Universidade Federal do Amazonas
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Programa de Pós-Graduação em Informática

Um Método Probabilístico para o Preenchimento Automatico de Formulários Web a Partir de Textos Ricos em Dados

Guilherme Alves Toda

Manaus – Amazonas
Março de 2010

Guilherme Alves Toda

Um Método Probabilístico para o Preenchimento Automatico de Formulários Web a Partir de Textos Ricos em Dados

Dissertação apresentada ao Programa de Pós-Graduação em Informática do Departamento de Ciência da Computação da Universidade Federal do Amazonas, como requisito parcial para obtenção do Título de Mestre em Informática. Área de concentração: Recuperação de Informação e Banco de Dados.

Orientador: Prof. Altigran Soares da Silva, Doutor

Guilherme Alves Toda

Um Método Probabilístico para o Preenchimento Automatico de Formulários Web a Partir de Textos Ricos em Dados

Dissertação apresentada ao Programa de Pós-Graduação em Informática do Departamento de Ciência da Computação da Universidade Federal do Amazonas, como requisito parcial para obtenção do Título de Mestre em Informática. Área de concentração: Recuperação de Informação e Banco de Dados.

Banca Examinadora

Prof. Altigran Soares da Silva, Dr. – Orientador
Departamento de Ciência da Computação – UFAM/PPGI

Prof. Alberto Henrique Frade Laender, Ph.D.
Departamento de Ciência da Computação – UFMG

Prof. João Marcos Bastos Cavalcanti, Ph.D.
Departamento de Ciência da Computação – UFAM/PPGI

Manaus – Amazonas
Março de 2010

Agradecimentos

Resumo

A solução mais comum atualmente para usuários interagirem com aplicações que utilizam banco de dados na Web é através do uso de formulários compostos por vários campos de entrada, como caixas de texto, listas de seleção e caixas de marcação. Apesar destes formulários serem efetivos e populares, em muitos casos, aplicações onde informações são fornecidas através de texto livre são geralmente preferidas pelos usuários.

Neste trabalho apresentaremos a proposta, a implementação e a avaliação de um novo método para preencher automaticamente formulários Web utilizando um texto rico em dados. Nossa solução toma como entrada um texto livre rico em dados (por exemplo, um anúncio), extrai seus dados implícitos e preenche os campos apropriados do formulário utilizando estes dados. Para essa tarefa, utilizamos o conhecimento obtido a partir de valores utilizados previamente pelos usuários para preencher os formulários. Nossa abordagem, chamada de *iForm*, utiliza características relacionadas ao conteúdo e ao estilo desses valores, que são combinadas através de uma Rede Bayesiana. Em nossos experimentos, mostramos que nossa abordagem é viável e efetiva, funcionando bem mesmo quando poucas submissões foram feitas ao formulário.

Palavras-chave: Extração de dados, Banco de Dados, Recuperação de Informação.

Abstract

On the Web of today the most prevalent solution for users to interact with data-intesive applications is the use of form-based interfaces composed by several data input fields, such as text boxes, radio buttons, pull-down lists and check boxes. Although these interfaces are popular and effectiver, in many cases, free text interfaces are preferred over form based ones.

In this work we present, the implementation and the evaluation of a novel method for automatically filling form-based input interfaces using data-rich text. Our solution takes a data-rich free text as input (e.g, an ad), extracts implicit data values from it and fills appropriate fields using them. For this task, we rely on knowledge obtained from values of previous submissions for each field, which are freely obtained from the usage of the interfaces. Our approach, called *iForm*, exploits features related to the content and the style of these values, which are combined through a Bayesian framework. Through extensive experimentation, we show that our approach is feasible and effective, and it works well even when only a few previous submissions to the input interface are available.

Keywords: Data Extraction, Data Base, Information Retrieval

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Caracterização do Problema/Motivação

Podemos encontrar na Web várias aplicações onde usuários casuais podem prover informações para serem adicionadas a um banco de dados visando um processamento futuro. Este é o caso de aplicações em que usuários postam currículos em *sites* de busca de emprego (ex, *CATHO*), introduzem meta-dados bibliográficos em bibliotecas digitais (ex, *DBLP*, *NDLDT*), postam ofertas em *sites* de anúncios (ex, *eBay*), anunciam eventos (ex, *DBWORLD*) e vários outros.

A solução mais comum nesses casos é disponibilizar formulários Web que contêm *vários campos de entrada de dados*, como caixas de texto, caixas de seleção, listas de seleção, caixas de marcação e outros mecanismos. Diferente dos formulários de busca mais populares na Web, os formulários Web de entrada de dados normalmente apresentam um grande número de campos.

A Figura 1.1 apresenta um formulário disponível em um *site* real que permite usuários postarem anúncios de celulares. Este formulário é composto por mais de 70 campos, incluindo listas de seleção, caixas de texto e campos de marcação. Assim, para colocar um anúncio de celular no *site*, os usuários precisam preencher todos os campos com as informações apropriadas. Tal tarefa, além de ser cansativa, é bastante sujeita a erros. Além disso, há certos casos em que muitos anúncios precisam ser submetidos ou o mesmo

anúncio é submetido a vários formulários, tornando o problema ainda mais complicado.

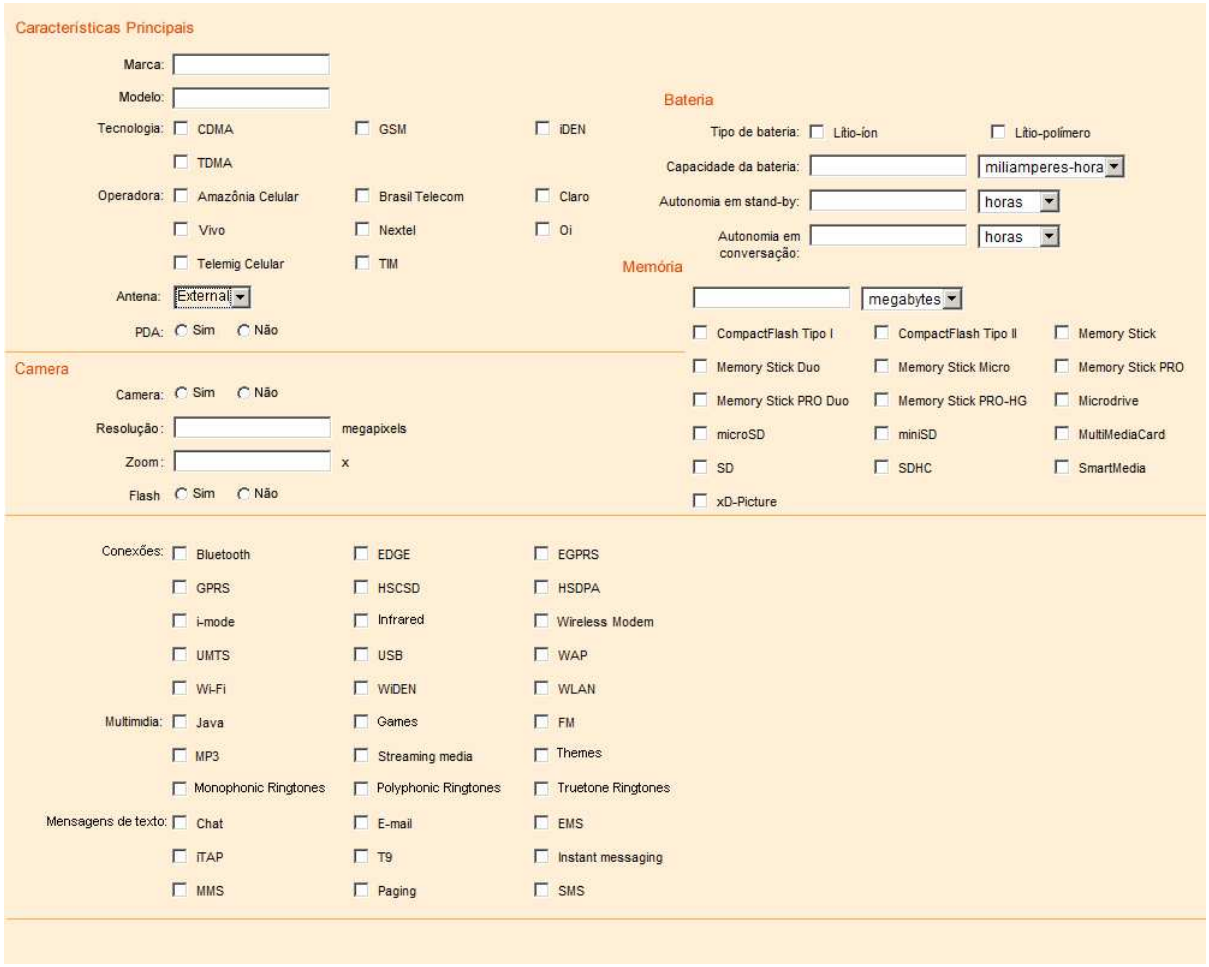


Figura 1.1: Formulário real de um anúncio de celular.

Apesar desses formulários serem populares e efetivos, em muitos casos formulários que aceitam como entrada um *texto rico em dados* [Embley et al., 1998], por exemplo, documentos ou pedaços de texto que contêm valores implícitos, são preferíveis. Na realidade, em vários casos os dados necessários para preencher os campos do formulário podem ser retirados de arquivos de textos nos quais já estão disponíveis. Por exemplo, um candidato a um emprego pode usar os dados disponíveis em seu currículo para preencher vários campos de formulários em diferentes *sites* de anúncios de empregos.

Figura 1.2: Exemplo de um texto rico em dados de anúncio de venda de celular.

1.2 Trabalho Proposto e Contribuições

Nesta dissertação apresentamos uma nova abordagem para o problema que consiste em um sistema que recebe como entrada um texto rico em dados (por exemplo, uma oferta ou anúncio), como ilustrado na Figura 1.2, e que é capaz de reconhecer os valores implícitos que sejam apropriados para preencher os campos de um formulário. Em aplicações práticas, o usuário pode verificar se os campos foram corretamente preenchidos pelo sistema e realizar as correções necessárias antes de enviar os valores ao banco de dados.

Esta situação é muito comum em *sites* de anúncios de comércio eletrônico (*e-commerce*) como *eBay*[1], *amazon.com*[2] e *TodaOferta.com*[3] que utilizam formulários para usuários registrarem ofertas. Nesses *sites*, podem existir diferentes formulários dependendo do produto oferecido. Por exemplo, nos experimentos apresentados neste trabalho, diferentes formulários foram encontradas nas categorias "veículos" e "celulares" no *site* de anúncios *TodaOferta.com*. Em alguns desses sites, como *eBay* e no próprio *TodaOferta*, os usuários podem entrar com as informações sobre o produto utilizando descrições genéricas retiradas de textos ricos em dados, preenchendo uma caixa de texto com essa informação, ou preencher um formulário estruturado com os detalhes do produto. Entretanto, quando os usuários deixam de preencher os dados de forma estruturada nos formulários, inviabilizam o uso e processamento destes dados diretamente por serviços de busca, mineração,

[1]http://www.ebay.com
[2]http://www.amazon.com
[3]http://www.todaoferta.com

recomendação e integração de dados.

Recentemente, alguns *sites* como *Craiglists*[4], *Googlebase*[5] e *Freebase*[6] foram oferecidos para permitir que usuários compartilhem e troquem dados em diferentes domínios (ex, receitas de cozinhas, estatísticas de esportes, etc.). Esses serviços mantêm bancos de dados que recebem dados diretamente dos usuários através da Web, por meio do preenchimento de formulários com múltiplos campos de entrada. Esses serviços também se beneficiariam de nosso método para ajudar os usuários a povoarem seus bancos de dados de uma forma simples e intuitiva.

Observamos que o problema de preencher formulários Web é mais complexo que os problemas tradicionais de extração de dados de informação (IE - Information Extraction [Sarawagi, 2008, Laender et al., 2002]), principalmente porque:

- Nos problemas de IE, exemplos de segmentos de texto a serem extraídos normalmente precisam ser rotulados manualmente como treino. No problema de preenchimento de formulário, este processo de treino pode ser inviável, uma vez que esperamos uma grande diversidade de formatos para os documentos de texto livre de entrada e rotular um conjunto representativo com tais documentos, pode requerer um grande esforço;

- No cenário de IE, todo segmento do texto de entrada precisa ser rotulado, até mesmo aqueles que não contêm nenhuma informação de interesse. Em nosso caso, é provável que nos documentos de texto ricos em dados contenham muitos segmentos com informações irrelevantes e exemplos com tais segmentos não são armazenados em submissões anteriores ao formulário.

Nosso método para resolver o problema de preencher automaticamente formulários, chamada de *iForm* [Toda et al., 2009], consiste em selecionar automaticamente segmentos do texto rico em dados e associá-los a campos apropriados do formulários. Nessa abordagem, os usuários fornecem um documento de texto (ou pedaços dele) como entrada e o

[4]http://www.craiglists.org
[5]http://www.google.com/base
[6]http://www.freebase.com

iForm automaticamente extrai os valores contidos para preencher o formulário, tomando como base as informações obtidas através dos valores previamente submetidos para cada campo. Para simplificar, documentos de texto rico em dados ou partes desses documentos, serão chamados de *texto de entrada* daqui em diante. Os usuários podem querer verificar o preenchimento do formulário feito por nosso método para realizar correções e então prosseguir com a submissão. Feito isso, os novos valores atribuídos a cada campo são armazenados e, assim, usados como uma evidência extra quando novos textos de entrada são fornecidos pelos usuários.

Como nossos experimentos demonstram, nossa abordagem funciona bem, mesmo quando poucas submissões prévias foram feitas ao formulário. Isto é conseguido usando um modelo que estima a probabilidade de cada campo no formulário, baseando-se nos valores usados anteriormente para preencher o formulário e no estilo dos termos que compõem esse valor, por exemplo, como uma palavra é escrita, nomes que começam com letras maiúsculas seguidas de letras minúsculas, etc.

Uma propriedade interessante relacionada à nossa estratégia para estimar tais probabilidades é que ela nos permite identificar corretamente segmentos no texto de entrada, que não necessariamente têm correspondência com nenhum dos valores previamente submetidos ao campo. Para isso, é necessário apenas que esses segmentos incluam termos típicos de um determinado campo ou tenham um estilo similar aos que foram submetidos ao campo.

Desse modo, o método iForm é flexível o suficiente para lidar com qualquer estilo de texto (por exemplo, pontuação, letras maiúsculas e minúsculas) ou estrutura (por exemplo, ordem dos valores implícitos). De fato, nosso método não utiliza características do texto, mas sim características dos campos com os seus valores. Além disso, neste trabalho também mostramos como lidar com regras impostas por um determinado formulário, como selecionar itens de uma lista de seleção ou selecionar e marcar as opções em uma caixa de marcação. Para desenvolvedores, utilizar nosso método não requer nenhum esforço extra além de construir o formulário.

Através da experimentação em um conjunto de dados reais, mostramos que nosso método é viável e efetivo, superando o estado-da-arte [Kristjansson et al., 2004] encontrado na literatura. Também mostramos que o método iForm tem um bom desempenho mesmo quando poucas submissões foram feitas ao formulário anteriormente.

1.3 Organização da Dissertação

Essa dissertação está organizada da seguinte maneira. O Capítulo 2 discute trabalhos relacionados. A seguir o Capítulo 3 descreve os detalhes do método proposto e o Capítulo 4 apresenta e analisa os resultados dos experimentos realizados para verificar a sua efetividade. Finalmente, o Capítulo 5 apresenta nossas observações finais, conclusões e trabalhos futuros.

Capítulo 2

Trabalhos Relacionados

Neste capítulo, apresentaremos os trabalhos relacionados. Inicialmente mostraremos os trabalhos relacionados ao problema de encontrar interfaces mais intuitivas que formulários, por exemplo, interfaces baseadas em palavras-chaves e interfaces para especificações de consultas em linguagem natural. Depois, discutimos as abordagens relacionadas ao problema geral de extração de dados. E por fim, apresentaremos os trabalhos relacionados ao problema de preenchimento de formulário.

2.1 Interfaces Baseadas em Palavras-Chaves e Especificação de Consultas em Linguagem Natural

Várias abordagens propostas na literatura recente têm tratado o problema de prover alternativas mais intuitivas para usuários acessarem bancos de dados Web do que interfaces baseadas em formulário. Essas soluções vão desde a tradução de consultas baseadas em palavras-chave até aquelas que possibilitam a especificação de consultas em linguagem natural. Uma abordagem que tem se tornado muito popular é traduzir consultas baseadas em palavras-chaves submetidas pelos usuários através de caixas de texto em consultas baseadas em formulários complexos [Calado et al., 2002] ou em comandos

SQL [Agrawal et al., 2002, Mesquita et al., 2007]. Outro problema bastante estudado é fornecer interfaces de linguagem natural para bancos de dados [Androutsopoulos et al., 1995, Li et al., 2005] e requisições de serviço [Al-Mumammed and Embley, 2007] (por exemplo, para encontrar um dentista e marcar uma consulta).

Diferente do método iForm, essas abordagem necessitam de um cuidadoso processo de configuração para os projetistas da interface e um conhecimento avançado das especificações da linguagem aceita pela interface. Por outro lado, enquanto esses sistemas suportam apenas especificações de consulta nosso método é especialmente adequado para o problema de *povoar* um banco de dados Web acessível apenas através de formulários. Além disso, enquanto nesses sistemas o usuário intencionalmente digita a consulta através de palavras-chave significativas ou pequenas frases, iForm aceita como entrada grandes porções de texto, como o apresentado na Figura 1.2. Tais porções de texto, normalmente têm dados significativos para preencher um formulário, mas não são necessariamente escritos para essa finalidade. Deste modo, várias palavras que não fazem sentido para preencher os campos do formulário podem aparecer no texto, por exemplo, preposições, verbos e artigos.

2.2 Abordagens para Extração de Dados

O problema de preencher automaticamente um formulário que resolvemos com o método iForm está relacionado ao problema de extração de dados de fontes textuais, descrito em diversos artigos. [Laender et al., 2002] e [Sarawagi, 2008], apresentam uma visão geral sobre este assunto, descrevendo várias abordagens que tratam o problema de extração de dados.

Nos métodos de indução de extratores (*wrapper induction*) apresentados nos trabalhos de [Muslea et al., 2001, Hsu and Dung, 1998], documentos de treino (por exemplo, páginas web) são fornecidos com exemplos de valores rotulados. A partir desses rótulos, as características das palavras que ocorrem nas vizinhanças desses valores são aprendidas pelo sistema. O extrator resultante funciona reconhecendo essas palavras e extraindo os

valores por elas delimitados em documentos de entrada similares aos fornecidos no treino.

De maneira similar funcionam os métodos de extração automática de dados apresentados em [Crescenzi et al., 2001], [Wang and Lochovsky, 2003] e [Reis et al., 2004], mas neste caso a extração dos valores de interesse é feita automaticamente a partir do reconhecimento nas páginas de entrada da estrutura do código HTML aprendida de páginas de exemplo fornecidas pelo usuário.

Outro trabalho relacionado é o RAPIER [Califf and Mooney, 1999], uma ferramenta que produz regras no formato de expressões regulares para preencher campos com valores de dados encontrados em textos. Na fase de treino, o RAPIER aprende: (1) um modelo do texto que ocorre antes do valor a extrair, (2) um modelo do texto que representa o valor a extrair, e (3) um modelo do texto que ocorre após o valor a extrair. Esses modelos são baseados em classes gramáticais e em léxicos com classes semânticas, extraídas do WordNet. Nosso método é mais flexível que o RAPIER, pois não necessita de modelos fixos obtidos no treino.

Mais próximas ao iForm são as abordagens probabilísticas para extração de informação de textos. Em [Freitag and McCallum, 2000] é apresentado um método que gera Hidden Markov Models (HMM) [Rabiner, 1989] independentes para identificar valores de atributos. A ferramenta DATAMOLD apresentada em [Borkar et al., 2001] também utiliza HMMs para extrair informações textuais não formatadas para realizar essa tarefa. O DATAMOLD necessita de uma fase inicial de treino em que um conjunto de exemplos seja manualmente rotulado. Essa abordagem probabilística obteve bons resultados em uma base real de endereços.

Modelos baseados em CRF (Condition Random Fields) [Lafferty et al., 2001] foram propostos como uma alternativa a HMM para o problema de extração de dados de fontes textuais. CRF têm sido utilizados para extrair informações em vários domínios como em [Peng and McCallum, 2006], onde dados bibliográficos são extraídos de artigos de pesquisa. Atualmente, devido a seus bons resultados [Mansuri and Sarawagi, 2006, Peng and McCallum, 2006], abordagens baseadas em CRF são consideradas o estado da

arte em extração de informação.

Como *iForm*, os trabalhos recentes [Mansuri and Sarawagi, 2006, Zhao et al., 2008] propõem que características podem ser aprendidas a partir de dados existentes em um banco de dados ou em tabelas de referência, reduzindo o treinamento manual. Nessa abordagem, entretanto, para características de contexto relacionado (ordem do campo e posicionamento) ainda é necessário um treinamento sobre o texto de entrada rotulado, o que não é necessário com iForm. Em [Zhao et al., 2008] os valores dos atributos presentes no texto de entrada precisam seguir uma ordem explicíta. Além disso, o método proposto por [Mansuri and Sarawagi, 2006] consegue tratar diferentes ordens dos valores dos atributos, mas necessita de um nível mínimo de regularidade em sua estrutura. Essa regularidade pode ser indisponível em alguns cenários de preenchimento de formulários Web, dado a diversidade dos formatos presentes no texto de entrada.

No problema abordado pelo método iForm, formulários podem ser preenchidos por vários usuários casuais da Web usando diferentes porções de texto, o que torna praticamente impossível encontrar uma estrutura uniforme. De tais textos, precisamos selecionar apenas alguns valores implícitos para preencher o formulário e tais valores são combinados com outras palavras não relacionadas no texto. Sob tais condições, o treinamento de exemplos do texto de entrada, necessário nos trabalhos baseados em CRF [Lafferty et al., 2001, Mansuri and Sarawagi, 2006, Zhao et al., 2008, Kristjansson et al., 2004] e em *Hidden Markov Models* (HMM) [Freitag and McCallum, 2000, Borkar et al., 2001], é inviável e pouco eficaz, devido à grande diversidade de estilos e formatos. Deste modo, em nosso método baseamos apenas nas informações dos valores de entrada submetidos pelo usuário para cada campo do formulário.

Outra abordagem recente relacionada ao problema de extração de informação é o FLUX-CiM [Vilarinho et al., 2007], que propõe um método flexível para extrair metadados de citações bibliográficas, como título e autores. Similar ao método iForm, esta abordagem também utiliza termos frequentes para identificar segmentos das citações. Porém, o FLUX-CiM assume que os valores no texto são delimitados por pontuações e

todos os segmentos no texto precisam ser extraídos.

No ONDUX [Cortez et al., 2010], outro método recente, o problema de extração de textos a partir de fontes textuais semi-estruturadas (por exemplo, informações bibliográficas e classificados) é tratado utilizando uma abordagem probabilística flexível que, como o método iForm, não necessita de um treino manual e se baseia apenas nas informações previamente disponíveis para associar segmentos do texto de entrada a atributos de um determinado domínio. Como nosso método, o ONDUX também utiliza estratégias eficientes de casamento de valores ao invés de se basear em estratégias de aprendizado de máquina. Entretanto, o ONDUX extrai todos os segmentos do texto de entrada. Já o método *iForm* tenta extrair apenas os segmentos com informações relevantes. Nos experimentos realizados, o ONDUX obteve bons resultados em diferentes domínios.

2.3 Problema de Preencher Automaticamente Formulários Web

Abordagens probabilísticas para extração de informação de texto livre baseadas em CRF foram também propostas como soluções para preencher formulários Web automaticamente [Kristjansson et al., 2004]. Essas abordagens se baseiam em características relacionadas ao contexto em que cada valor do atributo ocorre no texto de entrada, por exemplo, posicionamento e sequenciamento dos valores. Essas características geralmente são aprendidas por meio de um treino prévio sobre um conjunto representativo de textos de entrada, que são manualmente rotulados.

Um outro trabalho recente [Rukzio et al., 2008] lida com o problema de preencher automaticamente um formulário em dispositivos móveis. É proposto um arcabouço baseado em uma arquitetura de *proxy* para preencher formulários com dados que são normalmente requeridos em transações de comércio eletrônico utilizando dispositivos móveis. Nesse arcabouço, um banco de dados local móvel mantém valores submetidos por usuários em

vários formulários que são usados para preencher outros formulários. Essa abordagem consiste em gerar regras a partir de documentos Web (por exemplo, páginas HTML, XHTML, cHTML (i-Mode), WML/WAP) para identificar os campos a serem preenchidos nessas páginas. Assim, os formulários são preenchidos com informações armazenadas pelo usuário no dispositivo. Portanto, os valores já estão rotulados e o método extrai de um documento Web os campos a serem preenchidos. iForm, por outro lado, dado um formulário, extrai os valores de um texto livre como entrada.

Em resumo, iForm é um método probabilístico que, diferente das abordagens baseadas em CRF e HMM, não necessita de nenhum tipo de treinamento manual e se baseia apenas nas características relacionadas ao valores submetidos aos campos do formulário, conseguindo também associar valores que não foram submetidos anteriormente sem se basear na sequência ou ordem dos textos de entrada. Na prática, mesmo que o método iForm erre em algumas situações, o usuário pode corrigir e submeter as informações corretas ao banco de dados. Como é demonstrado através de experimentos, nosso método apresenta melhoras significativas quando comparada a outras abordagens de extração de textos para o preenchimento de formulários Web. t

Capítulo 3

O Método iForm

Neste capítulo, apresentaremos os detalhes do nosso método. Iniciamos com a apresentação de nossa modelagem para o problema de preenchimento automático de formulários Web. Em seguida, discutimos as características que nosso modelo leva em consideração para a seleção de segmentos de texto para o preenchimento de campos, e mostramos como estas características são combinadas. Finalmente, abordamos aspectos práticos da implementação do nosso método.

3.1 O Problema de Preenchimento de Formulários

O problema que abordamos nesta dissertação é preencher automaticamente os campos de um determinado formulário Web com os valores extraídos de um documento de texto rico em dados. Identificamos dentro deste problema dois subproblemas: (1) extrair valores do texto de entrada e (2) preencher os campos do formulário usando esses valores.

A maioria dos desafios do problema de preenchimento de formulários está relacionada ao subproblema, onde os valores adequados estão indistintamente presentes no texto juntamente com outras palavras não relacionadas. Além disso, não se pode assumir nenhum formato determinado ou ordem para esses valores.

Modelamos um *formulário* como um conjunto $\mathcal{F} = \{F_1, \ldots, F_n\}$, onde cada F_k é um *campo*. Consideramos três tipos de campo: caixas de texto, listas de seleção (*radio buttons*

são similares às listas de seleção em nossa solução) ou caixas de marcação. Cada campo é uma tripla $F = (l, t, c)$, onde l é um rótulo para o campo, t é o tipo do campo (ex, caixa de texto) e c é um conjunto de valores candidatos. No caso dos itens das listas de seleção, c corresponde à lista de itens disponíveis para seleção. Para as caixas de marcação, o conjunto c é {verdadeiro,falso}, identificando se a caixa está marcada ou não. Já para os campos de texto esse conjunto c é vazio.

No caso do iForm, a definição de um campo em um formulário é uma quádrupla $F' = (l, t, c, v)$, similar à tripla apresentada acima, onde é adicionado um conjunto v de valores previamente submetidos ao campo F'. Como no caso das caixas de marcação não existe símbolo submetido, pois os valores são booleanos, utilizamos o rótulo l para representar a submissão. Nesse caso $v = l$.

No caso dos campos textuais, consideramos que o conjunto v de valores previamente submetidos pelos usuários como um subconjunto representativo de seu domínio (ex, o conjunto de todos os valores possíveis). Além disso, é esperado que com os valores submetidos anteriormente possamos capturar uma fração de símbolos frequentes que podem aparecer em um determinado campo de texto. Como nossa avaliação experimental demonstra, isto permite que o nosso método utilize os símbolos disponíveis no conjunto de valores submetidos para reconhecer segmentos no texto de entrada e preencher corretamente os campos do formulário.

Documentos de texto livre são tratados como sequências de símbolos $t_1, \ldots, t_N$, representando palavras ou pontuações. A tarefa de extração executada pelo iForm consiste em identificar segmentos do documento de texto de entrada, por exemplo, uma sequência de símbolos contíguos, que são os valores adequados para os campos do formulário. Um segmento S_{ij} é composto por símbolos de $t_i, \ldots, t_j$, tal que $i \leq j$, $i \geq 1$ e $j \leq N$. Um conjunto válido de valores extraídos do texto de entrada deve respeitar duas condições: (1) apenas um único segmento pode ser associado a cada campo no formulário e (2) não pode haver nenhum conflito entre os segmentos extraídos, isto é, não podem existir segmentos S_{ab} e S_{cd} para $a < c$ tal que $b \geq c$. Ou seja, um mesmo símbolo t_i não pode ser

utilizado para preencher dois ou mais campos.

O método apresentado nesta dissertação consiste em estimar a probabilidade de um campo do formulário para cada segmento extraído do texto. Considere um texto de entrada I, composto por $N > 0$ símbolos (palavras). Seja S_{ab} um segmento, ou seja, uma sequência de símbolos em I que inclui os símbolos $t_a, t_{a+1}, \ldots, t_{b-1}, t_b$ $(0 < a \leq b \leq N)$. Consideramos S_{ab} como um valor adequado para o campo F se a probabilidade do campo dado esse segmento é maior que o limiar ϵ. Considerando L como o tamanho máximo do segmento, existem $N \times L$ segmentos em um texto com N símbolos[1].

O princípio por trás de nossa abordagem é utilizar as informações acerca dos valores usados anteriormente para preencher cada campo do formulário quando um novo texto é dado como entrada. Consideramos dois tipos de característica para esses valores: (1) os valores em si e os símbolos que compõem esses valores. Chamaremos estas características de *características relacionadas ao conteúdo*; e (2) o estilo, ou seja, o uso de letras maiúsculas ou minúsculas, pontuação, etc., que chamaremos de *característica relacionada ao estilo*. Vale ressaltar que nenhuma característica derivada do texto de entrada foi considerada em nosso método.

A seguir, discutiremos como essas características são usadas, e em seguida, apresentaremos a rede Bayesiana que combina essas características para computar a probabilidade entre um segmento e um campo. Finalmente, descrevemos nossos algoritmos para associar e mapear segmentos e campos baseando-se nessas probabilidades.

3.2 Características Consideradas

Nesta seção discutiremos os dois tipos de característica considerados pelo iForm, as *características relacionadas ao conteúdo* e as *relacionadas ao estilo*.

[1]Em nossos experimentos L não é maior que 10

3.2.1 Características Relacionadas ao Conteúdo

Consideramos que um segmento de texto pode ser usado para preencher um campo se seu conteúdo é similar ao conteúdo usado previamente para preencher esse campo. Para isso, utilizamos tanto os termos ou símbolos que compõem o segmento quanto o segmento como um todo. Essas características são capturadas através de funções de probabilidade como descrito a seguir.

Probabilidade Considerando Símbolos de um Segmento

Calculamos a *probabilidade* entre um campo F_j do formulário $\mathcal{F}$ e um segmento S_{ab} com base nos símbolos que compõem o segmento da seguinte forma:

$$TAF(F_j, S_{ab}) = \eta \sum_{\tau \in tokens(S_{ab})} \frac{\texttt{freq}(\tau, F_j)}{\sum_{F_i \in \mathcal{F}} \texttt{freq}(\tau, F_i)} \tag{3.1}$$

onde a função $tokens(S_{ab})$ retorna o conjunto de símbolos em S_{ab}, F_j é um campo em $\mathcal{F}$ e $\texttt{freq}(\tau, F_i)$ retorna a frequência do símbolo τ em um campo F_i, considerando os valores submetidos anteriormente pelo usuário do formulário $\mathcal{F}$. Assim, calculamos a probabilidade entre cada símbolo do segmento e um campo F_j divindo a frequência de τ nos valores submetidos pelos usuários a F_j pela frequência de τ em todos os campos do formulário. A intuição por trás dessa modelagem é que quanto mais concentradas forem as ocorrências anteriores de um símbolo em um campo, maior é a probabilidade do campo ser relacionado a tal símbolo.

Por fim, η é uma constante normalizadora [Pearl, 1988], cujo valor é dado por

$$\eta = \frac{1}{k + |avg(F_j) - k|} \tag{3.2}$$

onde k é o número de símbolos em S_{ab} e $avg(F_j)$ é o número médio de símbolos nos valores submetidos como entrada para o campo F_j em interações anteriores do usuário com o formulário. Assim, segmentos com tamanho menor que o número médio de símbolos do campo são penalizados. Isso faz com que valores extraídos apresentem um tamanho compatível com os valores típicos de seus respectivos campos.

Probabilidade Considerando Todo o Segmento

Calculamos também a *Probabilidade do Segmento* S_{ab} representar um valor de F_j com base na frequência do segmento inteiro nos campos, ao invés de comparar a frequência dos símbolos, dividindo a frequência de S_{ab} nos valores submetidos pelos usuários a F_j pela frequência de S_{ab} em todos os campos do formulário, conforme a equação abaixo.

$$VAF(F_j, S_{ab}) = \frac{\texttt{freq}(S_{ab}, F_j)}{\sum_{F_i \in \mathcal{F}} \texttt{freq}(S_{ab}, F_i)} \tag{3.3}$$

Utilizamos $TAF(F_j, S_{ab})$ e $VAF(F_j, S_{ab})$ para estimar a probabilidade de F_j dado S_{ab}, com base no *conteúdo* dos valores conhecidos de F. Esta probabilidade é combinada com a probabilidade de F_j dado S_{ab} com base nas características de estilo, descritas a seguir.

3.2.2 Características Relacionadas ao Estilo

Também consideramos como relevante para determinar quando um segmento pode ser usado para preencher um campo, a similaridade entre a forma como são escritos os símbolos do segmento e a forma como são escritos os valores submetidos ao campo.

Seja SV_j o conjunto dos valores submetidos anteriormente ao campo F_j. Para tratar as características relacionadas ao estilo, utilizamos um *Hidden Markov Model (HMM)* [Freitag and McCallum, 2000] $SM(F_j)$, chamado de *Modelo de Estilo de Valores*, que captura o estilo de escrita dos valores em SV_j. Este modelo é similar ao HMM usado em [Borkar et al., 2001] que também captura o estilo da escrita.

Para gerar um *Modelo de Estilo de Valores*, primeiro separamos cada valor de SV_j em símbolos (palavras) com base nos espaços entre eles. Em seguida, utilizando uma taxonomia similar à proposta em [Borkar et al., 2001], mostrada na Figura 3.1, codificamos esses valores como sequências de máscaras, representadas por expressões regulares, que representam o estilo dos caracteres encontrados nos símbolos. Chamaremos estas sequências de *Sequências de Máscaras de Símbolo*.

Por exemplo, considere o valor **"Edson Arantes"** do campo "Nome". Ao dividirmos

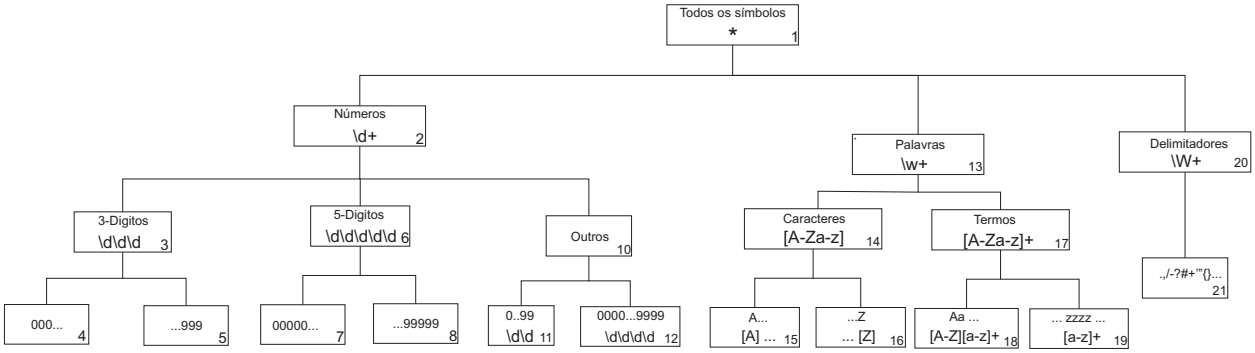


Figura 3.1: Exemplo da taxonomia de símbolos.

este valor com base nos espaços, teremos os símbolos **"Edson"** e **"Arantes"**. Desse modo, podemos codificar o primeiro símbolo como uma palavra que começa com uma letra maiúscula seguida de uma sequência de uma ou mais letras minúsculas, representada pela máscara de símbolo "[A-Z][a-z]+", mostrada no nó 18 da árvore de taxonomia na Figura 3.1, seguida também de uma palavra iniciada por uma letra maiúscula seguida de letras minúsculas, máscara "[A-Z][a-z]+". Portanto, para esse valor a máscara codificada gerada é **"[A-Z][a-z]+ [A-Z][a-z]+"**. Repetimos esse processo para todos os valores conhecidos de um campo, como é mostrado no exemplo da Tabela 3.1 para o campo "Nome". Note que a sequência de máscara de símbolos pode ser considerada como uma representação de estilo dos valores submetidos para cada campo.

Valores Conhecidos	Sequências de Máscaras Gerada
Edson Arantes	[A-Z][a-z]+ [A-Z][a-z]+
Gennaro I. Gattuso	[A-Z][a-z]+ [A-Z]. [A-Z][a-z]+
Van der Vaart	[A-Z][a-z]+ [a-z]+ [A-Z][a-z]+
Hugo Almeida	[A-Z][a-z]+ [A-Z][a-z]+
Robert Downey Jr.	[A-Z][a-z]+ [A-Z][a-z]+ [A-Z][a-z]+.
William H. Macy	[A-Z][a-z]+ [A-Z]. [A-Z][a-z]+

Tabela 3.1: Exemplo de máscaras geradas a partir de valores conhecidos para um campo do formulário.

Um grafo representando o *Modelo de Estilo dos Valores* $SM(F_j)$ é gerado com todas as *Sequências de Máscaras de Símbolo* encontradas em valores submetidos anteriormente ao campo F_j. Em $SM(F_j)$, cada nó é representado por uma *máscara de símbolo* que ocorre em SV_j. As arestas são formadas por pares ordenados $\langle n_x, n_y \rangle$ tais que o nó n_x é seguido

por n_y em alguma sequência de máscara de símbolos codificada nos valores SV_j. Assim, identificamos cada sequência de máscara de símbolos como um *caminho* em $SM(F_j)$. Além disso, o nó inicial do grafo contém arestas para os primeiros valores representados no grafo e o nó final é ligado por arestas que denotam a probabilidade de serem as últimas máscaras de seus respectivos valores.

Usando a abordagem de *probabilidade máxima* [Borkar et al., 2001], o *peso* de cada aresta em $SM(F_j)$ é calculado como:

$$w(SM(F_j), n_x, n_y) = \frac{\#\text{ de pares } \langle n_x, n_y \rangle \text{ em } SM(F_j)}{\#\text{ de pares } \langle n_x, n_z \rangle, \forall n_z \in SM(F_j)} \tag{3.4}$$

Portanto, o *peso* de uma aresta ligada aos nós n_x e n_y indica a probabilidade de uma *máscara de símbolo* n_x ocorrer seguida de uma máscara n_y. O grafo gerado com os valores conhecidos do campo "Nome", apresentados na Tabela 3.1, é mostrado na Figura 3.2.

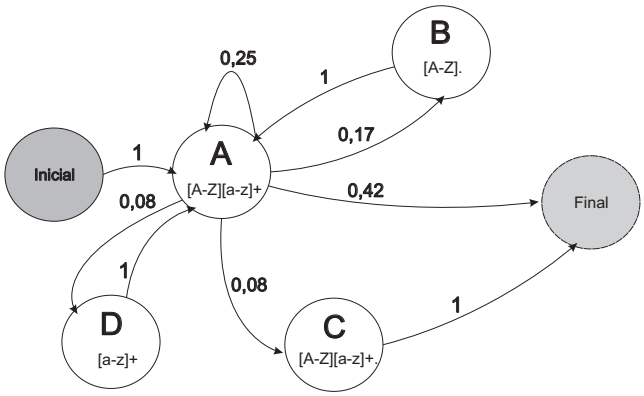


Figura 3.2: Modelo de Estilo de Valores gerado com as máscaras da Tabela 3.1.

Para realizar o casamento de um segmento S_{ab} com os valores de SV_j de acordo com o seu estilo de escrita, primeiro codificamos S_{ab} em uma *Sequência de Máscaras de Símbolo* utilizando a taxonomia apresentada acima. Então, computamos a probabilidade da *Sequência de Máscaras de Símbolo* de S_{ab} representar um *caminho* no grafo $SM(F_j)$, como:

$$style(SM(F_j), S_{ab}) = \frac{\sum_{\langle n_x, n_y \rangle \in caminho(S_{ab})} w(SM(F_j), n_x, n_y)}{\#\text{ de arestas em } caminho(S_{ab})} \tag{3.5}$$

onde $caminho(S_{ab})$ retorna o caminho percorrido no *Modelo de Estilo de Valores* $SM(F_j)$ associado ao segmento S_{ab} correspondente à *Sequência de Máscaras de Símbolo* de S_{ab}. Por exemplo, para um segmento S_x ="Giovanni dos Santos", sua *Sequência de Máscara de Símbolo* será "[A-Z][a-z]+ [a-z]+ [A-Z][a-z]+" e o $caminho(S_x)$ retornado no grafo $SM(F_j)$ da Figura 3.2 seria composto pelas arestas "$\langle Inicial, A\rangle, \langle A, D\rangle, \langle D, A\rangle, \langle A, Final\rangle$" , com uma probabilidade $style(SM(F_j), S_x) = (1 + 0,08 + 1 + 0,42)/4 = 0,625$.

Utilizamos $style(SM(F_j), S_{ab})$ para estimar a probabilidade de F_j dado S_{ab}, com base no *estilo* dos valores conhecidos em F_j.

3.3 Probabilidade de um Campo Dado um Segmento

Modelamos a computação da probabilidade do campo F_j dado o segmento S_{ab}, por meio de um modelo de Rede de Crença Bayesiana similar ao proposto por Ribeiro-Neto e Muntz[1996] que foi utilizado para classificar documentos em tarefas relacionadas a busca. Nossa rede Bayesiana fornece um formalismo gráfico para representar o modelo probabilístico desenvolvido, permitindo assim uma melhor visualização de como as características consideradas nos levam a uma probabilidade final do segmento dado o campo.

Nossa rede é ilustrada na Figura 3.3. Segmentos e campos de um formulário são representados na rede por símbolos, valores completos e estilos associados a eles. O nó S_{ab} no topo representa um segmento do texto de entrada. Na parte de baixo da figura, cada nó F_j modela um campo do formulário.

Os nós de V_1 a V_q representam os valores distintos submetidos anteriormente para os campos, os nós de T_1 a T_w modelam os símbolos que aparecem nesses valores e os nós de P_1 a P_r representam cada sequência de máscaras de símbolos distintos encontradas quando processamos os valores submetidos anteriormente a todos os campos. Os vetores $\mathbf{v}$, $\mathbf{t}$ e $\mathbf{p}$ são usados para referenciar todos os possíveis estados dos nós *raiz* de $V_1, \ldots, V_q$, $T_1, \ldots, T_w$ e $P_1, \ldots, P_r$, respectivamente.

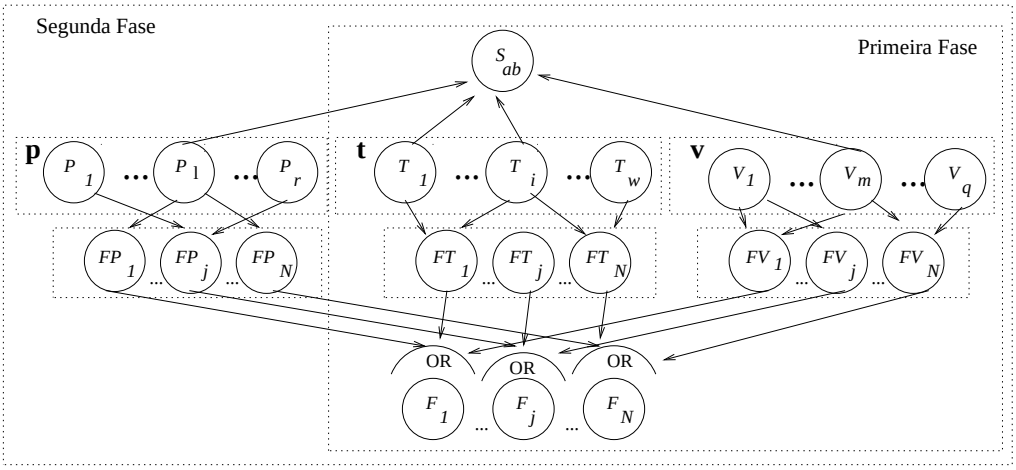


Figura 3.3: Rede de crenças para um segmento do texto S_{ab}, relacionado aos símbolos T_1 e T_i, atributo de valores completos V_m e caminho P_l derivados de seu estilo.

Nós de FV_1 a FV_N modelam a probabilidade de cada campo de F_1 a F_N, respectivamente, dada a ocorrência do valor no segmento S_{ab} (por exemplo, a palavra inteira que representa o segmento). Analogamente, os nós FT_1 a FT_N modelam a probabilidade do campo dada a ocorrência de um conjunto de símbolos distintos encontrados no segmento e os nós de FP_1 a FP_N modelam a probabilidade dos campos, dada a ocorrência do caminho a partir do *Modelo de Estilo do Valor*, casando a sequência de máscara do símbolo do segmento. A informação disponível nesses nós é combinada através de um operador *or* para computar a probabilidade final de cada campo dado o segmento S_{ab}, que é modelada pelos nós F_1 até F_N.

Cada nó X da rede é associado a uma variável aleatória *binária* descrita por X. Por exemplo, o nó S_{ab} é associado à variável aleatória *binária* S_{ab}. Seguindo essa notação, fica sempre claro quando estamos nos referindo ao segmento, ao nó na rede ou à suas variáveis binárias associadas. Para qualquer variável X no modelo, dizemos que é 1, denotados por x, indicando que o nó X está *ativo*.

Seguindo a rede Bayesiana descrita na Figura 3.3, consideramos a informação fornecida pelo campo para computar a probabilidade $P(FT_j|S_{ab})$, ou seja, a probabilidade que o nó FT_j está *ativado* dado que o nó S_{ab} é *ativo*. Pelo uso do modelo proposto

em [Ribeiro-Neto and Muntz, 1996], escrevemos:

$$P(FT_j|S_{ab}) = P(FT_j|T)\ P(S_{ab}|T)\ P(T) \tag{3.6}$$

onde T é o estado dos nós do símbolo raiz para cada símbolo ativo, que são exatamente aqueles no segmento S_{ab} e $P(S_{ab}|T)$ e $P(T)$, é definido como 1. Agora precisamos definir como computar o valor $P(FT_j|T)$, que deve ser relacionado à afinidade com que cada símbolo presente no segmento S_{ab} ocorre no campo F_j representado pelo nó FT_j. Tal probabilidade é modelada como:

$$P(FT_j|T) = TAF(F_j, S_{ab}) \tag{3.7}$$

Seguindo a rede da Figura 3.3, também calculamos o valor de $P(FV_j|S_{ab})$ como:

$$P(FV_j|S_{ab}) = P(FV_j|V)\ P(S_{ab}|V)\ P(V) \tag{3.8}$$

onde V é um estado que o único nó ativo é aquele associado ao valor encontrado em S_{ab}, $P(S_{ab}|V)$ e $P(V)$ é definido como 1, e $P(FV_j|V)$ é calculado analogamente como $P(FT_j|T)$, mas agora utilizando $VAF(F, S_{ab})$.

Por fim, também calcularemos o valor de $P(FP_j|S_{ab})$ como:

$$P(FP_j|S_{ab}) = P(FP_j|P)\ P(S_{ab}|P)\ P(P) \tag{3.9}$$

onde P é um estado onde o único nó ativo é aquele associado à sequência de máscara de símbolo derivada do segmento representado por S_{ab}, o que é gerado com a finalidade de derivar as sequências de máscara de símbolo de cada valor em cada um dos campo. Definimos a probabilidade à priori $P(P)$ e $P(S_{ab}|P)$ analogamente como $P(T)$ e $P(S_{ab}|T)$.

Agora podemos definir e calcular a probabilidade $P(FP_j|P)$ como o resultado do *Modelo do Estilo do Valor* de F_j ao processar o caminho associado ao segmento S_{ab}:

$$P(FP_j|P) = \begin{cases} 0 \quad \text{se } SM(F_j) \text{ não reconhece } caminho(P) \\ \\ \text{senão} \quad style(SM(F_j), P) \end{cases} \tag{3.10}$$

O resultado é zero se $caminho(P)$ não é reconhecido por $SM(F_j)$, ou seja, se ele não está contido em $SM(F_j)$ ou se suas máscaras iniciais e finais têm probabilidade zero com os nós iniciais e finais de $SM(F_J)$, respectivamente. Senão, $SM(F_j)$ intuitivamente verifica a probabilidade da sequência de máscaras seguida de um mesmo estilo de escrita dos valores conhecidos para esse campo, através da computação da média dos pesos das arestas que foram seguidas pelo $caminho(P)$.

A probabilidade condicional final $P(F_j|S_{ab})$ agora pode ser calculada usando o operador disjuntivo *or* sobre as probabilidades derivadas de cada característica. O processo de mapeamento descrito na próxima seção utiliza essas probabilidades em duas fases e a característica de estilo não é levada em consideração na primeira fase, como é mostrado na Figura 3.3. Assim, a probabilidade final para a primeira fase é calculada considerando apenas os símbolos e os valores, como é mostrado a seguir:

$$\begin{aligned} P(F_j|S_{ab}) &= 1 - (1 - P(FT_j|S_{ab})) \times \\ &\quad (1 - P(FV_j|S_{ab})) \end{aligned} \tag{3.11}$$

Se o estilo é levado em conta, as informações do conjunto de modelos de sequência precisam ser adicionadas ao cálculo. Nesse caso a equação resultante é:

$$\begin{aligned} P(F_j|S_{ab}) &= 1 - (1 - P(FT_j|S_{ab})) \times \\ &\quad (1 - P(FV_j|S_{ab})) \times \\ &\quad (1 - P(FP_j|S_{ab})) \end{aligned} \tag{3.12}$$

3.4 Como Evitar Computações Redundantes

A computação dos valores de $P(FT_i|S_{ab})$ para cada segmento S_{ab} possível leva a computações redundantes de várias probabilidades. Essas computações podem ser evitadas usando programação dinâmica. Considerando a Equação 3.1 sem a constante normalizadora, podemos definir mp_{ij}, como a matriz que contém as probabilidades de um campo F_j dado o segmento S_{ij}, conforme a seguir:

$$mp_{ij} = \begin{cases} p(T_i, F_k) & i = j \\ mp_{i(j-1)} + mp_{jj} & i < j \end{cases} \tag{3.13}$$

Considere o texto de entrada mostrado na Figura 3.4a e suponha que queremos verificar $P(FT_i|S_{ab})$ para S_{ab} ="Edson Arantes" e FT_i =`nome`. Seja $M_k : N \times N$ a matriz do campo F_k, onde o elemento da matriz mp_{ij} corresponde à probabilidade do segmento S_{ij} com F_k.

Primeiramente calculamos o caso mais simples, que é o dos elementos mp_{ij} tal que $i = j$. A Figura 3.4b mostra o resultado desse passo em nosso exemplo, onde $mp_{11} = p(T_1, F_k) = 1,0$, $mp_{22} = p(T_2, F_k) = 0,8$ e assim por diante. Estes valores foram calculados a partir de um conjunto hipotético de valores previamente fornecidos para F_k.

Então calculamos os elementos na primeira linha da esquerda para a direita e prossegue para as próximas linhas até que todos os elementos na matriz estejam definidos. A Figura 3.4c apresenta a matriz do exemplo após esse passo. Esse processo é repetido para as matrizes de cada campo.

Finalmente, aplicamos a Equação 3.1, sem a constante normalizadora η, para cada elemento em todas as matrizes a fim de calcular os valores da probabilidade final. Em nosso exemplo, o número médio de termos nos valores do campo `nome` é $2,5$. Portanto, os elementos da matriz que correspondem a segmentos com um único termo são multiplicado por $0,4$, como mostra a Equação 3.2. Para segmentos de tamanho 2,3,4 ou 5, multiplicamos por $0,4$; $0,29$; $0,18$ ou $0,13$, respectivamente. A propabilidade final para cada segmento em nosso exemplo é ilustrada na Figura 3.4d.

A partir da matriz de cada campo F_k, identificamos um conjunto P_k de pares segmento-

campo $\langle S_{ab}, F_k \rangle$ que são os valores potencialmente associados a F_k, ou seja, segmentos com probabilidade acima do limiar $\epsilon(0,4$ nesse caso). Por exemplo, os segmentos com probabilidade maior que $0,4$ são marcados por círculos na matriz do exemplo (Figura 3.4d). Nesse caso, os segmentos S_{12} = "Edson Arantes" e S_{13} = "Edson Arantes o" são estimados como *valores potenciais* para o campo `nome`.

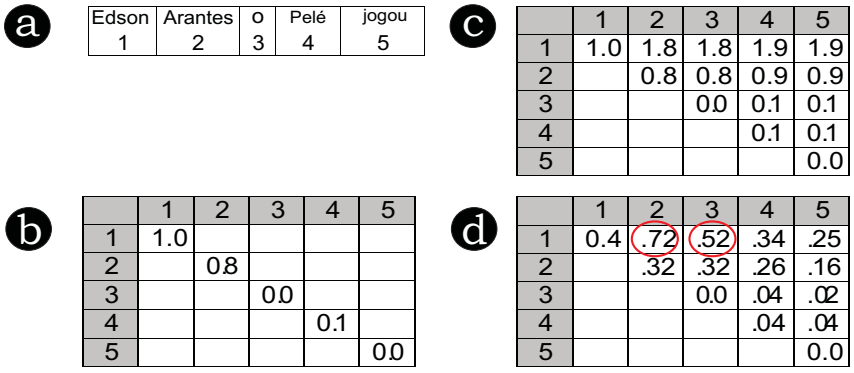


Figura 3.4: Exemplo de como nosso algoritmo calcula a afinidade de todos os segmentos de cada campo.

3.5 Mapeamento de Segmentos para um Campo

Seja C_j o conjunto de segmentos S_{ab} tal que $P(F_j|S_{ab})$ é maior que o limiar ϵ. Podemos dizer que C_j é um conjunto de *valores candidatos* para o campo F_j.

Nosso objetivo é encontrar um *mapeamento* $\mathcal{M}$ entre os valores candidatos e os campos no formulário tal que a probabilidade agregada tenha o valor máximo, satisfazendo as seguintes condições: (1) apenas um segmento é associado a cada campo e (2) os segmentos selecionados não sofrem sobreposição, isto é, não existe segmentos S_{ab} e S_{cd} para $a < c$ no mapeamento tal que $b \geq c$, ou seja, nenhum símbolo t_i pode ser utilizado para preencher dois ou mais campos. Isto pode ser feito seguindo o procedimento de duas fases descrito a seguir.

Na primeira fase, iniciamos pelo cálculo dos valores candidatos para cada campo F_j, baseado apenas nas características relacionadas ao conteúdo, ou seja, utilizando a Equação 3.11. Seja $\mathcal{I}$ um conjunto composto pela união dos conjuntos dos valores can-

didatos C_j para todos os campos F_j. Referimos a $\mathcal{I}$ como o *mapeamento inicial*, o qual contém os pares segmento-campo $\langle S_{ab}, F_j \rangle$. Assumimos que dois pares em $\mathcal{I}$ são *conflitantes* se eles violam alguma das condições acima. Portanto, o problema é encontrar o subconjunto de pares segmento-campo em $\mathcal{I}$ sem conflitos e cuja probabilidade agregada tenha valor máximo.

Para encontrar a solução ótima precisamos encontrar todos os possíveis subconjuntos – um número exponencial. Na prática, utilizamos uma heurística simples para encontrar um solução aproximada, mostrada no Algoritmo 1. Primeiro, extraímos o par com a maior probabilidade de $\mathcal{I}$ e verificamos se ela apresenta conflito com algum par em $\mathcal{I}$. Se o par não for conflitante, o adicionamos ao mapeamento final. Repetimos esse processo até que todos os pares em $\mathcal{I}$ forem extraídos. Isso marca o fim da primeira fase.

Algoritmo 1: Algoritmo para encontrar o mapeamento final.

```
Extrai o par segmento-campo com a maior probabilidade;
/* Verifica se ocorreu conflito */
se conflito(segmento,mapeamento) então
 | segmento é descartado;
senão
 | /* Verifica se o campo foi preenchido */
 | se preenchido(campo) então
 |  | segmento é descartado;
 | senão
 |  | /* Segmento é adicionado ao mapeamento final */
 |  | mapeamento ← segmento/campo;
 | fim se
fim se
```

Na segunda fase, se algum campo continua não mapeado por um segmento, utilizamos as probabilidades derivadas das características relacionadas ao estilo para tentar encontrar outras associações, utilizando a Equação 3.12 para calcular a probabilidade de cada campo dado um segmento. Então repetimos o processo de mapeamento, mas agora considerando apenas os pares dos segmentos e os campos que não foram mapeados na primeira fase.

Esse método de mapeamento com duas fases foi adotado após verificarmos através de experimentos que o estilo é menos preciso que as outras duas características utilizadas. Por outro lado, ainda é interessante utilizar as informações obtidas a partir do estilo

quando não há casamento de símbolos em um determinado campo. Assim, decidimos usar a informação de estilo como parte do processo de refinamento, realizada na segunda fase do mapeamento.

3.6 Preenchimento o Formulário

A ultima etapa de nossa abordagem consiste em utilizar o mapeamento final $\mathcal{M}$ para preencher os campos do formulário. No caso de caixas de texto, simplesmente preenchemos os campos com cada segmento mapeado a eles. Para as caixas de marcação, marcamos como verdadeiro os campos que foram mapeados em $\mathcal{M}$ e falso os demais.

No caso de listas de seleção, como os valores extraídos raramente são iguais aos itens da lista, este tipo de campo necessita de um trabalho a mais que será discutido a seguir.

Para selecionar um item em uma lista de seleção, procuramos encontrar um item tal que sua similaridade com o valor extraído seja máxima. Medimos a probabilidade utilizando uma versão "soft" da conhecida medida do cosseno, a softTF-IDF [Cohen et al., 2003]. Diferente da medida tradicional [Baeza-Yates and Ribeiro-Neto, 1999], a softTF-IDF não necessita que os termos casem exatamente e apresenta melhores resultados em nosso problema. A medida softTF-IDF também estima a similaridade entre os termos utilizando uma medida s de similaridade entre palavras. Deste modo, dado um valor A e um item da lista de seleção B, definimos $close(\theta, A, B)$ como o conjunto de pares de termos (a, b) onde $a \in A$ e $b \in B$, e tal que $s(a, b) > \theta$ e $b = \arg\max_{b' \in B} s(a, b')$; ou seja, b é um termo em B com a mais alta similaridade a a.

A similaridade entre um valor A e um item B em uma lista de seleção é definida por:

$$soft(A, B) = \frac{\sum\limits_{(a,b) \in close(\theta,A,B)} w(a, A) \cdot w(b, B) \cdot s(a, b)}{\sqrt{\sum\limits_{a \in A} w(a, A)^2} \cdot \sqrt{\sum\limits_{b \in B} w(b, B)^2}} \tag{3.14}$$

onde $w(a, A)$ e $w(b, B)$ são os pesos dos termos a e b relativo ao valor A e ao item B, respectivamente. $w(a, A)$ retorna 1 se a ocorrer em A ou 0, caso contrário. Para calcular $w(b, B)$ consideramos a frequência inversa do termo b na lista de seleção, ou seja,

$N_L/\texttt{freq}(b, L)$, onde N_L é o número de itens na lista de seleção L e $\texttt{freq}(b, L)$ é o número de valores em L que contêm o termo b.

Ou seja, a medida soft TF-IDF encontra um casamento aproximado entre o segmento extraído com algum valor da lista de seleção, por exemplo, para um campo gênero de filmes com os seguintes valores na lista: Ação, Comédia, Drama, Suspense e Terror. E o segmento extraído for "Ação/Aventura" a medida soft TF-IDF irá encontrar uma maior similaridade com o valor Ação e assim, nosso método irá preencher o campo com o valor Ação na lista de seleção.

Capítulo 4

Avaliação Experimental

Neste capítulo, apresentamos os resultados de experimentos conduzidos com o objetivo de avaliar a eficácia do método *iForm* na tarefa de preencher automaticamente formulários Web. O primeiro experimento avalia a sensibilidade do *iForm* com relação ao limiar ϵ, verificando a qualidade dos resultados quando variamos este limiar de $0,1$ até $0,9$. No segundo experimento testamos nosso método com formulários Web compostos por múltiplos tipos de campo, realizando submissões de *resenhas de filmes*, *ofertas de carros*, *ofertas de telefones celulares* e *ofertas de livros*. Também avaliamos como o número de submissões anteriores afeta a performance do *iForm* para cada caso. Por fim, realizamos experimentos com a coleção de *anúncios de empregos* [RISE, 1998], onde comparamos o *iForm* com um método baseado em CRF [Kristjansson et al., 2004].

4.1 Coleção Utilizadas

Em todos os experimentos realizados, simulamos a utilização de formulários Web reais onde um documento de texto livre é submetido de cada vez. Simulamos que os usuários manualmente verificam os resultados e, se necessário, corrigem os erros. Essa simulação é realizada de modo que após o método preencher os campos submetemos ao formulário os valores corretos, utilizando um gabarito, caso o método erre o preenchimento. Após essa iteração, a submissão será completada e os novos valores adicionados serão considerados

quando novas submissões forem processadas. Avaliamos o método verificando os erros encontrados em cada iteração.

	Submissões de teste		Submissões anteriores	
Coleção	#	Fonte	#	Fonte
Anúncio de empregos	50	RISE	100	RISE
Resenhas de filmes	50	Freebase and Wikipedia	10000	IMDb
Ofertas de carros	50	TodaOferta.com	10000	TodaOferta.com
Ofertas de celulares	50	TodaOferta.com	10000	TodaOferta.com
Ofertas de livros	50	TodaOferta.com	10000	Submarino.com

Tabela 4.1: Características de cada coleção utilizada nos experimentos

A Tabela 4.1 apresenta em detalhes cada coleção de dados usada em nossa avaliação experimental. A coluna "**Submissões de Teste**" mostra o número de documentos submetidos ao formulário. A coluna "**Submissões Anteriores**" refere-se ao número de submissões anteriores que simulamos em cada formulário. A coleção de *anúncio de empregos* [RISE, 1998] foi usada especificamente para a comparação com o método baseado em CRF [Kristjansson et al., 2004]. Como este *baseline* necessita de textos de entrada rotulados, utilizamos todos os 150 anúncios disponíveis no repositório RISE para realizar esse experimento. Isto explica porque seus números são diferentes das outras bases de dados, onde apenas 100 foram usadas para as submissões anteriores e as 50 restantes foram utilizadas para os testes. Em todas as coleções e experimentos apresentados aqui, não existe interseção entre os documentos de entrada de teste e os documentos de submissões anteriores.

4.2 Métricas para a Avaliação Experimental

Para avaliar os resultados de nossos experimentos utilizamos as conhecidas métricas de precisão, revocação e medida-F, bastante utilizadas em recuperação de informação e extração de dados. Aplicamos essas métricas para avaliar a qualidade do preenchimento de um único do campo e do formulário como um todo.

No caso das caixas de texto, calculamos precisão, revocação e medida-F por campo da seguinte maneira: seja A_i o conjunto de todos os termos de um texto de entrada que deve

ser usado para preencher um dado campo i no formulário. Seja S_i o conjunto de todos os termos do texto de entrada usados para preencher o campo i pelo iForm.

As medidas de precisão (P_i), revocação (R_i) e medida-F (F_i) para caixas de texto, lista de seleção e caixas de marcação são definidas como:

$$P_i = \frac{|A_i \cap S_i|}{|S_i|} \quad R_i = \frac{|A_i \cap S_i|}{|A_i|} \quad F_i = \frac{2(R_i.P_i)}{(R_i + P_i)} \tag{4.1}$$

A precisão por submissão, ou seja, a precisão de uma submissão inteira é calculada como a média das precisões de cada campo utilizado nessa submissão, observando que existem submissões em que nem todos os campos são preenchidos. Revocação e medida-F por submissão são calculadas da mesma maneira.

Para listas de seleção, o conjunto A_i contém o item na lista do campo i que deve ser escolhido e o conjunto S_i contém o item que foi escolhido para o campo i. No caso das caixas de marcação, A_i contém o valor booleano correto para o campo i e S_i contém o valor booleano que foi marcado para o campo i.

4.3 Variação do Limiar ϵ

Um parâmetro importante em nosso método é o valor do limiar ϵ. No Capítulo 3 consideramos um segmento S_{ab} como um valor adequado para o campo f se a probabilidade do campo dado esse segmento é maior que o valor desse limiar. Para estudar a utilização desse parâmetro, selecionamos aleatoriamente 25 documentos para cada base de dados, submetidos ao *iForm* variando o parâmetro ϵ de $0,1$ a $0,9$. Os resultados dos valores da medida-F por submissão obtidos estão ilustrados na Figura 4.1.

Podemos notar que os resultados podem variar de acordo com o domínio, sugerindo que um passo de treino pode ser útil para se obter bons resultados. Entretanto, nota-se que bons resultados foram obtidos quando utilizamos $\epsilon = 0,2$ com as entradas submetidas. Para a coleção *anúncio de empregos*, o melhor resultado foi obtido com $\epsilon = 0,5$. Isto pode ser explicado pelo pequeno número de documentos que simularam as submissões

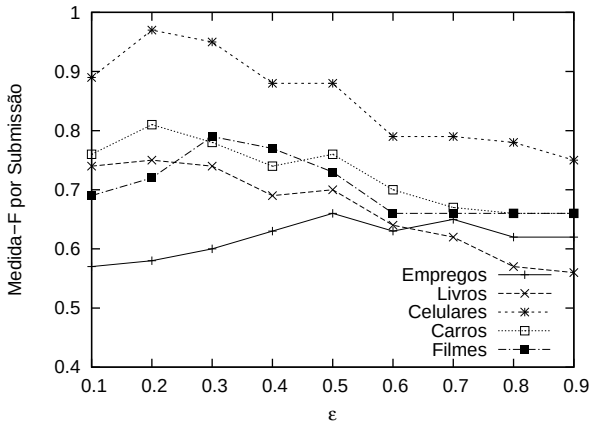


Figura 4.1: Resultados obtidos quando ϵ foi variado.

anteriores, o que requer um limiar mais restrito.

Para todos os outros experimentos aqui apresentados, utilizamos o valor de $0,2$ para ϵ, incluindo os experimentos com a coleção *anúncio de empregos*. Deixamos a possibilidade de introduzir um passo de treino para determinar um valor ótimo para este parâmetro como trabalho futuro.

4.4 Experimentos com Formulários Web com Múltiplos Tipos de Campos

Para avaliar o comportamento do método *iForm* com diferentes tipos de formulários Web de *sites* distintos, testamos nossa abordagem com submissões de *resenhas de filmes*, *ofertas de carros*, *ofertas de celulares* e *ofertas de livros*.

Para as coleções de *ofertas de carros*, *ofertas de celulares* e *ofertas de livros*, formulários com múltiplos campos e documentos de texto rico em dados foram obtidos do *site* de anúncios *TodaOferta.com*. Nesse *site*, usuários podem escolher entre preencher o formulário ou simplesmente submeter o anúncio como texto livre em uma caixa de texto. Para o teste, utilizamos as ofertas reais submetidas a essas caixas de texto para preenchermos automaticamente o formulário correspondente. No caso de *resenhas de filmes*, construímos um formulário Web, coletamos pequenas resenhas de filmes da Wikipedia e

Resenhas de Filmes

Tipo do campo	# Campos	P	R	F
Caixa de Texto	*4*	0,74	0,69	0,71
Valor por submissão		0,73	0,67	0,69

Ofertas de Carros

Tipo do campo	# Campos	P	R	F
Caixa de Texto	*5*	0,78	0,73	0,76
Caixa de marcação	*30*	0,79	0,79	0,79
Média		0,79	0,78	0,79
Valor por submissão		0,77	0,73	0,75

Ofertas de Celulares

Tipo do campo	# Campos	P	R	F
Caixa de Texto	*2*	0,89	0,69	0,78
Caixa de marcação	*35*	0,94	0,94	0,94
Média		0,94	0,93	0,93
Valor por submissão		0,96	0,94	0,95

Ofertas de Livros

Tipo do campo	# Campos	P	R	F
Caixa de Texto	*4*	0,88	0,67	0,76
Lista de Seleção	*1*	0,96	0,96	0,96
Média		0,90	0,73	0,80
Valor por submissão		0,89	0,67	0,76

Tabela 4.2: Resultados por campo e submissão para formulários web com múltiplos campos.

Freebase e simulamos as submissões anteriores com registros coletados do IMDb[1].

Os resultados deste experimento são apresentados na Tabela 4.2. Agrupamos os resultados por tipo de campo, ou seja, caixa de texto, caixa de marcação e lista de seleção, de acordo com a ocorrência em cada formulário. Os valores de precisão, revocação e medida-F são apresentados por submissão.

Como podemos notar, *iForm* alcança resultados de alta qualidade em todas as coleções de dados. No caso das ofertas de carros, a qualidade da tarefa de preencher o formulário foi praticamente a mesma tanto para os campos de texto como para os campos de marcação.

Melhores resultados foram obtidos com ofertas de celulares e livros, nas quais a medida-F média por campo alcançou valores acima de $0,80$. Como consequência, os valores por submissão da medida-F para essas coleções de dados foram maiores que 75%, o que significa que, na média, mais de 3/4 de cada submissão foram preenchidos corretamente no

[1]http://www.imdb.com

	Campo	Tipo do Campo	Ocorrência	P	R	F
1	Título	Caixa de Texto	50	0,80	0,56	0,66
2	Autor	Caixa de Texto	44	0,87	0,54	0,67
3	Editora	Caixa de Texto	45	0,84	0,57	0,68
4	Ano	Caixa de Texto	23	1,00	1,00	1,00
5	Encadernação	Lista de Seleção	14	0,96	0,96	0,96

Tabela 4.3: Resultados por atributo para a coleção Ofertas de Livros com múltiplos campos.

formulário Web. Uma inspeção mais detalhada nas ofertas que simulam as submissiões dos usuários nesses formulários, revela que os valores disponíveis nessas ofertas normalmente são mais uniformes que os valores de ofertas de carros e resenhas de filmes. Isto explica os excelentes resultados obtidos pelo iForm e corroboram nossos argumentos quanto à frequente re-utilização de textos ricos em dados que fornecem dados para preencher formulários Web.

No caso da coleção de resenhas de filmes, a inspeção das resenhas submetidas revela um alto grau de ambiguidade, o que é muito comum neste domínio. Por exemplo, existem atores que também são diretores ou diretores que também atuam. Além disso, títulos de filmes contêm palavras muito comuns que muitas vezes aparecem nas revisões e não estão indicando o título do filme (por exemplo, "Bad Boys") e cada resenha às vezes apresenta mais de um título. Outro fato é que frequentemente aparecem nomes e títulos onde todos os termos do segmento não ocorrem em submissões anteriores. Em tais casos, a característica de estilo realiza um papel importante na extração. Todos esses detalhes tornam essa coleção um verdadeiro desafio. Apesar disso, o *iForm* apresenta bons resultados. Como mostrado na Tabela 4.2, os valores por submissão da medida-F para essa coleção foram próximos a 70%.

Apresentamos também nas Tabelas 4.3, 4.4, 4.5 e 4.6 os resultados para cada campo dos formulários avaliados neste experimento. A coluna *ocorrência* indica o número de vezes que o campo deveria ser preenchido ou foi preenchido, mesmo que o preenchimento tenha sido incorreto. Nas Tabelas 4.3 e 4.4, notamos que o campo *Título* obteve os menores valores da medida-F, o que pode ser explicado devido à grande variação de termos e estilos

	Campo	Tipo do Campo	Ocorrência	P	R	F
1	Título	Caixa de Texto	50	0,39	0,32	0,35
2	Atores	Caixa de Texto	50	0,79	0,77	0,78
3	Diretores	Caixa de Texto	50	0,88	0,78	0,83
4	Gênero	Caixa de Texto	36	0,91	0,88	0,89

Tabela 4.4: Resultados por atributo para a coleção Resenhas de Filmes com múltiplos campos.

	Campo	Tipo do Campo	Ocorrência	P	R	F
1	Ano	Caixa de Texto	24	0,91	0,88	0,89
2	Versão	Caixa de Texto	50	0,52	0,49	0,5
3	Modelo	Caixa de Texto	44	0,90	0,88	0,89
4	Marca	Caixa de Texto	18	1,00	0,94	0,97
5	Cor	Caixa de Texto	15	0,58	0,47	0,52
6	Álcool	Caixa de Marcação	5	1,00	1,00	1,00
7	Gasolina	Caixa de Marcação	4	1,00	1,00	1,00
8	Air Bag Motorista	Caixa de Marcação	10	1,00	1,00	1,00
9	Alarme	Caixa de Marcação	2	1,00	1,00	1,00
10	Banco com ajuste	Caixa de Marcação	1	1,00	1,00	1,00
11	Computador de Bordo	Caixa de Marcação	7	1,00	1,00	1,00
12	Disqueteira	Caixa de Marcação	2	1,00	1,00	1,00
13	Rádio	Caixa de Marcação	4	0,75	0,75	0,75
14	CD player	Caixa de Marcação	6	0,67	0,67	0,67
15	Ar Quente	Caixa de Marcação	5	0,83	0,83	0,83
16	Rodas de Liga Leve	Caixa de Marcação	14	0,93	0,93	0,93
17	Teto Solar	Caixa de Marcação	1	1,00	1,00	1,00
18	Direção Hidráulica	Caixa de Marcação	15	0,79	0,79	0,79
19	Travas Elétricas	Caixa de Marcação	3	0,33	0,33	0,33
20	Vidros Elétricos	Caixa de Marcação	2	1,00	1,00	1,00
21	Limpador Traseiro	Caixa de Marcação	2	0,50	0,50	0,50
22	Retrovisores Elétricos	Caixa de Marcação	1	1,00	1,00	1,00
23	Faróis de Xenônio	Caixa de Marcação	4	1,00	1,00	1,00
24	Ar Condicionado	Caixa de Marcação	1	1,00	1,00	1,00
25	Desembaçador Traseiro	Caixa de Marcação	7	0,71	0,71	0,71
26	Encostos de Cabeça Traseiro	Caixa de Marcação	3	0,33	0,33	0,33
27	Bancos em Couro	Caixa de Marcação	1	1,00	1,00	1,00
28	Air Bags Laterais	Caixa de Marcação	2	1,00	1,00	1,00
29	Freio Abs	Caixa de Marcação	3	1,00	1,00	1,00
30	Controle Automático	Caixa de Marcação	4	0,00	0,00	0,00

Tabela 4.5: Resultados por atributo para a coleção Ofertas de Carros com múltiplos campos.

	Campo	Tipo do Campo	Ocorrência	P	R	F
1	Modelo	Caixa de Texto	33	0,81	0,42	0,56
2	Marca	Caixa de Texto	50	0,98	0,96	0,97
3	GSM	Caixa de Marcação	45	0,90	0,88	0,89
4	CDMA	Caixa de Marcação	1	1,00	0,94	0,97
5	MicroSD	Caixa de Marcação	26	0,58	0,47	0,52
6	MiniSD	Caixa de Marcação	2	1,00	1,00	1,00
7	Bluetooth	Caixa de Marcação	45	1,00	1,00	1,00
8	Modem Wireless	Caixa de Marcação	5	1,00	1,00	1,00
9	USB	Caixa de Marcação	40	1,00	1,00	1,00
10	WAP	Caixa de Marcação	32	1,00	1,00	1,00
11	Jogos	Caixa de Marcação	32	1,00	1,00	1,00
12	Reprodução de MP3	Caixa de Marcação	1	1,00	1,00	1,00
13	Rádio FM	Caixa de Marcação	41	1,00	1,00	1,00
14	E-mail	Caixa de Marcação	15	1,00	1,00	1,00
15	Entrada T9	Caixa de Marcação	2	1,00	1,00	1,00
16	MMS	Caixa de Marcação	48	1,00	1,00	1,00
17	SMS	Caixa de Marcação	17	1,00	1,00	1,00
18	Desbloqueado	Caixa de Marcação	40	1,00	1,00	1,00
19	TIM	Caixa de Marcação	38	1,00	1,00	1,00
20	SD	Caixa de Marcação	1	1,00	1,00	1,00
21	Edge	Caixa de Marcação	1	1,00	1,00	1,00
22	GPRS	Caixa de Marcação	3	1,00	1,00	1,00
23	Streaming Media	Caixa de Marcação	31	1,00	1,00	1,00
24	EMS	Caixa de Marcação	38	1,00	1,00	1,00
25	Bloqueio de Identificação	Caixa de Marcação	13	1,00	1,00	1,00
26	Transferência de Chamadas	Caixa de Marcação	19	1,00	1,00	1,00
27	Flip	Caixa de Marcação	37	1,00	1,00	1,00
28	Multimedia Card	Caixa de Marcação	1	1,00	1,00	1,00
29	EGPRS	Caixa de Marcação	2	1,00	1,00	1,00
30	Chat	Caixa de Marcação	2	1,00	1,00	1,00
31	Entrada iTap	Caixa de Marcação	2	1,00	1,00	1,00
32	Memory Stick Duo	Caixa de Marcação	2	0,5	0,5	0,5
33	HSCSD	Caixa de Marcação	2	0,59	0,59	0,59
34	HSDPA	Caixa de Marcação	9	1,00	1,00	1,00
35	UMTS	Caixa de Marcação	1	1,00	1,00	1,00
36	Memory Stick Micro	Caixa de Marcação	2	1,00	1,00	1,00
37	Memory Stick Pro Duo	Caixa de Marcação	22	1,00	1,00	1,00
38	Wlan	Caixa de Marcação	15	0,13	0,13	0,13
39	Wi-Fi	Caixa de Marcação	2	1,00	1,00	1,00
40	Memory Stick	Caixa de Marcação	3	0,00	0,00	0,00
41	Bateria Recarregável	Caixa de Marcação	12	1,00	1,00	1,00
42	Cabo USB	Caixa de Marcação	6	1,00	1,00	1,00
43	Software	Caixa de Marcação	3	1,00	1,00	1,00

Tabela 4.6: Resultados por atributo para a coleção Ofertas de Celulares com múltiplos campos.

que aparecem nas submissões para esses campos. Por outro lado, campos como *Gênero* no formulário de filmes da coleção *Resenhas de Filmes* e *Ano* e *Encadernação* para coleção *Ofertas de Livros*, que são campos com uma menor variedade entre os valores, obtiveram melhores resultados.

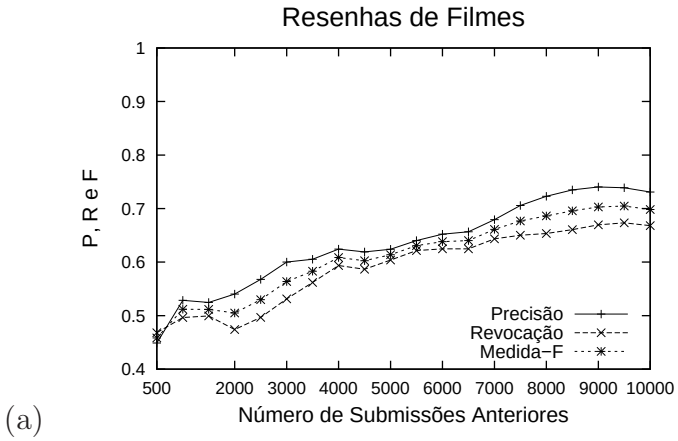
Nas Tabelas 4.5 e 4.6 é possível analisar como o *iForm* trata outros tipos de campo como as caixas de marcação. Para a coleção de *ofertas de carros* a maioria das caixas de marcação obteve preenchimento correto, mas em alguns casos, como nos campos 19, 20 e 22 e nos campos 21, 25 e 26, os termos "Elétricos" e "Traseiro", causaram ambiguidades levando a erros de preenchimento. O mesmo ocorre para os campos 32, 36, 37 e 40 da Tabela 4.6, pois os termos "Memory" e "Stick" aparecem em vários campos. No campo 30 da Tabela 4.5, foi obtido medidas iguais a zero devido à ambiguidade do termo "Automático" que muitas vezes é utilizado para representar o tipo de câmbio de um veículo, mas nesse caso o campo representa o "controle automático de velocidade" do automóvel.

Para os campos 34 e 38 da Tabela 4.6 os baixos valores obtidos são explicados por sentenças negativas descritas nos textos de entrada onde, no campo 38, treze das quinze ocorrências de valores para esse campo nos textos de entrada, indicavam que a função "**Wlan**" não estava disponível para o telefone celular anunciado. O mesmo ocorre para o campo 34 ("HSDPA").

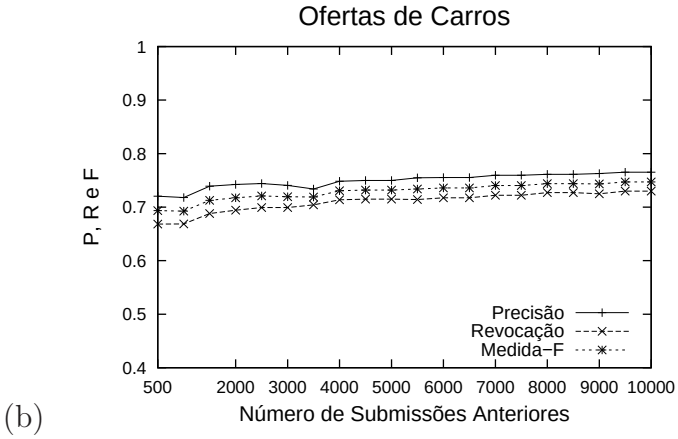
4.5 Variação do Número de Submissões Anteriores

Neste experimento, verificamos como o nosso método se comporta de acordo com o número de submissões anteriores disponíveis. O resultado deste experimento é apresentado na Figura 4.2. Para cada coleção, utilizamos um número crescente de submissões, que varia de 500 a 10000, e foi calculado a medida-F média por submissão para cada rodada de preenchimento sobre cada uma das coleções.

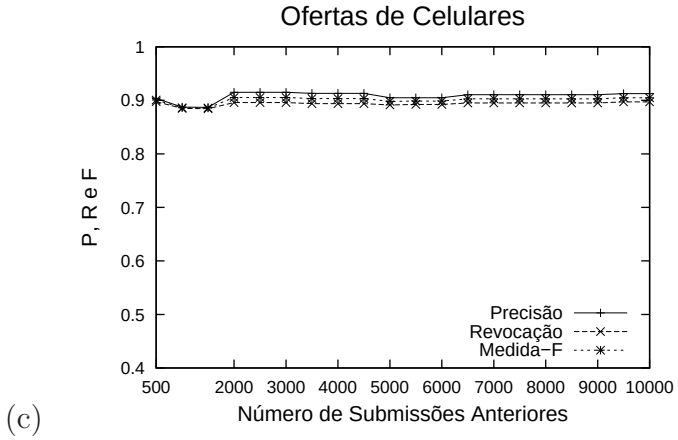
Como mostram as Figuras 4.2(a) e (d), para coleções de Resenhas de Filmes e Ofertas de Livros a qualidade alcançada pelo *iForm* aumenta proporcionalmente de acordo com



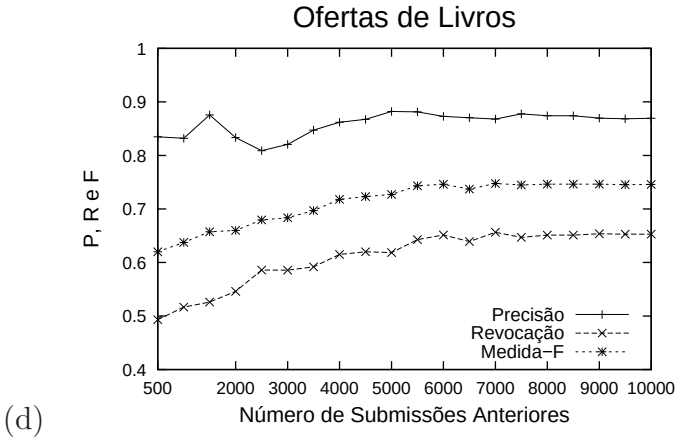
(a)



(b)



(c)



(d)

Figura 4.2: Comportamento da qualidade do preenchimento dos formulários com o aumento do número de submissões anteriores.

o número de submissões anteriores. Nas coleções de Ofertas de Carros e Ofertas de Celulares, apresentadas na Figuras 4.2(b) e (c), os valores da medida-F estabilizam com cerca de 3000 submissões anteriores e continuam os mesmos até 10000 submissões. Isto mostra que nosso método não necessita de muitas submissões anteriores para alcançar resultados de boa qualidade. Além disso, mesmo iniciando com poucas submissões o *iForm* é capaz de diminuir o esforço humano na tarefa de preencher o formulário.

4.6 Comparação com um Método Baseado em CRF

Em nosso último experimento, comparamos o *iForm* com um método baseado no modelo probabilístico CRF [Kristjansson et al., 2004] para o processo de extrair segmentos de textos de entrada e preencher um formulário.

Para esse experimento, utilizamos uma implementação pública disponibilizada por Sunita Sarawagi [2] que utiliza as mesmas características descritas em [Lafferty et al., 2001]: características de dicionário, função de cálculo da média das palavras e características de transição.

Utilizamos um sub-conjunto de 100 anúncios de emprego onde os segmentos a serem extraídos foram manualmente rotulados. Esses anúncios formam um conjunto adequado de treino para o método baseado em CRF, pois esse método precisa que exemplos dos valores a serem extraídos apareçam dentro do contexto em que ocorrem. Assim, não pudemos utilizar os 450 anúncios restantes da coleção RISE, pois os valores extraídos são fornecidos separadamente do anúncio em que ocorrem. Do mesmo conjunto de 100 documentos, utilizamos os segmentos rotulados para simular submissões ao formulário. Note que, diferente do método baseado em CRF, o *iForm* não necessita de textos de entrada rotulados para treino.

Em seguida, testamos os métodos utilizando um conjunto distinto de 50 documentos, onde os resultados da extração estavam disponíveis no RISE, nos permitindo verificar automaticamente os resultados. Esses resultados estão na Tabela 4.7 mostrando os valores

[2] `http://crf.sourceforge.net/`

Campo	iForm	CRF
Estado	**0,85**	0,81
Cidade	**0,70**	0,65
Linguagem	**0,84**	0,69
País	0,77	**0,87**
Formação Necessária	0,31	**0,59**
Plataforma	**0,47**	0,38
Título	**0,72**	0,49
Formação Desejada	**0,57**	0,37
Aplicação	**0,82**	0,37
Área	0,18	**0,23**
Empregador	**0,44**	0,22
Empresa	**0,41**	0,17
Salário	0,22	**0,25**
Média	**0,58**	0,48
Valor por Submissão	**0,59**	0,46

Tabela 4.7: Medida-F por campo e por submissão

da medida-F por campo e submissão.

De acordo com os resultados apresentados na Tabela 4.7, o método *iForm* obteve valores de medida-F superiores em nove campos, enquanto que o método baseado em CRF superou o *iForm* em apenas quatro campos. Estes valores estão indicados na tabela pelos números em negrito. A baixa qualidade dos resultados obtidos com CRF pode ser explicada pelo fato dos segmentos a serem extraídos de textos de entrada típicos, como anúncios de emprego, podem não aparecer em um contexto regular, o que é um requisito importante para CRF. Para o caso do iForm, esse contexto é menos importante, pois nosso método depende apenas das características relacionados aos *campos* ao invés de depender das características dos textos de entrada. Além disso, o iForm foi desenvolvido para explorar essas características relacionadas ao campo das submissões anteriores. Como vimos, para aplicar CRF a esse problema, um treinamento intensivo dos dados de exemplos representativos de texto de entrada é necessário.

Capítulo 5

Conclusão

Nesta dissertação apresentamos um método chamado *iForm* que utiliza valores de dados implícitos disponíveis em documentos de texto rico em dados. Propomos um novo método probabilístico para identificar valores implícitos em documentos de textos ricos em dados e preencher o formulário utilizando os valores.

Em nossa avaliação experimental, mostramos que o método *iForm* se comporta bem mesmo quando poucas submissões forem feitas aos formulários, também foram alcançados resultados satisfatórios para diferentes tipos de coleções e campos do formulário (ex, caixas de texto, listas de seleção e caixas de marcação) e por fim, mostramos que para o problema de preenchimento automático de formulários Web o método *iForm* obtêm melhores resultados que um método baseado no CRF, o estado-da-arte em extração de dados.

Assim, foi mostrado que nosso método pode ser útil como uma alternativa para preencher formulários automaticamente. Existem duas principais razões para essa conclusão: (1) qualquer formulário pode utilizar o iForm; (2) os usuários podem facilmente verificar se os resultados do iForm estão corretos e, caso contrário, corrigir os possíveis erros.

O iForm é um método escalável o suficiente para ser usado em várias aplicações onde necessita-se que os mesmos dados disponíveis em um texto de entrada sejam repetitivamente submetidos a diferentes formulários do mesmo domínio. Por exemplo, o mesmo anúncio de celular apresentado na Figura 1.2 pode ser usado em diferentes formulários de *sites* de anúncios de celulares.

Para trabalho futuro, investigaremos como ajudar os usuários a encontrar um formulário apropriado dado um texto de entrada. Isto inclui encontrar, entre um grande conjunto de formulários, quais deles podem responder à requisição do usuário. Esse problema necessita de uma solução eficiente e escalável para o cenário da Web. Nesse caso, seria interessante estender nosso método para lidar com formulários de consulta também.

O iForm assume que os valores do texto rico em dados são indicações positivas para preencher o formulário. Isto foi corroborado com uma inspeção realizada a fim de procurar por valores com referências negativas (por exemplo, "Modelo não é Fusca") em nossa coleção. Descobrimos que menos de 3,0% dos valores na coleção de *Ofertas deCelulares* eram negativos e que na coleção de *Ofertas de Carros* essa porcentagem é menor que 0,4%. Porcentagens similares foram encontradas nas outras coleções. Note que o usuário poderia facilmente corrigir o método nesses casos raros. Apesar disso, como trabalho futuro, pretendemos acoplar em nosso método uma solução para esse problema.

Referências Bibliográficas

[Agrawal et al., 2002] Agrawal, S., Chaudhuri, S., and Das, G. (2002). DBXplorer: A System for Keyword-Based Search Over Relational Databases. In *Proceedings of 18th the International Conference on Data Engineering*, pages 5–16.

[Al-Mumammed and Embley, 2007] Al-Mumammed, M. and Embley, D. (2007). Ontology-based Constraint Recognition for Free-Form Service Requests. In *Proceedings of the 23rd International Conference on Data Engineering*, pages 366–375.

[Androutsopoulos et al., 1995] Androutsopoulos, I., Ritchie, G. D., and Thanisch, P. (1995). Natural Language Interfaces to Databases - An Introduction. *Journal of Natural Language Engineering*, 1:29–81.

[Baeza-Yates and Ribeiro-Neto, 1999] Baeza-Yates, R. A. and Ribeiro-Neto, B. A. (1999). *Modern Information Retrieval*. ACM Press / Addison-Wesley.

[Borkar et al., 2001] Borkar, V., Deshmukh, K., and Sarawagi, S. (2001). Automatic Segmentation of Text into Structured Records. In *Proceedings of the 21st ACM SIGMOD International Conference on Management of Data*, pages 175–186.

[Calado et al., 2002] Calado, P., da Silva, A. S., Vieira, R. C., Laender, A. H. F., and Ribeiro-Neto, B. A. (2002). Searching Web Databases by Structuring Keyword-Based Queries. In *Proceedings of the 11th International Conference on Information and Knowledge Management*, pages 26–33.

[Califf and Mooney, 1999] Califf, M. E. and Mooney, R. J. (1999). Relational Learning of Pattern-Match Rules for Information Extraction. In *Proceedings of the 16th National Conference on Artificial Intelligence*, pages 328–334.

[Cohen et al., 2003] Cohen, W. W., Ravikumar, P. D., and Fienberg, S. E. (2003). A Comparison of String Distance Metrics for Name-Matching Tasks. In *Proceedings of IJCAI-03 Workshop on Information Integration on the Web*, pages 73–78.

[Cortez et al., 2010] Cortez, E., da Silva, A. S., Moura, E. S., and Gonçalves, M. A. (2010). ONDUX: On-Demand Unsupervised Learning for Information. In *Proceedings of the 30th ACM SIGMOD International Conference on Management of Data*, page Aceito para publicação.

[Crescenzi et al., 2001] Crescenzi, V., Mecca, G., and Merialdo, P. (2001). RoadRunner: Towards Automatic Data Extraction from Large Web Sites. In *Proceedings of 27th International Conference on Very Large Data Bases*, pages 109–118.

[Embley et al., 1998] Embley, D. W., Campbell, D. M., Smith, R. D., and Liddle, S. W. (1998). Ontology-based extraction and structuring of information from data-rich unstructured documents. In *Proceedings of the 7th International Conference on Information and Knowledge Management*, pages 52–59.

[Freitag and McCallum, 2000] Freitag, D. and McCallum, A. (2000). Information Extraction with HMM Structures Learned by Stochastic Optimization. In *Proceedings of the 15th National Conference on Artificial Intelligence*, pages 584–589.

[Hsu and Dung, 1998] Hsu, C.-N. and Dung, M.-T. (1998). Generating Finite-State Transducers for Semi-Structured Data Extraction from the Web. *Information Systems*, 23(8):521–538.

[Kristjansson et al., 2004] Kristjansson, T. T., Culotta, A., Viola, P. A., and McCallum, A. (2004). Interactive Information Extraction with Constrained Conditional Random

Fields. In *Proceedings of the 19th National Conference on Artificial Intelligence*, pages 412–418.

[Laender et al., 2002] Laender, A. H. F., Ribeiro-Neto, B. A., da Silva, A. S., and Teixeira, J. S. (2002). A Brief Survey of Web Data Extraction Tools. *SIGMOD Record*, 31(2):84–93.

[Lafferty et al., 2001] Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the 18th International Machine Learning Society*, pages 282–289.

[Li et al., 2005] Li, Y., Yang, H., and Jagadish, H. V. (2005). NaLIX: An Interactive Natural Language Interface for Querying XML. In *Proceedings of the 25th ACM SIGMOD International Conference on Management of Data*, pages 900–902.

[Mansuri and Sarawagi, 2006] Mansuri, I. R. and Sarawagi, S. (2006). Integrating Unstructured Data into Relational Databases. In *Proceedings of the 22nd International Conference on Data Engineering*, page 29.

[Mesquita et al., 2007] Mesquita, F., da Silva, A. S., de Moura, E. S., Calado, P., and Laender, A. H. F. (2007). LABRADOR: Efficiently Publishing Relational Databases on the Web by Using Keyword-Based Query Interfaces. *Information Processing and Management*, 43(4):983–1004.

[Muslea et al., 2001] Muslea, I., Minton, S., and Knoblock, C. (2001). Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, 4(1/2):93–114.

[Pearl, 1988] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, 2nd edition.

[Peng and McCallum, 2006] Peng, F. and McCallum, A. (2006). Information Extraction from Research Papers Using Conditional Random Fields. *Information Processing and Management*, 42(4):963–979.

[Rabiner, 1989] Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. 77(2):257–286.

[Reis et al., 2004] Reis, D., Golgher, P., Silva, A., and Laender, A. (2004). Automatic Web News Extraction Using Tree Edit Distance. In *Proceedings of the 13th International Conference on World Wide Web*, pages 502–511.

[Ribeiro-Neto and Muntz, 1996] Ribeiro-Neto, B. A. and Muntz, R. R. (1996). A Belief Network Model for IR. In *Proceedings of the 19th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 253–260.

[RISE, 1998] RISE (1998). Rise: A repository of online information sources used in information extraction tasks. *[http://www.isi.edu/info-agents/RISE/index.html] Information Sciences Institute / USC*.

[Rukzio et al., 2008] Rukzio, E., Noda, C., Luca, A. D., Hamard, J., and Coskun, F. (2008). Automatic Form Filling on Mobile Devices. *Pervasive and Mobile Computing*, 4(2):161–181.

[Sarawagi, 2008] Sarawagi, S. (2008). Information extraction. *Foundations and Trends in Databases*, 1(3):261–377.

[Toda et al., 2009] Toda, G. A., Cortez, E., de Sá Mesquita, F., da Silva, A. S., de Moura, E. S., and Neubert, M. S. (2009). Automatically filling form-based web interfaces with free text inputs. In *Proceedings of the 18th International Conference on World Wide Web*, pages 1163–1164.

[Vilarinho et al., 2007] Vilarinho, E. C. C., da Silva, A. S., Gonçalves, M. A., de Sá Mesquita, F., and de Moura, E. S. (2007). FLUX-CIM: Flexible Unsupervised Extraction

of Citation Metadata. In *Proceedings of the 7th Joint Conference on Digital Libraries*, pages 215–224.

[Wang and Lochovsky, 2003] Wang, J. and Lochovsky, F. H. (2003). Data Extraction and Label Assignment for Web Databases. In *Proceedings of 12th International World Wide Web Conference*, pages 187–196.

[Zhao et al., 2008] Zhao, C., Mahmud, J., and Ramakrishnan, I. V. (2008). Exploiting Structured Reference Data for Unsupervised Text Segmentation with Conditional Random Fields. In *Proceedings of the 8th SIAM International Conference on Data Mining*, pages 420–431.