetores de características. Os vetores são construídos a partir dos bigramas que compõe os textos comparados, tomando o cuidado de cada bigrama conter uma letra do bigrama anterior. Este técnica é parecida com a utilizada pelo algoritmo *string kernel* usando um n-grama de tamanho dois.

Um problema encontrado no processamento foi o volume de dados: mais de 145000 veículos foram extraídos das páginas Webs das 60 universidades. Esse total deixa o processo de agrupamento muito lento prolongando ainda mais a obtenção dos resultados. Para contornar esta situação, agrupou-se em duas etapas. Primeiro agrupou-se as universidades individualmente, pois para um volume de dados médio de aproximadamente 2400 veículos não foi muito demorado. Na segunda etapa agrupou-se os representantes desses grupos (agora unindo os dados de todas as universidades). O volume de dados agora é de aproximadamente 67000 veículos (bem abaixo do total inicial de 145000), que apesar de ser 46% menor que inicialmente, deixa o processamento lento. Uma consideração importante é que a primeira etapa de agrupamento colocou vários nomes de veículos sobre apenas um, e a segunda etapa utilizou esses representantes para unir em grupos maiores, mas os tamanhos dos grupos foram preservados. Se a primeira etapa criou os grupos "A"

com 60 membros e "B" com 27 membros, e a segunda etapa juntou os grupos "A" e "B" no grupo "C", então a informação que um tinha 60 nomes e o outro 27 foi passada e o grupo "C" possui 87 membros e não apenas dois.

3.4.1 Cálculo dos Pontos dos Veículos

Como será mostrado abaixo, o problema de pontuar os veículos de publicação de acordo com o prestígio foi abordado de duas formas. A primeira abordagem foi a minimização de uma função de erro sujeita a restrições, mas em virtude dos problemas observados nos resultados (mostrados no capítulo seguinte), viu-se a necessidade de propor um novo método. Este segundo método baseia-se na obtenção da pontuação de cada veículo através da soma do prestígio oriundo de cada universidade.

Primeira Abordagem: Minimização do Erro em um Sistema Linear

O objetivo deste método é a descoberta da pontuação dos veículos de modo que ao somar os pontos dos locais onde as universidades publicam e ordená-las pelo total, se tenha a mesma ordem vista no *ranking* padrão. O algoritmo executado para descobrir estas pontuações demandou muito tempo de processamento e será explicado a seguir. O agrupamento obtido foi também utilizado para gerar outros dois *rankings* de veículos, um para ordenar os veículos que mais aparecem em universidades distintas e outro para ordenar por número de ocorrência dos veículos. Estes resultados se encontram no próximo capítulo.

O terceiro *ranking* obtido é o objetivo deste trabalho e trata-se da ordenação dos veículos extraídos baseado no *ranking* padrão. Basicamente o processo consiste em descobrir valores/pontos para cada veículo de modo que após somar e calcular a pontuação total de uma universidade a ordem estabelecida no *ranking* padrão seja preservada. Estruturou-se o problema da seguinte forma:

$$Ax = b,$$

onde b é um vetor coluna com 60 elementos e cada um deles corresponde à pontuação total de uma universidade, x é um vetor coluna com tantos elementos quanto o número de veículos distintos (1770 veículos), e A é uma matriz de 60 universidades por 1770 veículos onde cada elemento i, j corresponde ao número de referências que a universidade i fez ao veículo j.

Como deseja-se forçar um ranking de universidades, os valores do vetor b podem ser escolhidos de modo que a ordem decrescente preserve a ordem do *ranking* padrão. Portanto utilizou-se uma regra para gerar os pontos das universidades: $(60 - R) \cdot 1000$, onde R é a posição no *ranking* padrão. Exemplo: "Yale University" está na 20ª posição

neste *ranking*, portanto sua pontuação será $(60 - 20) \cdot 1000 = 40000$. Os veículos serão pontuados de acordo com a magnitude desses valores, mas basta uma transformação no resultado final para trabalhar em outras faixas de valores. A matriz A é obtida diretamente a partir do resultado do agrupamento, portanto resta encontrar uma forma de resolver o sistema e obter o vetor x que são os pontos atribuídos aos veículos, para assim ser possível o *ranking*.

A matriz A possui mais colunas do que linhas, o que significa que falta informação para definir os pontos. Portanto existem múltiplas soluções para o sistema. Também, o fato de a matriz não ser quadrada torna impraticável sua inversão. Para resolver este problema estudou-se primeiramente uma forma de se calcular a pseudo-inversa da matriz A, o que levou ao algoritmo SVD [22]. Este algoritmo foi implementado e executado. O resultado não foi satisfatório por duas razões: o SVD possui um certo erro que é amplificado pela desigualdade entre o número de linhas e colunas da matriz A o que fez com que a multiplicação Ax fosse diferente de b; a segunda razão é que este algoritmo atribuiu pontos negativos aos veículos, o que na prática significa que existiriam veículos que quanto mais se publica neles menor a pontuação da universidade. Dessa forma, o resultado de atribuição de pontos não deveria permitir números negativos, pois esta atribuição não considera qualquer critério de prestígio, apenas uma possível solução matemática.

Como o SVD não obteve bons resultados estudou-se algoritmos de minimização, como o método Simplex, o gradiente conjugado e BFGS [45, 20]. Os algoritmos BFGS e Simplex foram utilizados a partir das bibliotecas científicas da linguagem Python e não foi necessária a implementação. Já o algoritmo do gradiente conjugado, apesar de existir na biblioteca científica de Python, foi implementado para se ter maior controle nas iterações.

Também foi implementado o método do gradiente, que é um método simples e rápido de codificar e permite bastante controle sobre as iterações. Este método seria usado para se obter um limitante do valor do erro e avaliar a eficácia e rapidez dos algoritmos mais elaborados. Ocorreu que o método do gradiente se mostrou o mais robusto para solucionar este problema.

O gradiente conjugado e o BFGS obtiveram valores de erro insignificantes, mas pontuaram alguns veículos com valores negativos, o que mostrou que estes métodos, sozinhos, não resolveriam o problema.

O método Simplex teve sua execução abortada, pois era muito lento e concluiu-se que ele teria o mesmo problema do gradiente conjugado e do BFGS, então o seu resultado não foi obtido. Por se tratar de uma função pronta da linguagem Python não se sabia qualquer informação a respeito do processamento e portanto não havia como saber se estava no final ou no começo das iterações (também não havia qualquer informação na tela durante o andamento para mostrar iterações, erro, etc.).

Já o método do gradiente (o simples e não o conjugado) permite maior controle sobre

os passos, e consegue ir diminuindo o erro ao longo das iterações, admitindo somente pontuações com valores positivos. O algoritmo é o seguinte:

- Os pontos dos veículos são inicializados com zero, fazendo $x = \begin{bmatrix} 0 & 0 & ... & 0 \end{bmatrix}$.
- As iterações ocorrem até o erro ser menor que um certo *delta* ou o número de iterações exceder o máximo.
- Calcula-se $e = b - Ax$. O vetor erro e é um gradiente que servirá para corrigir os pontos dos veículos.
- Para corrigir os pontos de uma universidade, pondera-se o valor de erro dela entre os veículos, fazendo:

* Para cada par (universidade u, veículo v):

- Calcula-se: $\delta_v = e_u \cdot \frac{\text{(referências da univ. u para o veículo v)}}{(\textbf{fator} \cdot \text{total de referências feitas por u})}$, onde *fator* é um termo para atenuar o gradiente.
- Os novos valores de x são dados por: $x_v = \begin{cases} 0 & \text{se } \delta_v < 0 \\ \delta_v & \text{caso contrário} \end{cases}$

O coeficiente atenuador chamado *fator* é utilizado para diminuir a intensidade do gradiente quanto mais próximo de um mínimo. Inicialmente ele vale um e não altera o cálculo. Quando o erro da iteração atual for maior que o erro da iteração anterior, ou seja, a "força" do gradiente passou a solução mínima e encontrou um ponto que é pior que o ponto anterior, o valor do *fator* é aumentado em 45% para diminuir a intensidade e permitir uma melhor aproximação do mínimo.

O *fator* também é usado para causar uma mudança brusca da solução no caso do gradiente avançar por uma "planície" de soluções. A planície de soluções é caracterizada pelo fato de que os pontos ao redor da solução atual possuem o mesmo valor da função erro: então, por mais que o gradiente procure uma solução melhor ele não consegue encontrá-la e pode começar a repetir valores. Quando se verifica que nas 10 iterações anteriores o valor do erro permaneceu o mesmo, o valor de *fator* altera-se para 0,9, o que amplifica a intensidade do gradiente. Isto faz com que o algoritmo saia dessa região, mas consequentemente eleva muito o valor do erro. A elevação acentuada do erro não é um problema, pois o caminho inicial percorrido não levou a uma boa solução então essa mudança brusca é como se o algoritmo recomeçasse de um outro ponto do espaço de soluções propiciando uma nova oportunidade de encontrar soluções melhores.

A diminuição da intensidade em 45%, a mudança brusca quando o mesmo valor de erro persiste por 10 iterações e aumento da intensidade mudando o valor de *fator* para 0,9 foram valores obtidos através de exaustivos testes em que se analisou como a mudança

desses números implicava na queda do erro ao longo das iterações. Essa configuração se mostrou a melhor opção visando alcançar um valor menor de erro em menos iterações.

Depois de definidos todos os parâmetros do algoritmo, ele foi executado para o problema em questão de modo completo, ou seja, até o erro se tornar 10^{-6}. Ao analisar o resultado do processamento verificou-se que havia erros devido às características dos dados. Se um veículo aparece na lista de publicações de somente uma única universidade pode ter ocorrido um de dois casos: somente uma universidade publicou neste veículo, ou ocorreu um erro na extração do nome. O erro de extração pode ter ocorrido porque tentou-se obter um nome de veículo de um texto que não o tinha ou havia um nome, mas o algoritmo não foi capaz de identificar. Dessa forma quando os pontos dos veículos estão sendo calculados pelo algoritmo do gradiente, esses veículos que aparecem em somente uma universidade servem para ajustar o somatório de pontos, portanto o que se observou foi a presença de veículos com altas pontuações, mas que se tratavam de erros de extração. Essa foi a razão da obtenção de um erro próximo de zero, pois dados os pontos dos veículos corretos de modo que a soma de pontos para cada universidade fique sempre abaixo do esperado, estes veículos que aparecem em uma única universidade têm seus pontos ajustados para chegar ao erro zero.

Viu-se a necessidade de excluir estes veículos que não aparecem em mais de uma universidade para corrigir este problema. Isto provavelmente eliminou veículos corretos, mas para haver uma melhor pontuação dos veículos deve existir uma "concorrência" entre as universidades, pois um veículo que uma única universidade publica não há como saber se ele é bom ou ruim.

Antes de executar novamente o algoritmo efetuou-se alguns testes para tentar encontrar outros possíveis erros que poderiam ocasionar o mesmo problema. Foi verificado que existiam vários veículos cujos nomes estavam errados (não um erro de grafia ou omissão, mas um texto que não é um nome de veículo) e que estavam presentes em duas ou três universidades. Acredita-se que isto foi gerado pelo fato de existirem estruturas similares em diferentes universidades que ao sofrerem ação dos algoritmos *Split* e *Tokenizer* geravam um "lixo" parecido. No momento do agrupamento, essas falhas parecidas acabam juntas e geram um nome de veículo falso que tem a força de pertencer a mais de uma universidade. Por exemplo, se existirem alguns falsos nomes extraídos, tais como, "pdf]" , "ps pdf" , "[pdf]" , "pdf" e "/pdf", eles poderiam ser todos agrupados sob o nome "pdf" (escolhido para representá-los) e se ocorresse em outra universidade um grupo com o nome "-pdf/", eles poderiam se unir no grupo "pdf", fazendo com que esse nome de veículo falso esteja presente em duas universidades. Também foram encontrados alguns poucos nomes de veículos falsos pertencentes a quatro universidades, mas a quantidade de veículos corretos pertencentes a esse número de universidades é muito alto. Optou-se por incluir no processamento somente os veículos que aparecem em quatro ou mais universidades para

eliminar esses nomes errados. Como já argumentado anteriormente, essa medida exclui alguns veículos corretos, mas beneficia o processamento em geral, que tentará encontrar os pontos utilizando veículos presentes em no mínimo quatro universidades.

Nesta nova configuração o algoritmo do gradiente foi novamente executado. Nesta última versão os valores de erro não se reduziram adequadamente gerando resultados muito ruins depois de várias iterações. Isso deve-se ao fato de cada veículo influenciar o cálculo dos pontos de no mínimo 4 universidades, não existindo mais aqueles veículos que serviam somente para ajuste da pontuação. O algoritmo do gradiente foi revisto com relação aos parâmetros de ajuste, mas aqueles definidos anteriormente se mostraram bons entre os outros testados. Uma alternativa pensada para melhorar a solução foi imaginar que se os pontos iniciais fossem uma solução próxima de um mínimo então o método do gradiente poderia avançar mais rápido para uma boa solução, ao contrário, se os pontos iniciais são aleatórios (no caso todos começam com zero) então pode ser que o método do gradiente tenha que percorrer um longo caminho no espaço de soluções para encontrar uma boa solução. Portanto tentou-se combinar os diferentes algoritmos para que um fornecesse os pontos iniciais para o outro. A melhor configuração obtida foi a sequência SVD, gradiente conjugado e Gradiente, ou seja, executa-se o SVD para se obter a pseudo-inversa da matriz e calcular os pontos dos veículos, essa pontuação é tida como os pontos iniciais para o método do gradiente conjugado, que após algumas iterações (suficientes para se obter um boa solução) fornece os pontos iniciais para o método do gradiente. Esta ligação entre os três algoritmos proporcionou um melhora na diminuição do erro. Independente desses procedimentos, era necessário realizar inúmeras iterações para se obter um valor de erro adequado. O erro diminuía pouco em relação ao aumento do número de iterações, portanto foram executados em paralelo instâncias do algoritmo do gradiente iterando no máximo 10.000, 50.000, 100.000, 150.000, 200.000 e 300.000 vezes (o que elevou muito o tempo de processamento).

Como o processamento era demorado optou-se por criar um outro algoritmo para tentar encontrar soluções melhores. Em virtude disto programou-se em linguagem Python um algoritmo genético [28].

Para este algoritmo é criada uma população de soluções geradas aleatoriamente cujo tamanho é um parâmetro de entrada, depois aplica-se um método de reprodução, mutação e seleção natural. O número de vezes que esse processo se repete é definido também na entrada do programa. A seguir está exposto cada um desses métodos:

- **Reprodução :** Aleatoriamente são selecionados pares de soluções que irão gerar três outras soluções. Uma delas é uma média entre os valores dos dois pares, e as outras duas soluções são as obtidas pela troca de segmentos da solução dos dois indivíduos. Este ponto que determina o segmento é o ponto de *crossover*.

- **Mutação :** Uma certa porcentagem dos indivíduos tem um de seus valores alterado aleatoriamente. Esta porcentagem é um parâmetro de entrada.

- **Seleção Natural :** Este método preserva os 40% melhores indivíduos, os 10% piores indivíduos (para manter uma variabilidade genética), e escolhe aleatoriamente os outros 30% dentre os restantes.

Foram executadas instâncias para o algoritmo genético com 100 e 1000 indivíduos, com 10, 50 e 100 iterações (para 100 indivíduos) e 100 e 500 iterações (para 1000 indivíduos). O processamento, assim como os outros algoritmos em execução, não era rápido, pois o cálculo para saber se um indivíduo era uma boa solução ou não demandava muito tempo.

A análise dos erros e problemas para o método do gradiente e o algoritmo genético, bem como os *rankings* obtidos, encontram-se na Seção 4.4.

Segunda Abordagem: Soma dos Prestígios

Como será visto nos resultados, o primeiro método proposto para pontuar os veículos não foi satisfatório, então uma nova metodologia foi criada. Foi desenvolvido um algoritmo que fosse rápido e fiel ao principal objetivo deste trabalho: ordenar os veículos de acordo com a reputação. Este algoritmo chama-se *Score* e também foi implementado em linguagem Python.

Antes de explicar o funcionamento do *Score* é necessário apresentar alguns termos importantes que são a base do algoritmo.

- **Abrangência Local**: É um valor atribuído a um veículo e representa o quão abrangente ele é dentro de uma universidade. O termo "local" é em virtude deste valor ser particular a uma universidade, portanto um mesmo veículo possui valores diferentes de abrangência para cada instituição. A abrangência local de um veículo é obtida dividindo-se o número de pesquisadores distintos que ali publicaram pelo número total de pesquisadores da instituição. Neste cálculo não importa o número de ocorrências do veículo, apenas quantos pesquisadores diferentes fizeram as publicações. Suponha, por exemplo, que somente dois pesquisadores entre os 100 de uma universidade realizam publicações no veículo X. Neste caso a abrangência local de X vale $0,02$ e indica que poucos pesquisadores consideram este veículo prestigiado. O valor máximo possível é obtido quando todos os pesquisadores publicaram em determinado veículo, resultando no valor 1.

- **Peso Local**: É um valor atribuído a um veículo dentro de uma instituição e representa sua "força" ou prestígio. Seu valor é obtido multiplicando-se a abrangência local pelo total de ocorrências de um veículo dentro de uma universidade. Suponha,

por exemplo, que dois veículos X e Y de uma instituição possuem abrangências locais 0,02 e 0,6 respectivamente, e ambos possuem 80 ocorrências cada um. Portanto o peso local de X é $1,6$ e o peso local de Y é 48. Isto significa que o veículo Y é muito mais prestigiado que o X para esta instituição.

- **Peso da Instituição**: É um valor atribuído a uma instituição e representa sua qualidade. O valor é obtido com base no *ranking* padrão e considera a posição da universidade nesta ordenação. O peso da instituição é dado por (60 - posição da universidade no ranking). A universidade em primeiro lugar terá o peso igual a 59 e a universidade em 58º terá peso igual a 2, por exemplo.

O peso local de um veículo fornece a informação necessária para avaliar o prestígio dentro de uma universidade. Quando se analisa um veículo do ponto de vista de um pesquisador, seu prestígio é medido através do número de ocorrências. Quando se analisa um veículo do ponto de vista da universidade, o prestígio é dado pelo número de pesquisadores diferentes que publicam nele. O prestigio dado pela universidade tem mais valor do que o atribuído por um único pesquisador, pois representa um senso comum entre os autores.

Basicamente a pontuação final de um veículo é dada pela soma dos pesos locais através das instituições, mas existe um outro fator envolvido. A posição da universidade no *ranking* padrão também deve influenciar a pontuação do veículo. O fato, por exemplo, de uma instituição nas primeiras posições publicar em um veículo lhe atribui mais prestígio que se a última universidade no *ranking* realizar esta publicação. Portanto, a parcela oriunda de cada universidade para compor a pontuação final do veículo é uma multiplicação entre o peso local e o peso da instituição. A seguinte fórmula exemplifica o cálculo:

$$\text{Pontuação do veículo } v = \sum_{u \in universidades} [\ \text{(peso-local-de-v-em-u)} \times \text{(peso-da-instituição-u)}\ ]$$

Esse cálculo pontua os veículos de publicação de acordo com a reputação. Após obter os valores pode-se utilizá-los para pontuar as universidades e obter um novo *ranking* das instituições. Este novo *ranking* pode ser comparado com o *ranking* padrão que foi a base do cálculo dos pesos das instituições. Esta comparação encontra-se no capítulo seguinte.

Para se ordenar novamente as universidades deve-se pontuá-las utilizando os veículos onde elas publicaram. Optou-se por ponderar a pontuação dos veículos multiplicando-a pelo total de ocorrências dentro da cada instituição. Portanto a pontuação da universidade é dado pela seguinte fórmula:

$$\text{Pontuação da universidade u} = \sum_{v \in veiculos-em-que-u-publicou} [\ \text{(pontuação-de-v)} \times \text{(total-de-ocorrências-de-v-em-u)}\ ]$$

O cálculo da pontuação da instituição é diretamente proporcional ao número de ocorrências, pois espera-se que quanto mais publicações são feitas maior a pontuação final. Não é usada a abrangência ou o peso local, pois estes valores refletem uma visão interna a respeito do veículo e não o que ele representa perante todas as universidades. Os resultados obtidos com o algoritmo *Score* são mostrados no próximo capítulo.

3.5 Divisão dos Veículos em Conferências e Periódicos

Com os veículos extraídos das 60 universidades resolveu-se dividi-los entre conferências e periódicos e analisar a porcentagem destes por professor. Esta análise também foi obtida por ano, pois esta informação também foi extraída pelo *Tokenizer*. Os resultados evidenciam qual tipo de veículo é o mais utilizado para publicação numa determinada universidade, bem como a forma que a relação entre conferências e periódicos vêm mudando através dos anos.

Para realizar essa identificação foi implementada outra ferramenta em linguagem Python que pudesse rotular um veículo como um periódico ou conferência. O sistema baseia-se em regras e expressões regulares para encontrar padrões nos textos. Também são utilizadas listas de nomes de veículos obtidas manualmente, as quais sabia-se antecipadamente se tratar de uma conferência ou periódico.

O programa trabalha da seguinte forma: primeiro procura-se por padrões que indiquem ser uma tese ou relatório técnico; depois procura-se por nomes de veículos que com certeza são conferências; em terceiro procura-se por nomes e abreviaturas que caracterizam um periódico; em quarto usa-se outra lista de conferências e abreviações para tentar encontrar um padrão, em quinto tenta-se encontrar padrões para uma referência Web ou um livro; e por último tenta-se verificar se é um periódico através da estrutura de volume, edição e página. A ordem das buscas é importante pois alguns padrões só podem ser rotulados se já foram descartadas outras possibilidades.

Portanto rotulou-se todos os veículos extraídos em conferências, periódicos e outros. Depois, para cada universidade, calculou-se o percentual de conferências por professor e o percentual de periódicos por professor para cada ano entre 1990 e 2008. O ano de 2009 foi descartado, pois é o ano corrente e ainda não se tem todas as referências divulgadas, e os anos anteriores à 1990 também não foram incluídos por se acreditar que a faixa temporal considerada de 18 anos é bem representativa.

Para que esse tipo de análise fosse confiável, verificou-se quantos veículos não tinham a informação do ano, já que esse é um dado vital para esta análise. Foram detectados que apenas 2% dos periódicos e 1% das conferências não tinham a informação do ano, sendo que no total tem-se 35544 veículos identificados como periódico e 60014 veículos identificados como conferência. Essa taxa de omissão do ano está bem baixa o que qualifica

melhor o resultado.

Para avaliar a qualidade de classificação da ferramenta coletou-se aleatoriamente 383 referências dentre as obtidas. Esse tamanho do espaço de amostras foi escolhido para garantir um nível de confiança de 95% com um intervalo de confiança de 5% para uma análise sobre os dados [14]. Para a análise proposta, o foco é sobre periódicos e conferências, portanto a contagem de erros seguiu essa lógica. Se a ferramenta classificou uma referência com um tipo que não é periódico ou conferência, considera-se simplesmente que o tipo é "outro". Então a verificação de erro ocorre entre três tipos de classes: periódico, conferência e outro. Desse modo, se uma referência foi classificada, por exemplo, como livro e na verdade era uma página Web, não se considera o erro. O erro obtido nas amostras é de 6,8%, que é considerado adequado e aumenta a confiabilidade nos resultados.

Os resultados obtidos com essa análise e discussões serão mostradas no próximo capítulo.

Capítulo 4

Resultados Obtidos

Neste capítulo os resultados obtidos estão estruturados de acordo com o tipo do algoritmo (ou *ranking*). Primeiramente é mostrado o *ranking* dos veículos ordenados por número de ocorrências em universidades (abrangência). O segundo *ranking* apresentado ordena os veículos por número total de ocorrências. Em seguida são apresentados os resultados da análise entre o percentual de conferências e periódicos por professor ao longo dos anos. O quarto *ranking* mostrado ordena os veículos baseando-se em reputação, conforme o algoritmo da primeira abordagem visto no capítulo anterior. Por último é apresentado o *ranking* baseado em reputação de acordo com o algoritmo da segunda abordagem.

4.1 *Ranking* de Veículos Baseado no Número de Ocorrências em Universidades

O primeiro *ranking* considera a abrangência do veículo e ordena-os por número de universidades que já o referenciaram, exemplo, se todas as 60 universidades referenciam o veículo "A" então este veículo recebe a pontuação 60, e se outro veículo "B" foi referenciado por 3 universidades então ele recebe a pontuação 3. Esse cálculo desconsidera o número de vezes que aquele veículo foi referenciado, sendo apenas importante saber em quantas universidades diferentes ele aparece. A lista com os 100 primeiros veículos ordenados por número de universidades está no Apêndice B. Esta lista também considera apenas os veículos que aparecem em mais de três universidades, que é a mesma restrição adotada para os outros *rankings* deste trabalho. Na Tabela 4.1 está a lista das 20 primeiras universidades de acordo com este *ranking*.

Este *ranking* apresenta um equilíbrio entre o número de conferências e periódicos: dentre os 20 primeiros, 10 são conferências. Na primeira posição, o periódico *IEEE Transactions on Computers* ocorre em 50 universidades diferentes, ou seja, 83% das instituições

Tabela 4.1: *Ranking* de veículos baseado em sua abrangência

RANK	Nome do Veículo
50	IEEE Transactions on Computers
49	IEEE International Conference on Distributed Computing Systems
44	International Conference on Parallel Processing
43	IEEE International Parallel and Distributed Processing Symposium
43	Journal of Parallel and Distributed Computing
41	IEEE Computer
41	IFIP International Conference on Theoretical Computer Science
40	Information Processing Letters
40	SIAM Journal on Computing
39	IEEE Infocom
39	IEEE/ACM Transactions on Networking
38	IEEE International Conference on Robotics and Automation
38	Communications of the ACM
37	International Joint Conference on Artificial Intelligence
37	IEEE Transactions on Parallel and Distributed Systems
36	IEEE Transactions on Pattern Analysis and Machine Intelligence
36	IEEE Conference on Computer Vision and Pattern Recognition
35	IEEE Transactions of Software Engineering
35	ACM International Conference on Supercomputing
35	International Conference on Very Large Data Bases
...	...

analisadas já publicaram (uma ou mais vezes) nestes periódicos. O fato de um veículo estar nas primeiras posições não fornece informações para avaliar sua qualidade, pois a importância de um veículo não está necessariamente associada ao número de universidades distintas em que ele ocorre.

4.2 *Ranking* de Veículos baseado no Número Total de Ocorrências

O segundo *ranking* ordena os veículos de acordo com o número de ocorrências considerando-se a soma em todas as 60 universidades. Os veículos são ordenados do maior para o menor, ou seja, o veículo com mais ocorrências é considerado melhor que um com menos. Esta lista (com os 100 primeiros veículos) se encontra no Apêndice C e na Tabela 4.2 consta seus 25 primeiros veículos. Já a Tabela 4.3 destaca, para os periódicos dentre os 25 primeiros veículos, o Fator de Impacto e o número de citações de acordo com o ISI [43].

Nas primeiras posições ocorre uma alternância entre periódicos e conferências, mas

Tabela 4.2: *Ranking* de veículos baseado no número total de ocorrências

Número de Ocorrências	Nome do Veículo
602	IEEE International Conference on Robotics and Automation
389	IEEE Transaction on Computers
362	IEEE Conference on Decision and Control
352	SIAM Journal on Computing
344	IEEE Transactions on Automatic Control
291	ACM International Symposium on Computer Architecture
267	IEEE Conference on Computer Vision and Pattern Recognition
262	IEEE Infocom
252	IEEE International Conference on Distributed Computing Systems
249	International Conference on Parallel Processing
231	IEEE / ACM Transactions on Networking
226	IFIP International Conference on Theoetical Computer Science
226	IEEE International Conference on Data Engineering
223	IEEE Transactions of Software Engineering
223	International Joint Conference on Artificial Intelligence
223	IEEE Symposium on Foundations of Computer Science
222	Journal of Parallel and Distributed Computing
201	Neural Information Processing Systems
200	AAAI Conference on Artificial Intelligence
199	IEEE International Conference on Computer Vision
198	IEEE Transactions on Pattern Analysis and Machine Intelligence
193	ACM SIAM Symposium on Discrete Algorithms
184	IEEE International Parallel and Distributed Processing Symposium
182	IEEE Computer
177	Information Processing Letters
...	...

Tabela 4.3: Fator de Impacto e número de citações para os periódicos do *ranking* baseado no número total de ocorrências

Número de Ocorrências	Nome do Veículo	Número de Citações / Fator de Impacto (JCR)
389	IEEE Transaction on computers	9240 / 2.611
352	SIAM Journal on Computing	4634 / 1.459
344	IEEE Transactions on Automatic Control	23227 / 3.293
231	IEEE / ACM Transactions on Networking	6129 / 2.576
223	IEEE Transactions of Software Engineering	5449 / 3,569
222	Journal of Parallel and Distributed Computing	1551 / 1,168
198	IEEE Transactions on Pattern Analysis and Machine Intelligence	24674 / 5.960
177	Information Processing Letters	2823 / 0,706

este último tipo de veículo está mais presente nestes 25 primeiros do *ranking*, totalizando 16 conferências (64%). Foi adicionado o número de citações e o fator de impacto para os veículos disponíveis na base JCR para verificar alguma compatibilidade, mas a ordenação obtida não se mostrou compatível com o fator de impacto ou com o total de citações.

Tanto a qualidade quanto o prestígio de um veículo não podem ser avaliados unicamente pelo número de ocorrências, mas seu valor indica uma tendência de publicação, sendo o veículo prestigiado ou não.

4.3 Análise da Percentagem entre Conferências e Periódicos ao Longo dos Anos

Nesta seção são mostrados os resultados referentes à comparação dos percentuais de conferências e periódicos por professor ao longo dos anos. São exibidos os dados de quatro universidades que ilustram bem o comportamento das outras instituições.

A Figura 4.1 contém os dados sobre “Massachusetts Institute of Technology”, a Figura 4.2 contém os dados sobre “University of California - Berkeley”, a Figura 4.3 contém os dados sobre “Princeton University”, a Figura 4.4 contém os dados sobre “Georgia Institute of Technology”, e a Figura 4.5 reúne os dados das 60 universidades. As quatro universidades escolhidas para exibir os resultados encontram-se entre as 10 primeiras no *ranking* padrão e foram escolhidas pela característica dos gráficos gerados.

Em todos os gráficos são mostrados crescentes distanciamentos entre as curvas, mas nos últimos anos ocorre uma diminuição do número médio de conferência e, em alguns casos, acompanhada da redução do número médio de periódicos. Uma causa provável para esse comportamento dos gráficos é a desatualização das páginas Web dos pesquisadores.

Os artigos recentes podem não ter sido inseridos na relação de publicações, gerando essa diferença. No caso, por exemplo de Princeton e MIT, onde existe uma diminuição brusca do número médio de conferências no último ano pode ter sido causado pelo fato da página Web ter sido atualizada com os periódicos, mas não com outros tipos de publicações. Essas quedas não condizem com o comportamento que se observa através dos anos nos gráficos, contudo as curvas mostram como o número médio de publicações em conferências vem aumentando, chegando a ser até o dobro do número médio de periódicos, como por exemplo, em Berkeley e Gatech.

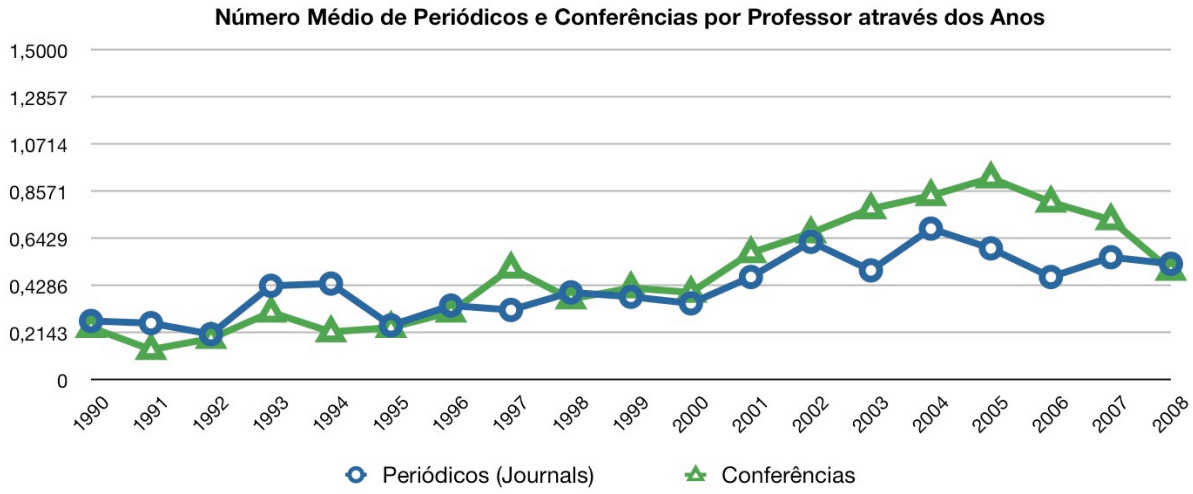


Figura 4.1: **MIT** - Gráfico do número médio de periódicos e conferências por professor através dos anos

4.4 *Ranking* Baseado em Reputação

Primeiro será exposto os resultados e análises para os testes realizados de acordo com a primeira abordagem, a minimização do erro em um sistema linear. Como este método apresentou alguns problemas, criou-se um segundo método para ordenar os veículos de acordo com a reputação. Os resultados, desta segunda abordagem, estão na seção seguinte ao da primeira.

4.4.1 Primeira Abordagem: Minimização do Erro em um Sistema Linear

Para o modelo proposto, ou seja, a construção de uma matriz que indica quantas referências uma universidade fez para os veículos e a resolução do sistema $Ax = b$, não

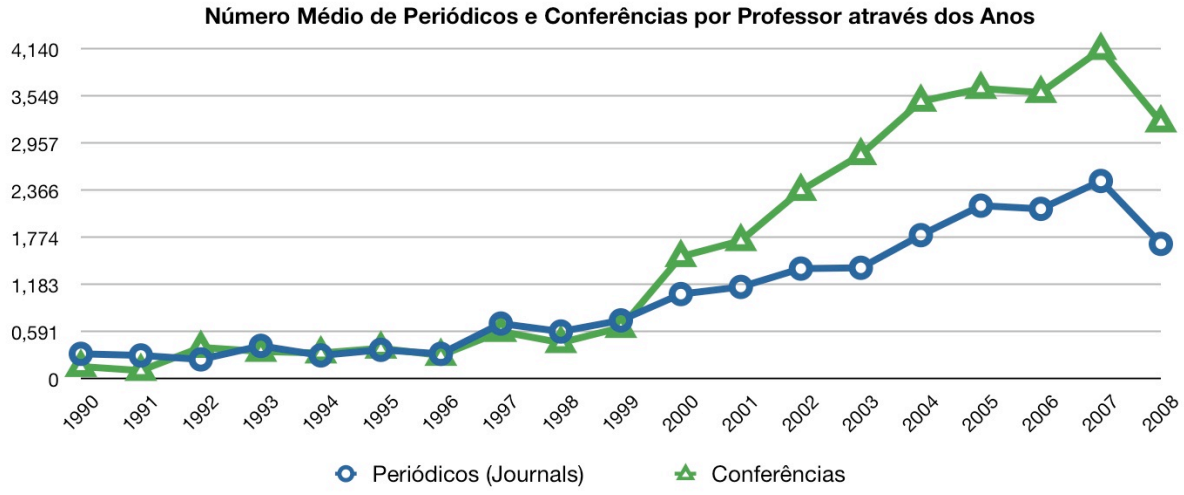


Figura 4.2: **Berkeley** - Gráfico do número médio de periódicos e conferências por professor através dos anos

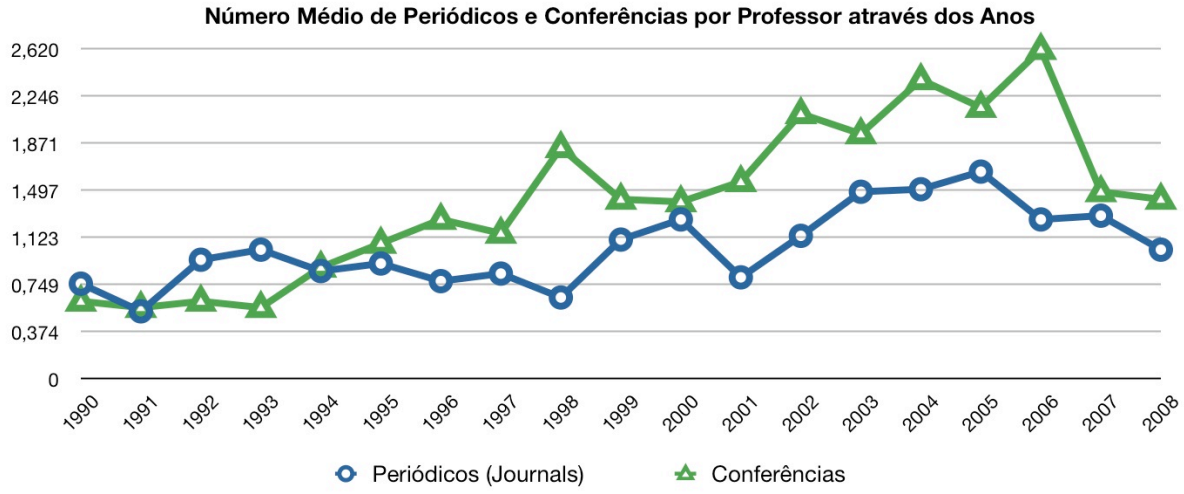


Figura 4.3: **Princeton** - Gráfico do número médio de periódicos e conferências por professor através dos anos

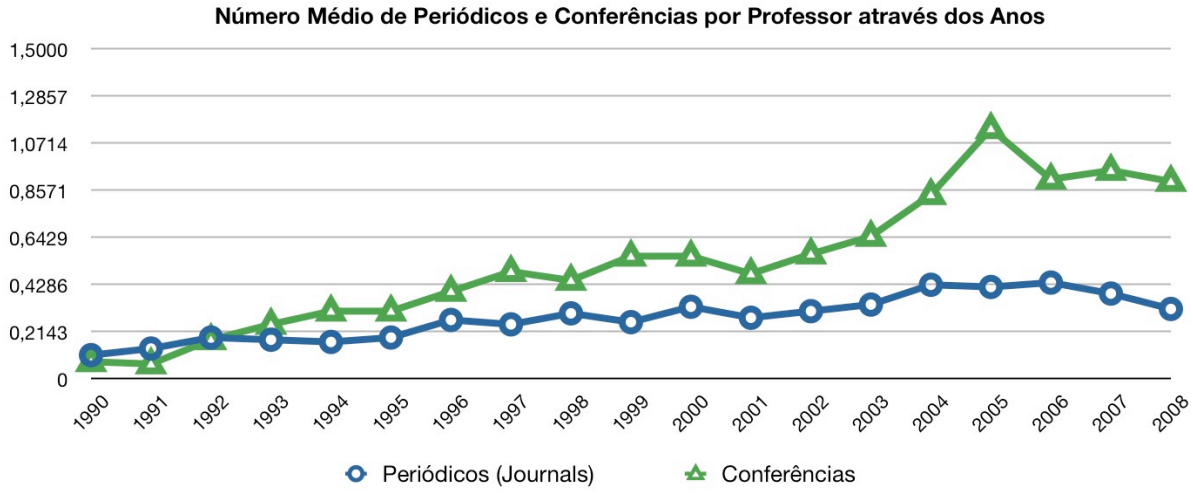


Figura 4.4: **Gatech** - Gráfico do número médio de periódicos e conferências por professor através dos anos

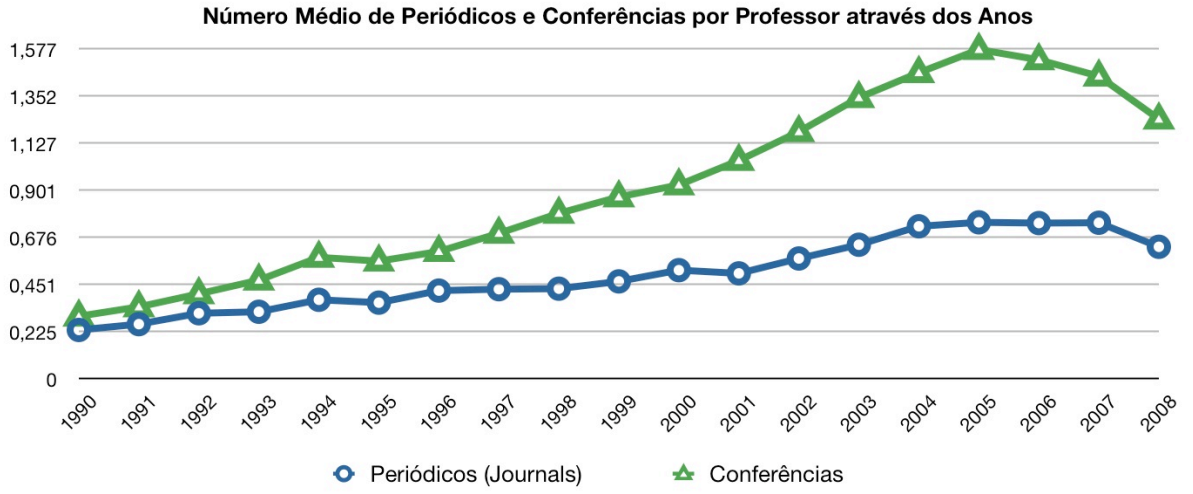


Figura 4.5: Gráfico do percentual de número médio e conferências por professor através dos anos considerando as 60 universidades

obteve um resultado satisfatório com relação ao erro obtido na minimização. As soluções obtidas com o algoritmo genético ficaram aquém do esperado e do erro obtido com o método do gradiente. Os valores de erro ficaram acima de 1 milhão contra pouco mais de 70000 do método do gradiente. Para entender o que significam esses valores de erro, de acordo com a construção do vetor b, a diferença entre uma universidade e outra imediatamente acima ou abaixo no *ranking* é de 1000 pontos. Dado que uma universidade deveria ter X pontos, mas pelo peso atribuído aos veículos ela ficou com Y, então o erro desta universidade será o valor absoluto de $(X - Y)$. O erro informado pela ferramenta e que foi mostrado acima trata-se do somatório dos erros de todas as 60 universidades. Portanto dizer que o valor de erro é por volta de 70000 significa que provavelmente todas as universidades estão deslocadas do valor esperado em mais de 1000 pontos. Isto não significa que o *ranking* está completamente desorganizado em relação ao do *ranking* padrão, pois o importante é manter a ordem entre as instituições e não coincidir perfeitamente os pontos. Um exemplo é se todas as universidades ficassem com 1000 pontos acima do esperado, então o erro seria de 60000 pontos, mas a ordem das instituições estaria correta em relação a original. Como o erro do algoritmo genético ficou muito alto para todos as instâncias executadas seu resultado foi descartado.

Como exposto acima, os valores de erro para as execuções do método do gradiente foram os melhores para a primeira abordagem. A Tabela 4.4 mostra o valor do erro de acordo com o número de iterações do algoritmo.

Tabela 4.4: Valores de erro para as execuções do método do gradiente

Número de iterações	Valor do erro
10.000	96646,69
50.000	73782,66
100.000	72030,18
150.000	72030,18
200.000	72030,18
300.000	72030,18

Percebe-se que a partir das 100000 iterações o valor de erro foi o mesmo, não sendo possível saber se isto se repetirá indefinidamente. Esta solução com 72030,18 pontos de erro pode ser a melhor possível para o modelo usado ou o algoritmo passa por uma "planície de soluções" e não consegue sair. Como o processamento demanda muito tempo, optou-se por finalizar os testes com esse valor de erro.

O *ranking* obtido foi comparado com o *ranking* padrão usando o coeficiente de correlação de Kendall. O valor do coeficiente foi 0,06, indicando que o *ranking* não está adequadamente correlacionado com o usado como base. A ordenação dos veículos também

não foi satisfatória e os motivos serão mostrados em seguida. A lista com os 100 primeiros veículos de acordo com este *ranking* está no Apêndice D e na Tabela 4.5 consta os primeiros 20 veículos para análise.

Tabela 4.5: *Ranking* de veículos baseado em reputação - primeira abordagem

Ponto Atribuído	Nome do Veículo	Número de Citações / Fator de Impacto (JCR)
16031,04	IEEE Computer Society Workshop on Perceptual Organization in Computer Vision	...
9511,75	Sigact News	...
8654,28	International Computer Software Applications Conference	...
6637,29	IEEE International Conference on Electronics	...
5292,22	Usenix	...
4827,20	Workshop on Hot Topics in Measurement Modeling of Computer Systems	...
4512,65	IEEE International Symposium on Defect Fault Tolerance in VLSI Systems	...
4307,92	ACM Workshop on Security of Ad-Hoc Sensor Networks	...
3851,13	IEEE Conference on Computational Complexity	...
3431,65	ACM Sigcomm Workshop on Hot Topics in Networks	...
3104,65	IEEE Computer Graphics and Applications	1930 / 1.866
2707,05	Communications of the ACM	12617 / 2.646
2700,91	IEEE International Computer Performance Dependability Symposium	...
2499,64	Visualization Handbook	...
2337,51	Geometric Modeling Processing	...
2326,12	Joint Conference on Empirical Methods Natural Language Processing very Large Corpora	...
2055,58	Computers & Operations Research	3389 / 1.366
2042,56	ACM Internet Measurement Conference IMC	...
2038,65	Large Installation System Administration Conference	...
2002,66	International Society for Magnetic Resonance Medicine	...
...	...	

Dentre os 20 primeiros veículos, somente cinco deles são periódicos. Este comportamento era esperado, já que as conferências são um tipo de veículo fortemente presente nas universidades. Um problema encontrado foi a presença de *workshops* nas primeiras posições. Este tipo de veículo não deveria estar entre os mais importantes de um *ranking*.

Analisando o primeiro veículo em específico ("IEEE Computer Society Workshop on

Perceptual Organization in Computer Vision") tem-se que ele foi referenciado somente quatro vezes e por quatro universidades diferentes. Uma busca nos dados revela que as universidades que referenciaram este Workshop foram: Rensselrer Polytechnic Insitute (48ª posição), University of Chicago (39ª posição), University of Massachusetts (20ª posição), e University of California-Berkeley (1ª posição). Portanto este veículo recebeu uma alta pontuação porque serviu como um ajuste matemático para posicionar Berkeley no topo do *ranking*. Para exemplificar o problema, se uma universidade publicou quatro vezes neste veículo e em nenhum outro, ela teria pontuação para ficar em primeiro lugar, o que não deveria ser verdade devido ao baixo número de publicações.

O segundo veículo no *ranking* ("Sigact News") teve valores um pouco mais altos: foi referenciado por 11 universidades diferentes e teve ao todo 16 referências. Sua pontuação alta foi porque dentre as universidades que o referenciaram estão algumas bem posicionadas como MIT, Cornell, Gatech, Caltech e Harvard. Novamente, para ilustrar o problema, é suficiente sete publicações neste veículo para que a instituição fique em primeiro lugar.

Quando foi proposto, este modelo matemático parecia adequado, pois usa o fato de que quanto mais se publica em veículos renomados, maior deve ser a pontuação da instituição. Ocorre que na tentativa de encontrar os pontos dos veículos, ou seja, fazer o cálculo inverso, os que são mais referenciados podem ter baixas pontuações já que seu multiplicador é alto. Por outro lado, os veículos menos referenciados podem receber pontos mais altos pois seu multiplicador é baixo. Analisando de outra forma este resultado, os veículos pouco referenciados e com altas pontuações seriam o diferencial que uma universidade tem para se destacar em relação as outras, já que os veículos muito referenciados podem não fornecem a distinção necessária para afirmar porque uma é melhor que a outra.

Foram analisadas outras formas de compor a matriz para resolução do problema, como por exemplo, inversamente proporcional ao número de referências, mas o problema persistiu.

Tentou-se aplicar um corte nos dados selecionando somente os veículos que estivessem presentes em mais de uma certa quantidade de universidades. Esse valor de corte foi sendo variado até o máximo de 50. Para cada instância aplicou-se os mesmos métodos de minimização, mas no modelo proposto de resolução o erro fica com um valor muito alto não sendo possível analisar corretamente o resultado.

Portanto o problema na computação do resultado está no modelo matemático proposto. Mesmo que fosse possível zerar o valor do erro com a minimização ainda estariam presentes veículos altamente pontuados que foram pouco referenciados e estão minimamente presentes nas universidades.

Em virtude dos resultados obtidos com essa abordagem, o *ranking* gerado foi descartado, pois o segundo método forneceu um resultado mais interessante.

4.4.2 Segunda Abordagem: Soma dos Prestígios

A segunda abordagem constrói a pontuação dos veículos iterativamente. Da forma como foi criada, a pontuação respeitará a reputação, mas quando se gera novamente o *ranking* das universidades a posição do *ranking* padrão não é respeitada. Primeiramente é mostrado o *ranking* de veículos de publicação obtido e em seguida o resultado da correlação entre o *ranking* gerado e o *ranking* padrão.

Nesta abordagem não existe um erro associado ao processo de pontuação, pois os veículos são pontuados diretamente, sem a necessidade de um processo de busca de um mínimo global. A Tabela 4.6 mostra os primeiros 20 veículos de acordo com esta ordenação e o apêndice E contém a listagem com os 100 primeiros veículos.

Tabela 4.6: *Ranking* de veículos baseado em reputação - segunda abordagem

Ponto Atribuído	Nome do Veículo	Número de Citações / Fator de Impacto (JCR)
5732,15	IEEE Transactions on Automatic Control	23227 / 3,293
3385,65	IEEE Conference on Decision and Control	...
2395,98	IEEE International Conference on Robotics and Automation	...
2268,90	SIAM Journal on Computing	4634 / 1,459
1328,88	ACM International Symposium on Computer Architecture	...
1196,15	IEEE Transactions on Computers	9240 / 2,611
1058,71	International Journal of Robotics Research	3472 / 2,882
974,39	Physical Review Letters	310717 / 7,180
973,92	IEEE Transactions on Signal Processing	15979 / 3,485
904,69	IEEE Transactions on Pattern Analysis and Machine Intelligence	24674 / 5,960
866,14	Neural Information Processing Systems	...
736,75	IEEE Symposium on Foundations of Computer Science	...
672,26	Communications of the ACM	12617 / 2,646
643,86	IEEE Transactions of Software Engineering	5449 / 3,569
619,98	IEEE Computer	...
600,31	American Control Conference	...
581,30	IFIP International Conference on Theoretical Computer Science	...
573,32	International Joint Conference on Artificial Intelligence	...
553,82	IEEE/ACM Transactions on Networking	6129 / 2,576
515,47	Automatica	12382 / 3,178
...	...	...

Como observado, o *ranking* não possui relação com o Fator de Impacto ou o total de citações. Parte dessa diferença é atribuída ao fato de ser considerado somente a área da Ciência da Computação para gerar a ordenação. A ordem dos veículos no *ranking* é considerada adequada, incluindo importantes periódicos, como por exemplo, *IEEE Transactions on Computers*, *Communications of the ACM* e *IEEE Computer*.

Utilizando as pontuações atribuídas aos veículos gerou-se uma nova ordenação das 60 universidades do *ranking* padrão. O método para o cálculo dos pontos de uma instituição seguiu o usado na primeira abordagem e foi explanado na Seção 3.4. O valor do coeficiente de correlação de Kendall foi de 0,43. Como este valor pode estar entre -1 e +1, então é equivalente à 71% de correlação. Apesar de o método de cálculo desta abordagem não tentar igualar o *ranking* resultante com o *ranking* padrão, foi obtido um valor de correlação bem acima da primeira abordagem que tem a premissa de "forçar" o *ranking*. Por essas razões o *ranking* de veículos baseado no algoritmo *Score* é mais fiel à ordenação do *ranking* padrão e tem qualidade superior ao obtido na primeira abordagem.

Uma última análise foi feita relacionando o *ranking* baseado em reputação (utilizando o algoritmo *Score*), o de abrangência e o de número de ocorrências. Para cada um dos 20 veículos da Tabela 4.6 foi buscado sua abrangência (número de universidades distintas em que ele ocorre) e seu número total de ocorrências. Estes dados estão na Tabela 4.7. Apesar dos seis primeiros veículos no *ranking* baseado em reputação serem uma permutação dos seis primeiros no *ranking* do número de ocorrências, existe uma grande variação da posição dos veículos ao longo dos *rankings*, como por exemplo, o veículo *Physical Review Letters* que varia entre as posições 8ª, 109ª e 449ª.

Tabela 4.7: Comparação entre o *ranking* do algoritmo *Score*, o de abrangência e o de número de ocorrências

Rank - Score	Nome do Veículo	Abrangência	Número de Ocorrências
1	IEEE Transactions on Automatic Control	14 (258ª)	344 (5ª)
2	IEEE Conference on Decision and Control	24 (74ª)	362 (3ª)
3	IEEE International Conference on Robotics and Automation	38 (12ª)	602 (1ª)
4	SIAM Journal on Computing	40 (8ª)	352 (4ª)
5	ACM International Symposium on Computer Architecture	35 (18ª)	291 (6ª)
6	IEEE Transactions on Computers	50 (1ª)	389 (2ª)
7	International Journal of Robotics Research	21 (106ª)	139 (36ª)
8	Physical Review Letters	10 (449ª)	76 (109ª)
9	IEEE Transactions on Signal Processing	14 (259ª)	108 (59ª)
10	IEEE Transactions on Pattern Analysis and Machine Intelligence	36 (16ª)	198 (21ª)
11	Neural Information Processing Systems	31 (35ª)	201 (18ª)
12	IEEE Symposium on Foundations of Computer Science	32 (28ª)	223 (14ª)
13	Communications of the ACM	38 (12ª)	136 (38ª)
14	IEEE Transactions of Software Engineering	35 (18ª)	223 (14ª)
15	IEEE Computer	41 (6ª)	182 (24ª)
16	American Control Conference	21 (106ª)	106 (61ª)
17	IFIP International Conference on Theoretical Computer Science	41 (6ª)	226 (12ª)
18	International Joint Conference on Artificial Intelligence	37 (14ª)	223 (14ª)
19	IEEE/ACM Transactions on Networking	39 (10ª)	231 (11ª)
20	Automatica	12 (350ª)	71 (117ª)

Capítulo 5

Conclusões e Trabalhos Futuros

O objetivo deste trabalho foi a criação de um *ranking* de publicações baseado na reputação. Os dados foram obtidos acessando as páginas Web dos professores dos departamentos/faculdades de Ciência da Computação das 60 principais universidades dos EUA. Através de um *ranking* padrão (escolhido dentre os estudados) pontuou-se os veículos utilizando duas abordagens: uma minimização em um sistema linear e um algoritmo de pontuação iterativo. Para ambos os métodos, o cálculo da pontuação de uma universidade é o somatório de multiplicações entre o valor de um veículo pelo número de vezes que se publicou nele.

O processo de obtenção dos nomes dos veículos de publicação mostrou-se adequado, mas ainda são necessárias melhorias, pois a variabilidade encontrada na forma de exibição de uma publicação foi muito grande. O aspecto importante é que as ferramentas de extração implementadas não dão falso positivo, ou seja, se ela reporta que encontrou determinado nome de veículo, ele com certeza está presente no texto da publicação.

Dessa forma, os dois primeiros *rankings* gerados (Tabelas 4.1 e 4.2) são válidos porque se é mostrado que um nome aparece em várias universidades ou é referenciado muitas vezes, realmente este nome estava escrito nos textos de publicação extraídos. O erro associado a esses dois *rankings* fica restrito à falta de um nome de veículo, ou seja, aqueles que não fora possíveis extrair, e que por essa razão não estarão presentes no *ranking*, e a um erro no agrupamento, que após algumas verificações manuais percebeu-se que erra em alguns casos especiais.

O agrupamento foi bem sucedido, mas como comentado acima, apresentou alguns erros. Um exemplo desse erro foi quanto ao agrupamento do veículo "Journal of the ACM". Todas variações deste veículo ficaram no mesmo grupo com exceção de alguns poucos (cinco nomes) que estavam fora, mas eram claramente do mesmo grupo. A grafia desses nomes era "j. acm" e não tinham a informação do acrônimo. Provavelmente por serem muito pequenos em relação ao original e por não terem um acrônimo extraído ficaram em

um grupo a parte. Esse foi um problema detectado, mas que não prejudicava a confiabilidade dos dados. Em geral o programa para agrupamento trabalhou adequadamente, inclusive para encontrar um nome padrão para representar o grupo.

Quanto ao *ranking* por reputação utilizando a primeira abordagem, o problema quanto a modelagem faz com que o resultado não seja seguro. Já o *ranking* obtido pela segunda abordagem reflete melhor a ordenação usada como base e sua metodologia transmite mais confiança ao analisar o resultado.

O caminho a ser percorrido para obter esse tipo de *ranking* passa por inúmeros problemas computacionais, cada um com uma grande complexidade associada. Portanto as ferramentas criadas e as análises feitas não são a solução definitiva para o problema, mas são boas fontes de informação para novas pesquisas.

De forma geral este trabalho mostrou a forte influência das conferências na área da Ciência da Computação e como essa influência vem crescendo ao longo dos anos. Essa informação fornece o suporte para mostrar que as métricas atuais devem contemplar outros veículos além dos indexados, visando qualificar melhor as instituições e seus pesquisadores. Além disso os resultados dão suporte para outras ferramentas que trabalhem sobre veículos de publicação, principalmente as que querem mostrar a valorização das conferências.

Uma primeira sugestão para um futuro trabalho é o refinamento das ferramentas de extração e a aplicação delas em outros contextos e bases de dados. Uma outra possível melhoria seria a criação de um programa para agrupamento próprio no qual se tem total confiança nos resultados apresentados e são conhecidos os erros, dessa forma pode-se mensurá-los.

A forma de cálculo da pontuação dos veículos, na segunda abordagem, poderia também ser melhorada. Uma possível modificação seria a mudança no cálculo do peso local do veículo. Ao invés de se utilizar diretamente o número de ocorrências, utiliza-se uma ponderação deste valor, como por exemplo, o número de ocorrências dividido pelo número de artigos já publicados naquele veículo. Dessa forma diferenciam-se os veículos que têm como característica o elevado volume de publicações dos que publicam um menor número de edições ou volume de artigos.

Uma última sugestão seria a aplicação de regras para expansão de palavras ou nome de veículos abreviados para melhorar a precisão dos resultados. Antes de qualquer processamento, sobre uma referência extraída, alguns termos poderiam ser trocados para auxiliar a obtenção do nome do veículo de publicação e o agrupamento dos dados. Algumas possíveis expansões seriam, por exemplo, trocar “J. ACM” para “Journal of the ACM”, “Conf.” para “Conference”, “Intl.” para “International”, e “Trans.” para “Transactions”. O sistema de identificação do nome do veículo e de agrupamento trabalharam adequadamente mesmo na presença de abreviações, mas essa melhoria poderia extinguir alguns pequenos problemas de agrupamento, como o caso do “j. acm”.

Apêndice A

Teste da Ferramenta *Tokenizer*

Neste apêndice consta os testes realizados para 92 referências extraídas da página do professor Zohar Manna do departamento de Ciência da Computação da Universidade de Stanford. A coluna chamada "Igual?" recebe uma letra para identificar a igualdade entre o veículo segmentado e o nome correto do veículo. Os marcados com "E" são os erros, os marcados com "P" são os acertos parciais e os marcados com "A" são os acertos exatos. Somente foi considerado acerto parcial se faltou ou está a mais um pequeno texto além do nome do veículo. Foram 79,35% de acertos exatos, 7,62% de acertos parciais e 13,04% de erros.

Tabela A.1: Professor Zohar Manna

Igual?	**Veículos Segmentados**	**Referências**
A	j. formal aspects of computing	M.Slanina, H.Sipma, Z.Manna, Deductive Verification of Alternating Systems. J. Formal Aspects of Computing, pp. 507-560 Vol 20, July 2008.
A	formal methods in computer-aided design	A. Bradley, Z. Manna, Checking Safety by Inductive Generalization of Counterexamples to Induction. Formal Methods in Computer-Aided Design (FMCAD), Vol. 7, 2007
A	symposium on logical foundations of computer science	Z. Manna, H. Sipma, T. Zhang, Verifying Balanced Trees. In Proceedings of the Symposium on Logical Foundations of Computer Science (LFCS), Lecture Notes in Computer Science, vol. 4514, pp. 363-378, Springer-Verlag, 2007.
E	colocated with	Cesar Sanchez, Henny B. Sipma and Zohar Manna A Family of Distributed Deadlock Avoidance Protocols and their reachable State Spaces, In Proceedings of the 10th International Conference on Fundamental Approaches to Software Engineering (FASE'07). colocated with ETAPS'07, vol. 4422 of Lecture Notes in Computer Science (LNCS), pp. 155-169, Springer-Verlag, 2007.
E	collocated with	Cesar Sanchez, Henny B. Sipma and Zohar Manna Generating Efficient Distributed Deadlock Avoidance Controllers, In The Fifteenth International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS 2007). collocated with IPDPS'07 (21st IEEE International Parallel and Distributed Processing Symposium), IEEE Computer Society Press, 2007.

A	10th international conference on principles of distributed systems	Cesar Sanchez, Henny B. Sipma, Christopher Gill, Zohar Manna, Distributed Priority Inheritance for Real-Time and Embedded Systems. In Proc. 10th International Conference on Principles of Distributed Systems (OPODIS), Lecture Notes in Computer Science 4305, Springer-Verlag, 2006, pp. 110-125.
A	international colloquium on theoretical aspects of computing	Matteo Slanina, Henny B. Sipma, and Zohar Manna, Proving ATL* Properties of Infinite-State Systems. In Proc. International Colloquium on Theoretical Aspects of Computing (ICTAC 2006), Lecture Notes in Computer Science, Vol. 4281, Springer Verlag, 2006.
A	international colloquium on theoretical aspects of computing	Aaron R. Bradley and Zohar Manna, Verification Constraint Problems with Strengthening. In Proc. International Colloquium on Theoretical Aspects of Computing (ICTAC 2006), Lecture Notes in Computer Science, Vol. 4281, Springer Verlag, 2006.
A	6th acm & ieee conference on embedded software	Cesar Sanchez, Henny B. Sipma, Zohar Manna, Christopher D. Gill, Efficient Distributed Deadlock Avoidance with Liveness Guarantees. In Proceedings of the 6th ACM & IEEE Conference on Embedded Software (EMSOFT'06), pp. 12-20, ACM Press, 2006.
A	information and computation	Ting Zhang, Henny B. Sipma, Zohar Manna, Decision procedures for term algebras with integer constraints, In Information and Computation, volume 204, pp 1526-1574, October 2006.
A	20th ieee int'l parallel and distributed processing symposium	Cesar Sanchez, Henny B. Sipma, Zohar Manna, Venkita Subramonian, and Christopher D. Gill, On Efficient Distributed Deadlock Avoidance for Distributed Real-Time and Embedded Systems, In Proc. of the 20th IEEE Int'l Parallel and Distributed Processing Symposium (IPDPS'06), IEEE Computer Society Press, 2006.
A	hybrid systems : computation and control	Sriram Sankaranarayanan, Henny B. Sipma, Zohar Manna, Fixed point iteration for computing the time elapse operator, In Proc. Hybrid Systems: Computation and Control (HSCC 2006), Lecture Notes in Computer Science. Vol. 3927, Springer Verlag, 2006.
P	verification	Aaron R. Bradley, Henny Sipma, and Zohar Manna, What's Decidable About Arrays?, In Proc. Verification, Model-Checking, and Abstract-Interpretation (VMCAI), LNCS, Vol. 3855, Springer Verlag, 2006 January 2006.
E	efficient strongly relational polyhedral analysis	Sriram Sankaranarayanan, Michael Colon Henny Sipma, Zohar Manna, Efficient Strongly Relational Polyhedral Analysis, In Proc. Verification, Model-Checking, and Abstract Interpretation (VMCAI), LNCS, Vol. 3855, Springer Verlag, 2006.
A	foundations of software technology and theoretical computer science	Ting Zhang, Henny B. Sipma, Zohar Manna, Decision Procedures for Queues with Integer Constraints, In Proc. Foundations of Software Technology and Theoretical Computer Science (FSTTCS) December 2005, Lecture Notes in Computer Science, Volume 3821, Springer-Verlag.
A	formal techniques for networked and distributed systems	Cesar Sanchez, Henny B. Sipma, Venkita Subramonian, Christopher Gill, Zohar Manna, Thread Allocation Protocols for Distributed Real-Time and Embedded Systems. In Proc. Formal Techniques for Networked and Distributed Systems (FORTE) October 2005, Lecture Notes in Computer Science, Volume 3731, Springer-Verlag.
A	formal techniques for networked and distributed systems	Cesar Sanchez, Matteo Slanina, Henny B. Sipma, Zohar Manna, Expressive Completeness of an Event-Pattern Reactive Programming Language. In Proc. Formal Techniques for Networked and Distributed Systems (FORTE) October 2005, Lecture Notes in Computer Science, Volume 3731, Springer-Verlag.
A	conference on algebra and coalgebra in computer science	Cesar Sanchez, Henny B. Sipma, Matteo Slanina, Zohar Manna, Final Semantics for Event-Pattern Reactive Programs, In Proc. Conference on Algebra and Coalgebra in Computer Science (CALCO) September 2005, Lecture Notes in Computer Science, Volume 3629, Springer-Verlag.

A	concurrency theory	Aaron R. Bradley, Henny Sipma, and Zohar Manna, Termination Analysis of Integer Linear Loops, In Proc. Concurrency Theory (CONCUR) August 2005, Lecture Notes In Computer Science, Volume 3653, Springer-Verlag.
A	conference on automated deduction	Ting Zhang, Henny B. Sipma, Zohar Manna, The Decidability of the First-order Theory of Knuth-Bendix Order, In Proc. Conference on Automated Deduction (CADE) July 2005, Lecture Notes in Computer Science, Volume 3632, Springer-Verlag.
P	automata	Aaron R. Bradley, Henny Sipma, and Zohar Manna, The Polyranking Principle, In Proc. Automata, Languages and Programming (ICALP) July 2005, Lecture Notes in Computer Science, Volume 3580, Springer-Verlag.
A	computer-aided verification	Aaron R. Bradley, Henny Sipma, and Zohar Manna, Linear Ranking with Reachability, In Proc. Computer-Aided Verification (CAV) July 2005, Lecture Notes in Computer Science, Volume 3576, Springer-Verlag.
A	12th international symposium of temporal representation and reasoning	Ben d'Angelo, Sriram Sankaranarayanan, Cesar Sanchez, Will Robinson, Bernd Finkbeiner, Henny B. Sipma, Sandeep Mehrotra, Zohar Manna, LOLA: Runtime Monitoring of Synchronous Systems, In Proc. of the 12th International Symposium of Temporal Representation and Reasoning (TIME 2005), pp. 166-174, IEEE Computer Society Press, 2005.
P	verification	Aaron R. Bradley, Henny Sipma, and Zohar Manna, Termination of Polynomial Programs, In Proc. Verification, Model-Checking, and Abstract-Interpretation (VMCAI) January 2005, Lecture Notes in Computer Science, Volume 3385, Springer-Verlag.
P	verification	Sriram Sankaranarayanan, Henny Sipma, Zohar Manna, Scalable Analysis of Linear Systems using Mathematical Programming, In Proc. Verification, Model-Checking, and Abstract Interpretation (VMCAI) January 2005, Lecture Notes in Computer Science, Volume 3385, Springer-Verlag.
A	17th international conference on theorem proving in higher order logics	Ting Zhang, Henny B. Sipma and Zohar Manna, Term Algebras with Length Function and Bounded Quantifier Alternation, In Proc. 17th International Conference on Theorem Proving in Higher Order Logics (TPHOL), Lecture Notes in Computer Science 3223, Springer-Verlag, 2004, pp. 321-336.
A	11th static analysis symposium	Sriram Sankaranarayanan, Henny B. Sipma and Zohar Manna, Constraint-based Linear Relations Analysis, In Proc. 11th Static Analysis Symposium (SAS'2004), Vol 3148 of Lecture Notes in Computer Science, Springer Verlag, pp 53-68, August 2004.
A	2nd international joint conference on automated reasoning	Ting Zhang and Henny Sipma and Zohar Manna, Decision Procedures for Recursive Data Structures with Integer Constraints, In Proc. 2nd International Joint Conference on Automated Reasoning (IJCAR'04), Vol 3097 of Lecture Notes in Computer Science, Springer Verlag, pp 152-167, July 2004. (Best Paper Award)
A	hybrid systems : computation and control	Sriram Sankaranarayanan, Henny B. Sipma, and Zohar Manna, Constructing Invariants for Hybrid Systems, In Hybrid Systems: Computation and Control (HSCC 2004), Lecture Notes in Computer Science, Volume 2993, pp. 539-554 (March 2004)
A	acm principles of programming languages	Sriram Sankaranarayanan, Henny B. Sipma, and Zohar Manna, Non-Linear Loop Invariant Generation using Grobner Bases, in ACM Principles of Programming Languages (POPL 2004). pp. 318-329 (January 2004)
A	formal methods at the crossroads : from panacea to foundational support	Zohar Manna and Calogero Zarba. Combining Decision Procedures. In Formal Methods at the Crossroads: from Panacea to Foundational Support, Lecture Notes in Computer Science, Volume 2787, Springer-Verlag, November 2003, pp. 381-422.

A	embedded software	Cesar Snchez, Sriram Sankaranarayanan, Henny B. Sipma, Ting Zhang, David Dill, and Zohar Manna, Event Correlation: Language and Semantics. In Embedded Software (EMSOFT 2003), Lecture Notes in Computer Science (LNCS) 2855, Springer-Verlag, pp. 323-339, 2003.
A	verification : theory and practice	Sriram Sankaranarayanan, Henny B. Sipma, and Zohar Manna, Petri-Net Analysis using Invariant Generation, In Verification: Theory and Practice, Lecture Notes in Computer Science(LNCS) 2772 (Invited Paper).
A	automated reasoning with analytic tableaux and related methods : position papers and tutorials	Calogero G. Zarba, Zohar Manna, and Henny B. Sipma, Combining theories sharing dense orders, In Automated Reasoning with Analytic Tableaux and Related Methods: Position Papers and Tutorials (2003).
A	theoretical computer science	Nikolaj Bjorner, Zohar Manna, Henny Sipma and Tomas Uribe. Deductive Verification of Real-time Systems using STeP. Theoretical Computer Science, (253) 2001, pp 27-60.
E	27th intl	Zohar Manna, Henny Sipma, Alternating the Temporal Picture for Safety. In Proc. 27th Intl. Colloq. Aut. Lang. Prog.(ICALP 2000). LNCS 1853, Springer-Verlag, pp 429-450.
A	formal methods in system design	Nikolaj Bjorner, Anca Browne, Michael Colon, Bernd Finkbeiner, Zohar Manna, Henny Sipma, Tomas Uribe. Verifying Temporal Properties of Reactive Systems: A STeP Tutorial. In Formal Methods in System Design, vol 16, pp 227-270.
E		Zohar Manna and Henny Sipma, Verification of Parameterized Systems by Dynamic Induction on Diagrams, CAV'99, pp 25-43, LNCS 1633, Springer-Verlag, 1999.
A	formal methods in system design	Henny B. Sipma, Tomas E. Uribe and Zohar Manna. Deductive Model Checking. Formal Methods in System Design, Vol 15, pp 49-74 (1999).
E	tool support for system specification	Zohar Manna, Nikolaj Bjorner, Anca Browne, Michael Colon, Bernd Finkbeiner, Mark Pichora, Henny B. Sipma, Tomas Uribe. An Update on STeP: Deductive-Algorithmic Verification of Reactive Systems. In Tool Support for System Specification, Development and Verification, Advances in Computing Science, pp 174-188, Springer Verlag, 1999.
P	hybrid systems : computation and control	Zohar Manna and Henny Sipma. Deductive Verification of Hybrid Systems using STeP. In Hybrid Systems: Computation and Control, Lecture Notes in Computer Science 1386, Springer-Verlag, 1998.
A	compositionality : the significant difference	Bernd Finkbeiner, Zohar Manna and Henny Sipma. Deductive Verification of Modular Systems. Compositionality: The Significant Difference, COMPOS'98, LNCS vol. 1536, pp 239-275, 1998.
E		Zohar Manna, Anca Browne, Henny Sipma, and Tomas Uribe. Visual Abstraction for Temporal Verification. In AMAST'98, Vol 1548 of LNCS, pp 28-41, Springer-Verlag, 1998.
A	requirements targeting software and systems engineering	Zohar Manna, Michael Colon, Bernd Finkbeiner, Henny Sipma, and Tomas Uribe. Abstraction and Modular Verification of Infinite-State Reactive Systems. Requirements Targeting Software and Systems Engineering (RTSE), LNCS 1526, Springer Verlag, pp 273-292, 1998.
A	technical report	Luca de Alfaro, Zohar Manna and Henny B. Sipma. Decomposing, Transforming and Composing Diagrams: The joys of Modular Verification, Technical Report STAN-CS-98-1614, Stanford University, 1998.
A	embedded systems	Yonit Kesten, Zohar Manna and Amir Pnueli. Verification of Clocked and Hybrid Systems. In Embedded Systems, LNCS, Springer-Verlag, 1998.
A	theoretical computer science	Nikolaj Bjorner, I. Anca Browne and Zohar Manna. Automatic Generation of Invariants and Intermediate Assertions. Theoretical Computer Science, vol. 173(1), pp. 49-87, February 1997. Original version appeared in 1st International Conference on Principles and Practice of Constraint Programming, Lecture Notes in Computer Science 976, Cassis, France, pp. 589-623, September 1995.

A	international conference on temporal logic	Nikolaj Bjorner, Uri Lerner, and Zohar Manna. Deductive Verification of Parameterized Fault-Tolerant Systems: A Case Study. In Proc. of International Conference on Temporal Logic, Kluwer, July 1997.
E		Nikolaj Bjorner, Zohar Manna, Henny B. Sipma and Tomas E. Uribe. Deductive Verification of Real-Time Systems Using STeP. In Proc. of ARTS'97, vol. 1231 of LNCS, pp. 22-43, Springer-Verlag, May 1997.
E		Luca de Alfaro, Zohar Manna, Henny B. Sipma, and Tomas E. Uribe. Visual Verification of Reactive Systems. In Proc. of TACAS'97, vol. 1217 of LNCS, pp. 334-350, Springer Verlag, 1997.
A	14th symposium on theoretical aspects of computer science	Luca de Alfaro, Arjun Kapur and Zohar Manna. Hybrid Diagrams: A Deductive-Algorithmic Approach to Hybrid System Verification. In Proc. of 14th Symposium on Theoretical Aspects of Computer Science, vol. 1200 of Lecture Notes in Computer Science, pp. 153-164, Springer Verlag, Feb. 1997.
A	second asian computing science conf.	Anca Browne, Zohar Manna and Henny B. Sipma. Hierarchical Verification using Verification Diagrams. In Second Asian Computing Science Conf., LNCS vol. 1179, pp. 276-286, December 1996.
A	8th international conference on computer-aided verification	Zohar Manna and the STeP group. STeP: Deductive-Algorithmic Verification of Reactive and Real-time Systems. In 8th International Conference on Computer-Aided Verification, LNCS vol. 1102, pp. 415-418, Springer-Verlag, July 1996.
A	8th international conference on computer-aided verification	Luca de Alfaro and Zohar Manna. Temporal Verification by Diagram Transformations. In 8th International Conference on Computer-Aided Verification, LNCS vol. 1102, pp. 287-299, Springer-Verlag, July 1996.
A	8th international conference on computer-aided verification	Henny B. Sipma, Tomas E. Uribe and Zohar Manna. Deductive Model Checking. In 8th International Conference on Computer-Aided Verification, LNCS vol. 1102, pp. 209-219, Springer-Verlag, July 1996.
P	cade- workshop on visual reasoning	Anca Browne, Luca de Alfaro, Zohar Manna, Henny B. Sipma and Tomas E. Uribe. Diagram-Based Formalisms for the Verification of Reactive Systems, in CADE-13 Workshop on Visual Reasoning, New Brunswick, NJ, July 1996.
A	hybrid systems iii	Yonit Kesten, Zohar Manna and Amir Pnueli. Verifying Clocked Transition Systems. In Hybrid Systems III, LNCS vol. 1066, pp. 13-40, Springer-Verlag, 1996.
A	technical report	Zohar Manna and Amir Pnueli. Clocked Transition Systems. A tribute to Prof. C.S. Tang on his 70th birthday. Stanford CSD Technical Report STAN-CS-TR-96-1566. In Logic and Software Engineering, pp. 3-42, World Scientific Pub., 1996.
A	15th conference on the foundations of software technology and theoretical computer science	I. Anca Browne, Zohar Manna and Henny Sipma. Generalized Temporal Verification Diagrams. In 15th Conference on the Foundations of Software Technology and Theoretical Computer Science, vol. 1026 of LNCS, pp. 484-498, Bangalore, India, December 1995.
A	technical report	Zohar Manna and the STeP group. STeP: The Stanford Temporal Prover (Educational Release), User's Manual. Technical Report STAN-CS-TR-95-1562, Computer Science Department, Stanford University, November 1995.
A	theoretical computer science	Nikolaj Bjorner, I. Anca Browne and Zohar Manna. Automatic Generation of Invariants and Intermediate Assertions. Theoretical Computer Science, vol. 173(1), pp. 49-87, February 1997. Original version appeared in 1st International Conference on Principles and Practice of Constraint Programming, Lecture Notes in Computer Science 976, Cassis, France, pp. 589-623, September 1995.
E	j. van leeuwen	Anuchit Anuchitanukul, Zohar Manna and Tomas E. Uribe. Differential BDDs. In J. van Leeuwen, ed, Computer Science Today , Lecture Notes in Computer Science, Vol. 1000, pp. 218-233, Springer-Verlag, Sep. 1995.

A	4th international conference on algebraic methodology and software technology	Luca de Alfaro and Zohar Manna, Verification in Continuous Time by Discrete Reasoning, 4th International Conference on Algebraic Methodology and Software Technology (AMAST), Montreal, Canada, pp. 292-306, July 1995.
A	theory and practice of software development	Zohar Manna and the STeP group. STeP: The Stanford Temporal Prover (2-page abstract). In TAPSOFT'95: Theory and Practice of Software Development, Lecture Notes in Computer Science 915, pp. 793-794, May 1995.
A	ieee symposium on logic in computer science	Edward Chang, Zohar Manna and Amir Pnueli. Compositional Verification of Real-time Systems. In IEEE Symposium on Logic in Computer Science, pp. 458-465, Paris, 1994.
A	international symposium on formal techniques real time and fault tolerant systems	Arjun Kapur, Thomas A. Henzinger, Zohar Manna and Amir Pnueli. Proving Safety Properties of Hybrid Systems. International Symposium on Formal Techniques in Real Time and Fault Tolerant Systems, Lecture Notes in Computer Science 863, pp. 431-454, Springer-Verlag, 1994.
A	international symposium on formal techniques real time and fault tolerant systems	Henny B. Sipma and Zohar Manna, Specification and Verification of Controlled Systems, International Symposium on Formal Techniques in Real Time and Fault Tolerant Systems, Lecture Notes in Computer Science 863, pp. 641-659, Springer-Verlag, 1994.
A	information and computation	Thomas A. Henzinger, Zohar Manna and Amir Pnueli. Temporal Proof Methodologies for Timed Transition Systems. Information and Computation, Vol. 112, No. 2, 1994, pp. 273-337. ??®A shorter version appeared in the 18th Annual ACM Symposium on Principles of Programming Languages, pp. 353-366, Orlando, Florida, January 1991.
A	rex symposium a decade of concurrency	Yonit Kesten, Zohar Manna and Amir Pnueli. Temporal Verification of Simulation and Refinement. In REX Symposium A Decade of Concurrency, Lecture Notes in Computer Science 803, pp. 273-346, Springer-Verlag, 1994.
A	first international conference on temporal logic	Hugh McGuire, Zohar Manna and Richard Waldinger, Annotation-Based Deduction in Temporal Logic. First International Conference on Temporal Logic, Lecture Notes in Computer Science 827, pp. 430-444, Springer-Verlag, 1994.
A	technical report	Zohar Manna and the STeP group. STeP: The Stanford Temporal Prover. Technical Report STAN-CS-TR-94-1518, Computer Science Department, Stanford University, July 1994.
A	specification and validation methods	Zohar Manna and Amir Pnueli. Verification of Parameterized Programs. In Specification and Validation Methods (E. Borger, ed.), Oxford University Press, pp. 167-230, 1994.
A	international symposium on theoretical aspects of computer software	Zohar Manna and Amir Pnueli. Temporal Verification Diagrams. In International Symposium on Theoretical Aspects of Computer Software, Lecture Notes in Computer Science 789, Springer-Verlag, pp. 726-765, 1994.
A	acta informatica	Zohar Manna and Amir Pnueli. Models for Reactivity. Acta Informatica, Vol. 30, pp. 609-678, 1993.
A	hybrid systems	Thomas A. Henzinger, Zohar Manna and Amir Pnueli. Towards Refining Temporal Specifications into Hybrid Systems. In Hybrid Systems, Lecture Notes in Computer Science 736, Springer-Verlag, pp. 60-76, 1993.
A	5th conference on computer aided verification	Yonit Kesten, Zohar Manna, Hugh McGuire and Amir Pnueli. A Decision Algorithm for Full Propositional Temporal Logic. In 5th Conference on Computer Aided Verification, Lecture Notes in Computer Science 697, Springer-Verlag, pp. 97-109, 1993.
A	hybrid systems	Zohar Manna and Amir Pnueli. Verifying Hybrid Systems. In Hybrid Systems, Lecture Notes in Computer Science 736, Springer-Verlag, pp. 4-35, 1993.

A	program design calculi	Zohar Manna and Amir Pnueli. A Temporal Proof Methodology for Reactive Systems. in Program Design Calculi, NATO ASI Series, Series F: Computer and System Sciences, Springer-Verlag, Vol. 118, 1993. ??®An extended version is also available.
A	19th international colloquium on automata	Edward Chang, Zohar Manna and Amir Pnueli. Characterization of Temporal Property Classes. In 19th International Colloquium on Automata, Languages, and Programming, Lecture Notes in Computer Science 623, Springer-Verlag, pp. 474-486, 1992.
A	rex workshop real-time : theory in practice	Thomas A. Henzinger, Zohar Manna and Amir Pnueli. Timed Transition Systems. In REX workshop Real-Time: Theory in Practice, Lecture Notes in Computer Science 600, Springer-Verlag, pp. 226-251, 1992.
P	19th international colloquium on automata	Thomas A. Henzinger, Zohar Manna and Amir Pnueli. What Good are Digital Clocks? In 19th International Colloquium on Automata, Languages, and Programming, Lecture Notes in Computer Science 623, Springer-Verlag, pp. 545-558, 1992.
A	rex workshop real-time : theory in practice	Oded Maler, Zohar Manna and Amir Pnueli. From Timed to Hybrid Systems. In REX workshop Real-Time: Theory in Practice, Lecture Notes in Computer Science 600, Springer-Verlag, pp. 447-484, 1992.
A	25th anniversary of inria	Zohar Manna and Amir Pnueli. Time for Concurrency. In 25th Anniversary of INRIA, Lecture Notes in Computer Science 653, Springer-Verlag, pp. 129-153, 1992.
E	logic	Zohar Manna and Richard Waldinger. Fundamentals of Deductive Program Synthesis. In Logic, Algebra, and Computation (F.L. Bauer, ed.), NATO Advanced Science Institutes Series, subseries F: Computer and System Sciences, Vol. 79, Springer-Verlag, 1991, pp. 41-107. Also in IEEE Transactions on Software Engineering, Vol. 18, No. 8 (August 1992), pp. 674-704.
A	technical report	Zohar Manna and Amir Pnueli. Temporal Specification and Verification of Reactive Modules. Weizmann Institute of Science Technical Report, March 1992.
A	logic and algebra of specifications	Edward Chang, Zohar Manna and Amir Pnueli. The Safety-Progress Classification. In Logic and Algebra of Specifications (F.L. Bauer, W. Brauer, and H. Schwichtenberg, eds.), NATO Advanced Science Institutes Series, Springer-Verlag, pp. 143-202, 1991.
A	cmu computer science : a 25th anniversary commemorative	Zohar Manna and Amir Pnueli. Tools and Rules for the Practicing Verifier . In CMU Computer Science: A 25th Anniversary Commemorative, (R.F. Rashid, ed.), ACM Press and Addison-Wesley, pp. 125-159, 1991.
A	mathematical foundations of computer science	Zohar Manna and Amir Pnueli. On the Faithfulness of Formal Models. In Mathematical Foundations of Computer Science, Lecture Notes in Computer Science 520, Springer-Verlag, pp. 28-42, 1991.
A	beauty is our business	Zohar Manna and Amir Pnueli. An Exercise in the Verification of Multi-Process Programs. In Beauty is Our Business (W.H.J. Feijen, A.J.M. van Gasteren, D. Gries, J. Misra, eds.), Springer-Verlag, pp. 289-301, 1990.
A	9th symposium on principles of distributed computing	Zohar Manna and Amir Pnueli. A Hierarchy of Temporal Properties. In 9th Symposium on Principles of Distributed Computing, Quebec, Canada, pp. 377-408, Aug. 1990.
A	theoretical computer science journal	Zohar Manna and Amir Pnueli. Completing the Temporal Picture. In Theoretical Computer Science Journal, Vol. 83, No. 1, 1991, pp. 97-130. Also in 16th International Colloquium on Automata, Languages, and Programming, Springer-Verlag, Berlin, pp. 534-558, 1989.
E	linear time	Zohar Manna and Amir Pnueli. The Anchored Version of the Temporal Framework. In Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Lecture Notes in Computer Science 354, Springer-Verlag, Berlin, pp. 201-284, 1989.

Apêndice B

Ranking Baseado no Número de Ocorrências em Universidades

Neste apêndice consta o *ranking* de veículos de publicação baseado no número de universidades que referenciam aquele veículo. Exemplo: se 38 universidades referenciam o veículo "Communications of the ACM" então ele recebe 38 pontos. Esta pontuação desconsidera o número de vezes que aquele veículo foi referenciado, sendo apenas importante saber em quantas universidades diferentes ele aparece. Nesta listagem consta os veículos que aparecem em ao menos 22 universidades (totalizando aproximadamente 100 veículos), pois a lista geral é muito extensa.

Tabela B.1: *Ranking* Baseado no Número de Ocorrências em Universidades

RANK	**Nome do Veículo**
50	ieee transactions on computers
49	ieee international conference on distributed computing systems
44	international conference on parallel processing
43	ieee international parallel and distributed processing symposium
43	journal of parallel and distributed computing
41	ieee computer
41	ifip international conference on theoetical computer science
40	information processing letters
40	siam journal on computing
39	ieee infocom
39	ieee / acm transactions on networking
38	ieee international conference on robotics and automation
38	communications of the acm
37	international joint conference on artificial intelligence
37	ieee transactions on parallel and distributed systems
36	ieee transactions on pattern analysis and machine intelligence
36	ieee conference on computer vision and pattern recognition
35	ieee transactions of software engineering
35	acm international conference on supercomputing
35	international conference on very large data bases

35	acm international symposium on computer architecture
34	ieee transactions on knowlege and data engineering
34	acm symposium on principles of distributed computing
34	acm siam symposium on discrete algorithms
33	aaai conference on artificial intelligence
33	journal of the acm
33	acm symposium on parallel algorithms and architectures
32	international conference on software engineering advances
32	ieee international conference on communications
32	journal of computer and systems sciences
32	ieee journal of selected areas in communications
32	international journal of computer vision
32	ieee symposium on foundations of computer science
32	acm transactions on programming languages and systems
31	neural information processing systems
31	ieee international conference on network protocols
30	ieee transactions on systems
30	international parallel processing symposium
29	bioinformatics
29	artificial intelligence
29	international conference on pattern recognition
29	international conference on parallel architectures and compilation techniques
29	ieee symposium on security and privacy
29	ieee international conference on data engineering
28	journal of algorithms
28	acm computing surveys
28	ai magazine
28	design automation conference
28	acm sigplan symposium on programming language design and implementation
28	international conference on machine learning
28	allerton conference on communications
27	ieee international symposium on high performance distributed computing
27	computer networks
27	european conference on computer vision
27	acm transactions on graphics
26	journal of machine learning research
26	annual acm symposium on theory of computing
26	seventh siam coference on parallel processing for scientific computing
26	machine learning
26	acm symposium on theory of computing
26	ieee transactions on information theory
26	acm sigops symposium on operating systems principles
26	international symposium on high performance computer architecture
25	ieee computer graphics and applications
25	information and computation
25	journal of computational biology
25	ieee micro
25	acm conference on architectural support for programming languages and operating systems
25	ieee international conference on data mining
25	acm sigmod conference
25	international conference on information and knowledge management
25	ieee international conference on computer vision
25	j. of artificial intelligence research
24	usenix symposium on operating systems design and implementation

24	national conference on articial intelligence
24	ieee conference on decision and control
24	ieee internet computing
24	international symposium on microarchitecture
24	acm sigcomm
24	international journal on parallel programming
24	ieee iccd
23	ieee wireless and communications and networking conference
23	ieee rjs international conference on intelligent robots and systems
23	ieee internacional conference on acoustics
23	siam international conference on data mining
23	computer science
23	usenix networked and distributed system security symposium
23	algorithms
23	ieee transactions on mobile computing
23	technical symposium on computer science education
23	ieee transactons on visualization and computer graphics
23	ieee computer magazine
22	acm transactions on information systems
22	tenth workshop on hot topics in operating systems
22	ieee international conference on image processing
22	infocom
22	hawaii international conference on systems science
22	annual meeting of the association for computational linguistics
22	international conference
22	siam j. computing
22	international conference on dependable systems and networks
22	international conference on high performance computing
22	acm conference on computer and communications security
22	acm transactions on database systems
22	international workshop on quality of service

Apêndice C

Ranking Baseado no Número Total de Ocorrências

Neste apêndice consta o *ranking* de veículos de publicação baseado no número total de ocorrências. Nesta listagem consta apenas os veículos que foram referenciados no mínimo 79 vezes (totalizando aproximadamente 100 veículos), pois a lista geral é muito extensa.

Tabela C.1: *Ranking* de Veículos Baseado no Número Total de Ocorrências

Número de Ocorrências	**Nome do Veículo**
602	ieee international conference on robotics and automation
389	ieee transactions on computers
362	ieee conference on decision and control
352	siam journal on computing
344	ieee transactions on automatic control
291	acm international symposium on computer architecture
267	ieee conference on computer vision and pattern recognition
262	ieee infocom
252	ieee international conference on distributed computing systems
249	international conference on parallel processing
231	ieee / acm transactions on networking
226	ifip international conference on theoetical computer science
226	ieee international conference on data engineering
223	ieee transactions of software engineering
223	international joint conference on artificial intelligence
223	ieee symposium on foundations of computer science
222	journal of parallel and distributed computing
201	neural information processing systems
200	aaai conference on artificial intelligence
199	ieee international conference on computer vision
198	ieee transactions on pattern analysis and machine intelligence
193	acm siam symposium on discrete algorithms
184	ieee international parallel and distributed processing symposium
182	ieee computer
177	information processing letters

168	international conference on very large data bases
162	journal of computer and systems sciences
160	international conference on software engineering advances
158	national conference on articial intelligence
158	x94 acm transactions on graphics
154	ieee transactions on parallel and distributed systems
152	acm international conference on supercomputing
150	international conference on machine learning
149	acm symposium on theory of computing
145	annual acm symposium on theory of computing
139	acm symposium on principles of distributed computing
139	international journal of robotics research
136	communications of the acm
133	ieee journal of selected areas in communications
132	artificial intelligence
128	siggraph
125	international journal of computer vision
123	appl. phys. lett
123	journal of the acm
122	ieee transactons on visualization and computer graphics
120	information and computation
120	international parallel processing symposium
119	design automation conference
117	acm symposium on parallel algorithms and architectures
116	ieee internacional conference on acoustics
115	ieee international conference on communications
114	bioinformatics
113	ieee conference on computer vision and patt. recog
113	international symposium on microarchitecture
113	ieee transactions on information theory
112	ieee transactions on knowlege and data engineering
112	ieee international symposium on high performance distributed computing
110	ieee computer graphics and applications
108	ieee transactions on signal processing
107	international conference on computer aided design
106	american control conference
104	acm transactions on programming languages and systems
104	european conference on computer vision
103	real time systems symposium
102	j. chem. phys
102	algorithms
102	international conference on parallel architectures and compilation techniques
102	acm seventh international conference on knowledge discovery and data mining
100	ieee international conference on network protocols
98	technical symposium on computer science education
98	first symposium on networked systems design and implementation
96	ieee rjs international conference on intelligent robots and systems
96	applied physics letter s
96	siam j. computing
96	ieee symposium on security and privacy
93	acm sigmod internation conference on management of data
92	journal of algorithms
91	acm asia and south pacific design automation conference
91	acm conference on architectural support for programming languages and operating systems

90	advances in neural information processing systems
88	international conference on computer design
88	machine learning
88	acm conference on computer and communications security
88	international symposium on high performance computer architecture
86	ai magazine
85	ieee transactions on paral lel and distributed systems
85	annual joint conference of the ieee computer and communications societies
85	university of michigan
85	j. of artificial intelligence research
84	acm sigcomm
84	acm conference on embedded networked sensor systems
83	usenix networked and distributed system security symposium
82	acm / ieee design automation conference
82	ieee international conference on data mining
81	nature
81	ieee international symposium on circuits and systems
81	acm sigops symposium on operating systems principles
80	icde
80	journal of machine learning research
79	international conference on pattern recognition
79	ieee transactions on computer-aided design of integrated circuits and sustems
79	ieee design automation and test in europe
79	international conference on computer aided verification

Apêndice D

Ranking de Veículos Baseado em Reputação - Primeira Abordagem

Neste apêndice consta o *ranking* de veículos de publicação baseado na reputação de acordo com a primeira abordagem. Nesta listagem consta apenas os veículos que obtiveram mais de 100 pontos, totalizando 100 veículos no total.

Tabela D.1: *Ranking* de Veículos Baseado em Reputação - Primeira Abordagem

Ponto Atribuído	**Nome do Veículo**
16031.0447269	ieee computer society workshop on perceptual organization in computer vision
9511.74941369	sigact news
8654.28007318	international computer software applications conference
6637.29401967	ieee international conference on electronics
5292.21923047	usenix
4827.2029357	workshop on hot topics in measurement modeling of computer systems
4512.64733861	ieee international symposium on defect fault tolerance in vlsi systems
4307.92163076	acm workshop on security of ad hoc sensor networks
3851.12853508	ieee conference on computational complexity
3431.65529065	acm sigcomm workshop on hot topics in networks
3104.65137345	ieee computer graphics and applications
2707.05178624	cacm
2700.9148644	ieee international computer performance dependability symposium
2499.63833407	visualization handbook
2337.5067128	geometric modeling processing
2326.12595045	joint conference on empirical methods natural language processing very large corpora
2055.58389217	computers and operations research
2042.55824594	acm internet measurement conference imc
2038.64757633	large installation system administration conference
2002.66327968	international society for magnetic resonance medicine
1937.16338249	non-photorealistic animation rendering
1783.36089064	ieee confernce on sensor
1736.2133411	hot interconnect
1410.56425961	ima conference on mathematics of surfaces

1394.84981831	sigcomm workshop on experimental approaches to wireless network design analysis
1325.70388477	acm symposium on user interface software and technology
1318.23278269	ieee international conference on humanoid robots
1300.11503996	workshop on configuration
1244.37207027	integration, the vlsi journal
1102.36769064	international workshop on distributed algorithms
1009.55166496	imacs world congress on systems simulation scientific computation
668.96704748	international conference on computer vision systems
663.555346167	medicine meets virtual reality
634.072252942	real-time systems journal
608.987303561	annual foundations of software technology theoretical computer science
542.011110768	workshop on parallel distributed real-time systems
492.525103158	acm conference on hypertext
490.724519111	european conference on speech communication technology
442.52927746	j. comp. system sci.
434.655558377	simulation
401.097764085	interspeech
382.683799434	neuroimage
368.292069675	acm mobisys
347.255482446	ieee symposioum on new frontiers in dynamic spectrum access networks
333.985498947	ieee trans. mag.
315.505164448	ieee transactions on mobile computing
265.889910054	international conference on parallel and distributed computing and systems
258.958354896	workshop on system effects of logic soft errors
255.172820187	ieee transactions on communication
247.279245735	acm ieee joint conference on digital libraries
227.000040654	symp biocomput
225.942821577	ieee international conference on embedded real-time computing systems applications
208.911797399	international provenance annotation workshop
206.775843462	ieee computer society annual symposium on vlsi
196.15192728	conference in uncertainty in artifical intelligence
182.054297909	ieee control systems magazine
180.698046893	supercomputing conference
176.462061318	ieee international symposium on wearable computing
175.682366549	acm wireless networks
171.232947807	international conference on computer design
169.237952516	international symposium on mixed and augmented reality
163.700107075	ieee wireless communications
162.080575225	ieee asilomar conference on signals
161.296106495	human-robot interaction
155.014740645	acm sensys
150.619650505	international workshop on network operating system support for digital audio video
149.126656858	international conference on software engineering knowledge engineering
146.761027302	international conference on distributed computing systems
145.672050177	international workshop on qualitative reasoning
145.145507068	conf. on robotics automation
143.929609242	ieee international symposium on network computing applications
142.844311759	international conference on machine learning
140.958274307	international conferences on algorithmic learning theory
138.611387679	siam conference on geometric design
137.156362236	conference on innovative applications in artificial intelligence
131.980109239	international journal of climatology

127.946399669	international wireless internet conference
126.621432271	acm conference on computer communicatons security
126.444166067	international symposium on networks-on-chip
124.953632608	international conference on field service robotics
124.841714532	ieee real-time systems symposium
121.570992172	international conference on intelligent virtual agents
120.966007019	journal of field robotics
119.978130958	ieee transactions on acoustics
118.512392056	journal of mathematical imaging vision
118.17093579	international test conference
114.846189143	ieee journal of solid-state circuits vol.
114.569115691	ieee international symposium on biomedical imaging
113.927159152	international conference on email and anti spam
112.416553039	financial crypto
110.335836238	computer networks isdn systems
108.269140635	workshop on workstation operating systems
106.606019225	ieee workshop on mobile computing systems applications
105.773917707	ieee symposium on foundations of computer science
104.646368895	ieee international symposium on fault-tolerant computing
104.166022873	international symposium on computer performance modeling
103.99934949	ieee transactions on speech audio processing
102.628686737	ifip ieee international workshop on distributed systems operations and management
101.932715833	international workshop on languages and compilers for parallel computing
101.265428523	conference on natural language learning

Apêndice E

Ranking de Veículos Baseado em Reputação - Segunda Abordagem

Neste apêndice consta o *ranking* de veículos de publicação baseado na reputação de acordo com a segunda abordagem. Nesta listagem consta apenas os veículos que obtiveram mais de 169 pontos, totalizando 100 veículos no total.

Tabela E.1: *Ranking* de Veículos Baseado em Reputação - Segunda Abordagem

Ponto Atribuído	**Nome do Veículo**
5732.15289363	ieee transactions on automatic control
3385.65274092	ieee conference on decision and control
2395.98133107	ieee international conference on robotics and automation
2268.89752977	siam journal on computing
1328.88478446	acm international symposium on computer architecture
1196.15111208	ieee transactions on computers
1058.70968529	international journal of robotics research
974.397398785	phys rev letters
973.928500624	ieee transactions on signal processing
904.686106613	ieee transactions on pattern analysis and machine intelligence
866.141474183	neural information processing systems
736.757846451	ieee symposium on foundations of computer science
672.266597481	communications of the acm
643.858575971	ieee transactions of software engineering
619.987660451	ieee computer
600.307605807	american control conference
581.304097742	ifip international conference on theoetical computer science
573.325921493	international joint conference on artificial intelligence
553.820610047	ieee / acm transactions on networking
515.475579622	automatica
511.607050583	annual princeton conference on information sciences and systems
504.850108125	aaai conference on artificial intelligence
504.804655972	conference on decision and control
484.942073191	journal of the american mathematical society
483.550057774	lear algebra and its applications
478.878241232	system and control letters

474.817455255	american mathematical society pp
465.039908316	ieee international conference on computer vision
459.348516505	first symposium on networked systems design and implementation
456.233417988	cug conference
455.324051017	journal of parallel and distributed computing
448.291633106	ieee international conference on distributed computing systems
439.475307268	information processing letters
431.166263293	ieee conference on computer vision and pattern recognition
419.857374631	ieee international conference on data engineering
416.252603343	acm siam symposium on discrete algorithms
416.248361133	ieee infocom
409.437996818	acm symposium on theory of computing
408.726723947	acm transactions on graphics
400.881785381	national conference on articial intelligence
400.186060593	international conference on parallel processing
390.148115678	international conference on machine learning
382.177062919	international conference on very large data bases
376.383412737	artificial intelligence
369.686696813	acm sigcomm computer communication review
346.385287774	journal of computer and systems sciences
328.319693088	ieee international parallel and distributed processing symposium
323.299201062	acm conference on architectural support for programming languages and operating systems
316.258897232	annual acm symposium on theory of computing
316.134731551	usenix symposium on operating systems design and implementation
303.57646939	siggraph
296.8491642	ieee international symposium on high performance distributed computing
290.907755546	acm international conference on supercomputing
289.833010368	ieee computer graphics and applications
288.108039556	international journal of computer vision
284.656943333	siam j. computing
283.427150699	journal of the acm
283.402264022	ieee transactions on parallel and distributed systems
274.745001184	annual joint conference of the ieee computer and communications societies
273.369543625	acm symposium on principles of distributed computing
266.46893613	design automation conference
260.431057764	acm symposium on parallel algorithms and architectures
258.904655433	international symposium on high performance computer architecture
258.192113943	algorithms
252.629828493	conference on cooperation
252.035605428	ieee transactions on information theory
245.263134294	acm sigops symposium on operating systems principles
240.688852349	j. acm
237.505976024	acm sigmod international conference on management of data conference
236.011975843	international symposium on information processing in sensor networks
234.335937063	information and computation
233.733024913	ieee micro
233.597951784	acm conference on embedded networked sensor systems
227.312613053	appl. phys. lett
225.708591068	fourth workshop on hot topics in networks
225.39503943	bioinformatics
219.268332084	technical symposium on computer science education
218.722875817	research papers
218.222333239	ieee journal of selected areas in communications
217.906740566	international conference on software engineering advances

214.032597969	usenix networked and distributed system security symposium
210.12756465	acm transactions on programming languages and systems
209.977924125	acm sigmod internation conference on management of data
209.970741571	j. of artificial intelligence research
200.609023467	international symposium on microarchitecture
199.957312639	acm sigplan notices
195.612473875	ieee transactons on visualization and computer graphics
194.000920714	ieee journal of solid-state circuits vol
192.882481659	ieee international conference on network protocols
191.484426909	acm operating systems review
190.862572708	acm sigmod conference
185.885441151	ieee symposium on security and privacy
183.542042207	acm transactions graphics
182.638989125	machine learning
181.016090421	international conference on parallel architectures and compilation techniques
178.959707816	european conference on computer vision
178.137237497	advances in neural information processing systems
177.370699952	international conference on computer design
171.184437679	ieee transactions on paral lel and distributed systems
169.926616235	acm conference on computer and communications security

Referências Bibliográficas

[1] M. N. Murty A. K. Jain and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[2] H. Abdi. The Kendall rank correlation coefficient. *Encyclopedia of Measurement and Statistics. Thousand Oaks (CA): Sage*, pages 1–7, 2007.

[3] Anonymous. Academic ranking of world universities. http://www.arwu.org/rank2008/EN2008.htm, último acesso em 03/Fev/2009.

[4] Anonymous. The agony and the ecstasy: the history and the meaning of the journal impact factor. http://thomsonreuters.com/products_services/science/free/essays/history_of_journal_impact_factor/, último acesso em 15/Set/2009.

[5] Anonymous. Best value colleges for 2009 and how they were chosen. http://www.usatoday.com/news/education/best-value-colleges.htm, último acesso em 03/Fev/2009.

[6] Anonymous. City-data.com - stats about all us cities. http://www.city-data.com/, último acesso em 08/Jun/2009.

[7] Anonymous. Expected citation rates, half-life, and impact ratios. http://thomsonreuters.com/products_services/science/free/essays/expected_citation_rates/, último acesso em 15/Set/2009.

[8] Anonymous. History of citation indexing. http://thomsonreuters.com/products_services/science/free/essays/history_of_citation_indexing/, último acesso em 02/Out/2009.

[9] Anonymous. Ibge - cidades@. http://www.ibge.gov.br/cidadesat/topwi ndow.htm?1, último acesso em 08/Jun/2009.

[10] Anonymous. Journal citation reports. http://www.thomsonreuters.com/products_services/science/science_products/scholarly_research_analysis/research_evaluation/journal_citation_reports, último acesso em 24/Ago/2009.

[11] Anonymous. Ks01 usual resident population: Census 2001, key statistics for urban areas. http://www.statistics.gov.uk/StatBase/ssdataset.asp?vlnk=8271&Pos=2&ColRank=1&Rank=224, último acesso em 08/Jun/2009.

[12] Anonymous. The national survey of graduate faculty. http://www.cra.org/statistics/nrcstudy2/nrcsurvey.html, último acesso em 03/Fev/2009.

[13] Anonymous. Rankings - computer science. http://grad-schools.usnews.rankingsandreviews.com/best-graduate-schools/top-computer-science-schools/rankings/, último acesso em 02/Jul/2009.

[14] Anonymous. Sample size calculator - creative research systems. http://www.surveysystem.com/sscalc.htm, último acesso em 03/Fev/2009.

[15] Anonymous. The thomson reuters impact factor. http://thomsonreuters.com/products_services/science/free/essays/impact_factor/, último acesso em 21/Ago/2009.

[16] Anonymous. World university rankings 2008 - the - the top 200 world universities. http://www.timeshighereducation.co.uk/, último acesso em 03/Fev/2009.

[17] Kurt Bollacker, Steve Lawrence, and C. Lee Giles. CiteSeer: An autonomous web agent for automatic retrieval and identification of interesting publications. In Katia P. Sycara and Michael Wooldridge, editors, *Proceedings of the Second International Conference on Autonomous Agents*, pages 116–123, New York, 1998.

[18] Eli Cortez, Altigran S. da Silva, Marcos André Gonçalves, Filipe Mesquita, and Edleno S. de Moura. Flux-cim: flexible unsupervised extraction of citation metadata. In *JCDL '07: Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, pages 215–224, New York, NY, USA, 2007. ACM.

[19] Eli Cortez, Altigran S. da Silva, Marcos André Gonçalves, Filipe Mesquita, and Edleno S. de Moura. A flexible approach for extracting metadata from bibliographic citations. *Journal of the American Society for Information Science and Technology*, 60(6):1144–1158, 2009.

[20] L. N. de Castro. Análise e Síntese de Estratégias de Aprendizado para Redes Neurais Artificiais. Master's thesis, Dissertação (Mestrado) Universidade Estadual de Campinas-UNICAMP, 1998.

[21] C. Lee Giles, Kurt Bollacker, and Steve Lawrence. CiteSeer: An automatic citation indexing system. In Ian Witten, Rob Akscyn, and Frank M. Shipman III, editors, *Digital Libraries 98 - The Third ACM Conference on Digital Libraries*, pages 89–98, Pittsburgh, PA, June 23–26 1998.

[22] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, 1970.

[23] Abby Goodrum, Katherine W. McCain, Steve Lawrence, and C. Lee Giles. Scholarly publishing in the internet age: a citation analysis of computer science literature. *Inf. Process. Manage.*, 37(5):661–675, 2001.

[24] Yi Han, Seng Wai Loke, and Leon Sterling. Agents for citation finding on the world wide web. Technical Report 96/40, University of Melbourne, 1996.

[25] J. E. Hirsch. An index to quantify an individual's scientific research output. *Proceedings of the National Academy of Sciences*, 102(46):16569–16572, 2005.

[26] Herbert Van de Sompel Johan Bollen and Marko A. Rodriguez. Towards usage-based impact metrics: first results from the mesur project. In *JCDL '08: Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*, pages 231–240, New York, NY, USA, 2008. ACM.

[27] Jack P. C. Kleijnen and Willem J. H. Van Groenendaal. Measuring the quality of publications: new methodology and case study. *Information Processing and Management*, 36(4):551–570, 2000.

[28] J. R. Koza. Survey of genetic algorithms and genetic programming. In *WESCON/'95. Conference record. 'Microelectronics Communications Technology Producing Quality Products Mobile and Portable Power Emerging Technologies'*, pages 589–, Nov 1995.

[29] Steve Lawrence, Kurt Bollacker, and C. Lee Giles. Indexing and retrieval of scientific literature. In *CIKM '99: Proceedings of the eighth international conference on Information and knowledge management*, pages 139–146, New York, NY, USA, 1999.

[30] Michael Ley. The dblp computer science bibliography. http://dblp.uni-trier.de/, último acesso em 24/Set/2009.

[31] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444, 2002.

[32] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, 2001.

[33] Sridhar Nerur, Riyaz Sikora, George Mangalaraj, and VenuGopal Balijepally. Assessing the relative influence of journals in a citation network. *Commun. ACM*, 48(11):71–74, 2005.

[34] Michael Nielsen. Why the h-index is little use. http://michaelnielsen.org/blog/why-the-h-index-is-virtually-no-use/, último acesso em 14/Set/2009.

[35] T. Opthof. Sense and nonsense about the impact factor. *Cardiovasc Res*, 33(1):1–7, January 1997.

[36] Urubatan Rocha Pacheco. Análise de redes sociais em dados bibliográficos. Master's thesis, Universidade de Campinas, Dezembro 2008.

[37] T.A.S. Pardo and M.G.V. Nunes. Segmentação Textual Automática: Uma Revisão Bibliográfica. *Série de Relatórios Técnicos do Instituto de Ciências Matemáticas e de Computação - ICMC*, 185, 2003.

[38] David A. Patterson. The health of research conferences and the dearth of big idea papers. *Commun. ACM*, 47(12):23–24, 2004.

[39] Fuchun Peng and Andrew McCallum. Information extraction from research papers using conditional random fields. *Information Processing & Management*, 42(4):963 – 979, 2006.

[40] L. Pevzner and M.A. Hearst. A Critique and Improvement of an Evaluation Metric for Text Segmentation. *Computational Linguistics*, 28(1):19–36, 2002.

[41] V. Prince and A. Ladadie. Text Segmentation Based on Document Understanding for Information Retrieval. *Lecture Notes in Computer Science*, 4592:295, 2007.

[42] Jie Ren and Richard N. Taylor. Automatic and versatile publications ranking for research institutions and scholars. *Commun. ACM*, 50(6):81–85, June 2007.

[43] Thomson Reuters. Isi web of knowledge. http://apps.isiknowledge.com/, último acesso em 24/Set/2009.

[44] Gerard Salton. Another look at automatic text-retrieval systems. *Commun. ACM*, 29(7):648–656, 1986.

[45] R. Shamir. The efficiency of the simplex method: A survey. *Management Science*, 33(3):301–334, 1987.

[46] Thomson. Isi highly cited. http://hcr3.isiknowledge.com/home.cgi, último acesso em 24/Set/2009.

[47] Masao Utiyama and Hitoshi Isahara. A statistical model for domain-independent text segmentation. In *ACL '01: Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 499–506, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[48] Lynn C. Hattendorf Westney. Historical rankings of science and technology: A citationist perspective. *The Journal of the Association of History and Computing*, 1(1), 1998.

[49] Jun Xu and Yalou Huang. Using svm to extract acronyms from text. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 11(4):369–373, 2007.

[50] R. Xu and Donald Wunsch II. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645, 2005.