Seleção de Valores para Preenchimento Automático de Formulários Web

Gustavo Zanini Kantorski[1], Tiago Guimarães Moraes[1], Carlos Alberto Heuser[1]

[1] Universidade Federal do Rio Grande do Sul
{gzkantorski, tgmoraes, heuser}@inf.ufrgs.br

Abstract. In this paper we present a proposal for the implementation and evaluation of some strategies for automatically filling Web forms. The goal is to minimize the number of queries that must be generated for automatic filling of Web forms. Our solution takes a form as input, extracts fields from it and fills appropriate fields. Experiments are presented to evaluate the proposed strategies.

Categories and Subject Descriptors: H.3.3 Information Systems [**Information Search and Retrieval**]: Databases

Keywords: Hidden Web, Filling forms, Data extraction

1. INTRODUÇÃO

Muitas informações encontradas na Web são disponibilizadas de forma dinâmica e são acessíveis por meio de formulários de consulta. Geralmente estas informações estão contidas em bases de dados tradicionais e as máquinas de busca não conseguem indexar o seu conteúdo. Esta parte da Web, cujo conteúdo está escondido atrás de formulários, é denominada Web Oculta ou Web Profunda. Bergman (2001) afirma que a informação disponível na Web Oculta é, atualmente, 400 a 500 vezes maior que a disponível na Web tradicional. A Web Oculta é caracterizada pelo seu crescimento, diversidade de domínio e grande número de bancos de dados estruturados [He et al. 2009].

Nesse contexto, um dos desafios existentes entre as diferentes arquiteturas que recuperam informações escondidas atrás de interfaces de consulta é o preenchimento automático dos campos existentes nos formulários. Logo, o desafio pretendido não é simples, na medida em que as interfaces foram projetadas para serem utilizadas por seres humanos. Além disso, a maioria das arquiteturas quer recuperar a maior quantidade de dados escondida por trás das interfaces. Assim, para alcançar a maior cobertura dos dados, a solução mais simples é a combinação de todos os possíveis valores dos campos de entrada e a respectiva submissão de cada uma das combinações. Todavia essa solução não é factível quando o número e valores de campos são grandes. Constata-se que um formulário, com cinco campos onde cada campo possua trinta valores possíveis, tem 24.300.000 possibilidades de preenchimento.

Uma questão que deve ser considerada, é que de todas as combinações, algumas podem retornar resultados idênticos ou mesmo não precisariam ser executadas. Outras combinações podem não retornar dados. Assim, um dos desafios de preenchimento automático de formulários está na tarefa de minimizar as submissões e maximizar a cobertura da base de dados. Pode-se resumir esse desafio por meio de duas questões principais: i. Quais os elementos do formulário devem ser preenchidos e ii. Quais os valores que devem ser informados nesses elementos.

Desse modo, o presente artigo propõe várias estratégias para seleção de valores para o preenchimento de formulários que permitem maximizar a cobertura dos dados escondidos por trás de formulários. Uma segunda contribuição é a possibilidade de geração de valores para o preenchimento de campos que não possuem valores presentes no corpo do formulário tais como, campos de texto.

O artigo está organizado da seguinte forma. A seção 2 discute os trabalhos relacionados. Seção 3 apresenta o problema de preenchimento de formulários, a arquitetura de referência e as extensões propostas neste trabalho. A seção 4 descreve as estratégias propostas para seleção de valores de preenchimento de formulários. Na seção 5 são apresentados resultados e análises dos experimentos. Considerações finais e trabalhos futuros são discutidos na seção 6.

2. TRABALHOS RELACIONADOS

Diferentes trabalhos apresentam soluções para o preenchimento de formulários na Web Oculta [Raghavan e Garcia-Molina 2001], [Liddle et al. 2003], [Wu et al. 2006], [Alvarez et al. 2007], [Madhavan et al. 2008], [Barbosa e Freire 2004], [Toda et al. 2010], [Khare et al. 2010]. No entanto, a maioria não apresenta de maneira clara como os valores são selecionados para preenchimento dos formulários.

Raghavan e Garcia-Molina (2001) tratam a questão do preenchimento de formulários por meio de tabelas, denominadas de *Label Value Set* (LVS), que são associadas aos campos dos formulários. Cada linha na tabela LVS é da forma *(L,V)*, onde *L* é um rótulo e $V=\{v_1,v_2,\ldots,v_n\}$ é um conjunto de valores pertencentes ao rótulo. O conjunto *V* tem uma função M_V que associa pesos, no intervalo entre [0,1], para cada membro do conjunto. Cada v_i representa um valor possível que pode ser associado a um campo *E* do formulário se *label(E)* combinar com *L*. $M_V(v_i)$ representa uma estimativa de quão correto o valor v_i é em relação a *E*. A principal questão, neste método, é a necessidade de criar e popular a tabela LVS com os valores desejados para as consultas e associar os valores aos campos dos formulários.

Para Liddle et al. (2003) o preenchimento automático de formulários é realizado pela atribuição do valor *default* aos campos do formulário. As informações recuperadas são analisadas e novas consultas são submetidas, exaustivamente, até que um determinado limiar seja alcançado. A principal questão deste método é que não existe tratamento para campos texto que possuem domínio infinito. Esses campos são ignorados no preenchimento dos formulários. Caso os campos sejam obrigatórios é necessária a intervenção do usuário.

Barbosa e Freire (2004) apresentam uma abordagem para tratamento de formulários baseados em palavras chaves. O objetivo é descobrir palavras que consigam atingir uma maior cobertura dos dados. A descoberta das palavras é baseada nos dados existentes dentro da própria coleção, ao invés de uma geração aleatória de palavras. Desse modo, é selecionada uma amostra de palavras baseada na freqüência dos resultados. A partir dessa amostra são construídas consultas para avaliação da cobertura dessas palavras. A principal vantagem deste método é que formulários baseados em palavras chaves não requerem nenhum conhecimento detalhado de estrutura ou esquema dos dados. Um contraponto que o trabalho de [Barbosa e Freire 2004] não aborda é para formulários que possuem campos que não são palavras chaves.

Wu et al. (2006) apresenta uma forma de preenchimento baseado em um processo de retroalimentação dos valores preenchidos no formulário. O formulário é visualizado como uma base de dados DB como uma única tabela com *n* registros $\{t_1,t_2,\ldots,t_n\}$ sobre um conjunto de *m* atributos $AS=\{attr_1,attr_2,\ldots,attr_m\}$. O conjunto de valores de atributos distintos (DAV) consiste de todos os distintos valores de atributos que aparecem em DB. Um grafo *G(V,E)* atributo-valor (AVG) para DB é um grafo não direcional que representa a relação entre os valores se eles coexistem em uma instância relacional $t_k \in DB$. A determinação dos valores que são utilizados para o preenchimento dos formulários é alcançada pela utilização de um valor inicial e a partir dos resultados são extraídos valores relacionados com o valor utilizado inicialmente na consulta. A questão principal do trabalho proposto por [Wu 2006] é que a consulta deve ter somente um predicado de igualdade. Não é possível combinar os campos do formulário para várias consultas.

O trabalho proposto por Alvarez et al. (2007) apresenta uma arquitetura denominada DeepBot para *crawler* na Web Oculta. Alvarez et al. (2007) utiliza um conjunto de atributos de domínio, que

representam campos de um formulário e um conjunto de consultas associada ao domínio. O conjunto de atributos possui um nome e um conjunto de apelidos. O conjunto de consultas é uma lista de pares (atributo, valor). Para realizar o preenchimento dos formulários são extraídos os rótulos associados aos campos. Os rótulos são comparados aos atributos do domínio por intermédio de técnicas de similaridade textual. Determinados os atributos que representam os campos, os valores do conjunto de consultas são utilizados para o preenchimento dos campos do formulário. Alvarez et al. (2007) não apresenta nenhum detalhe de como a lista de consultas é construída para cada atributo.

Toda et al. (2010) apresenta uma solução de preenchimento de formulários baseada na extração de valores a partir de um documento de texto livre. A solução proposta por [Toda et al. 2010] é dividida em dois subproblemas: a) extrair os valores do documento de texto e b) preencher os campos do formulário usando os valores extraídos. A solução explora características relacionadas ao conteúdo e o estilo dos valores extraídos do documento texto, que são combinadas em uma rede bayesiana. Desta forma são computadas as probabilidades condicionais de um campo estar associado a um valor extraído do documento texto. A abordagem apresentada por [Toda et al. 2010] é dependente do conhecimento obtido de submissões de valores anteriores para cada campo.

O trabalho proposto por [Madhavan et al. 2008] descreve um sistema para *"surfacing"* no conteúdo da Web Oculta. O objetivo é indexar as páginas HTML resultantes de consultas realizadas em formulários. Formulários HTML normalmente possuem mais que uma entrada e uma estratégia ingênua de preenchimento, como o produto cartesiano de todas as possibilidades, pode não ser factível devido ao grande número de consultas que podem ser geradas. Desta forma, [Madhavan et al. 2008] utiliza o conceito de *templates* para os formulários. Um *template* é composto por um subconjunto de campos do formulário, chamados de campos vinculados. Os demais campos são chamados de campos livres. Múltiplas submissões para o formulário podem ser geradas pela associação de diferentes valores para os campos vinculados. Se existem n campos em um formulário, existem $(2^n - 1)$ *templates*. Por exemplo, um formulário que possui três campos, c_1, c_2 e c_3, possui sete *templates*: c_1, c_2, c_3, c_1c_2, c_1c_3, c_2c_3, $c_1c_2c_3$. O número de campos vinculados em um *template* define a dimensão do *template*.

Quando valores são associados aos campos vinculados de um *template* tem-se uma instância de *template*. [Madhavan et al. 2008] determina a quantidade de valores que são associados a um *template* mas não detalha como os valores são gerados. Assim, para cada *template* são geradas várias instâncias de *template*. Cada instância de *template* é submetida e o resultado é avaliado para verificar a quantidade de informações recuperadas. Todos os resultados gerados pelas instâncias de *template* são comparados por uma função que define a assinatura da instância do *template*. Um *template* é considerado informativo se os resultados gerados pelas instâncias de *template* forem suficientemente distintos. Caso contrário o *template* é considerado não informativo e é descartado. Madhavan 2008 utiliza um algoritmo, denominado ISIT (*Incremental Search for Informative Templates)*, para verificar se um *template* é informativo ou não. A vantagem é que todos os *templates* com dimensão maior que possuem o mesmo campo vinculado que resultou no *template* não informativo também são descartados.

A figura 1 ilustra a arquitetura proposta por [Madhavan et al. 2008] e as contribuições propostas em nosso trabalho para selecionar os valores para preenchimento automático de formulários. O módulo de *templates* candidatos representa todos os *templates* que podem ser gerados pelo formulário HTML. O módulo de preenchimento de formulário e geração de instâncias de *templates* adiciona os valores de entrada para cada campo vinculado dos *templates*. Os *templates* são gerados em ordem de dimensão. Primeiro todos os *templates* de dimensão 1, depois os de dimensão 2 e assim sucessivamente. Cada instância de *template* é submetida e uma assinatura é gerada. Após a submissão de todas as instâncias de um *template*, o algoritmo ISIT avalia se o *template* é informativo. Os resultados de cada submissão de instâncias de *templates* são armazenados para a geração de valores futuros que serão utilizados nos próximos *templates*. Maiores detalhes sobre as contribuições à arquitetura de referência são apresentadas na seção 3.

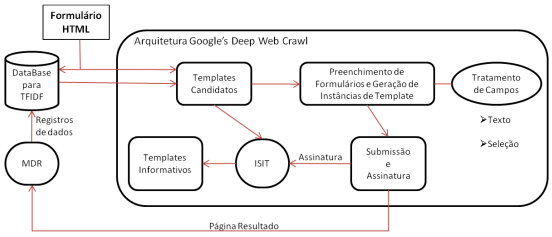

Figura 1. Arquitetura baseada no trabalho de [Madhavan 2008]

3. PREENCHIMENTO AUTOMÁTICO DE FORMULÁRIOS

O problema de preenchimento automático de formulários é definido em termos da escolha do grupo de consultas que devem ser submetidas ao formulário. Esta seção apresenta as características de formulários HTML e as extensões propostas à arquitetura de referência de [Madhavan et al. 2008] com o objetivo de selecionar o grupo de consultas para submissão.

Normalmente os dados presentes na Web Oculta estão escondidos por trás de formulários. Os formulários mais utilizados são os formulários HTML. Um exemplo de formulário HTML para o domínio de empregos é mostrado na figura 2. Um formulário HTML é delimitado pelas tags <*form*> e </*form*> (Figura 2b). Os formulários possuem campos para inserção de dados, que são as caixas de texto (*text boxes*), listas de seleção (*selection lists*), *check boxes, radio buttons* e *submit buttons.* Uma caixa de texto é representada como uma caixa vazia com ou sem um valor *default.* Na Figura 2a o campo *Keywords* é um exemplo de caixa de texto. Um campo do tipo *selection list* mostra para o usuário uma lista de opções. Existem dois tipos de listas de seleção, listas de seleção única (*combo box*) e listas de seleção múltipla (*list box*). Os campos *Location* e *Category* são exemplos de listas de seleção, na Figura 2a. *Radio buttons* e *check boxes* são variações de listas de seleção onde os projetistas mostram as opções para o usuário escolher. Desta forma, tanto *check boxes* quanto *radio buttons* podem ser tratados como *selection lists*. Resumindo, existem dois tipos de campos: campos com domínio finito tais como listas de seleção e campos com domínio infinito, tais como caixas de texto.

A geração dos valores para campos de domínio finito é relativamente fácil, uma vez que, os valores a serem selecionados estão fixos dentro do próprio código do formulário. A seleção de valores para campos de domínio infinito não é uma tarefa trivial. Isto ocorre porque não é possível saber de antemão a quantidade de valores, assim como, a qualidade dos valores escolhidos. A quantidade refere-se ao número total de valores selecionados para atingir a cobertura desejada e a qualidade diz respeito a escolha de valores que retornam mais dados distintos.

As consultas submetidas para um formulário podem ser feitas por meio de dois métodos, *get* ou *post.* No método *get* os parâmetros são adicionados na ação do formulário e incluídos como parte da URL na requisição HTTP. No método *post* os parâmetros são enviados dentro da requisição HTTP e a URL contém somente a ação a ser executada. Os experimentos realizados neste trabalho consideram tanto formulários com métodos *get* quanto os com método *post.*

Para que seja possível a geração dos valores a serem preenchidos no formulário foi implementada uma arquitetura semelhante à proposta de [Madhavan et al. 2008], mostrada na figura 1. A contribuição na arquitetura original foi a implementação do algoritmo MDR [Liu e Zhao 2003] (Figura 1), com a finalidade de encontrar dentro da página resultante de uma submissão qual a região que contém somente dados. Informações referentes a propagandas e leiaute da página de resultado são desconsideradas. Uma segunda contribuição foi a criação de uma base de dados onde os resultados das submissões são armazenados. Essa base de dados serve para geração dos valores para campos com domínios infinitos. Esses valores serão utilizados para a criação do grupo de consultas que serão submetidas ao formulário.

A intuição é que a probabilidade de cobertura seja maior para valores baseados nos dados recuperados de submissões anteriores comparados com a geração de valores aleatórios.

Figura 2. (a) Exemplo de Formulário e (b) Código HTML do formulário

A seleção dos valores para campos de domínio infinito é realizada por uma técnica baseada em TF-IDF [Salton e McGill 1983]. A base de dados para TF-IDF, mostrada na figura 1, é utilizada para geração dos valores. Os resultados de cada submissão são armazenados nessa base de dados que é processada e de onde são extraídos *t tokens* ranqueados conforme a frequência e a ocorrência de cada *token* no banco de dados. A cada novo *template*, que possua campos com domínio infinito, a base de dados é novamente processada e são extraídos os *t tokens* mais relevantes. Assim, para *template*s diferentes que possuam o mesmo campo de domínio infinito, valores distintos serão gerados.

Os *template*s são representados pelos campos do formulário e suas respectivas combinações. Assim para o formulário da Figura 2a temos os seguintes *template*s: *q, lid, fn, q&lid, q&fn, lid&fn* e *q&lid&fn*. As instâncias de *template* são implementadas através das URLs para submissão dos formulários conforme seja o método *get* ou *post*. A seleção dos valores de cada *template* depende do tipo de campo, finito ou infinito, e da dimensão do *template*. Para *template*s com dimensão um e campos com domínio finito serão geradas tantas submissões conforme o número de opções existentes para cada campo. Para campos de domínio infinito não existe um número pré-determinado de valores. Para *template*s com dimensão maior que um, onde existirá a combinação de valores, serão utilizadas quatro estratégias de seleção. A seção 4 descreve em detalhes cada uma delas. Todas as instâncias de *template* são geradas e apenas algumas serão submetidas conforme a estratégia utilizada para seleção.

4. ESTRATÉGIAS DE SELEÇÃO DE VALORES

Esta seção descreve as quatro estratégias propostas para selecionar os valores que serão utilizados para preencher os campos dos formulários. Esses valores formarão, juntamente com os campos, o grupo de consultas que serão submetidas aos formulários.

Considere um formulário com dois campos A e B. O campo A tem quatro valores possíveis *{a1, a2, a3, a4}* e o campo B possui seis valores *{b1, b2, b3, b4, b5 ,b6}*. Os *template*s para este formulário são A, B e AB. O número de submissões que podem ser geradas para este formulário é a combinação de submissões formadas por cada campo individualmente mais o produto cartesiano entre os campos. Portanto, temos quatro submissões para o campo A, seis submissões para o campo B e 24 submissões para a combinação de campos A e B, totalizando 34 possíveis submissões. As instâncias de *template*s são: *A=a1, A=a2,..., A=a4, B=b1, B=b2,…,B=b6, A=a1&B=b1,…., A=a4&B=b6.*

A maneira mais ingênua de pensar o problema é considerar a submissão de todas as possibilidades de valores para os campos do formulário. Para formulários com poucos campos e com poucos valores para cada campo essa estratégia pode ser viável, mas para formulários onde os campos possuem muitos

valores essa estratégia é proibitiva. Considerando um formulário com vários campos de entrada, milhões de possibilidades de preenchimento podem existir. Considere que por trás do suposto formulário a base de dados contenha um pequeno número de registros. Provavelmente um pequeno número de submissões seja necessário e suficiente para recuperar todas as tuplas da base de dados. Portanto, são avaliadas quatro estratégias que selecionam *k* instâncias de *template* para submissão.

Uma estratégia para seleção de valores é denominada *k-first.* O algoritmo de geração das instâncias inicia com o primeiro campo e para cada valor deste campo, combina com todas as possibilidades do segundo campo, e assim sucessivamente, até que todos os campos tenham sido combinados. Após a geração de todas as instâncias do *template*, o algoritmo seleciona as *k* primeiras geradas para submissão. Supondo que o valor de *k* seja igual a 6, as quatro instâncias para o *template* A mais as seis instâncias para o *template* B são selecionadas. Para o *template* A&B, que possui 24 instâncias, somente seis serão selecionadas: *A=a1&B=b1, A=a1&B=b2, A=a1&B=b3, A=a1&B=b4, A=a1&B=b5, A=a1&B=b6*, conforme a figura 3a.

Outra estratégia avaliada, denominada de *k-random*, seleciona um subconjunto *k* de instâncias de *template* de maneira estocástica. As instâncias geradas são equiprováveis dentro do espaço de busca de todas as instâncias. Nenhuma análise é considerada para a escolha das instâncias. Um exemplo é mostrado na figura 3b.

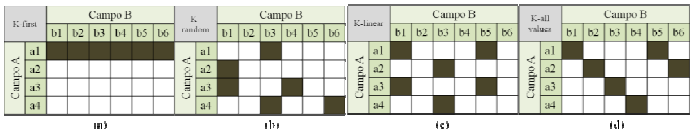

Figura 3. Estratégias: (a) *k-first* (b) *k-random* (c) *k-linear* (d) k-allValues

A estratégia seguinte, chamada *k-linear*, seleciona o subconjunto de k de instâncias de acordo com um passo linear *p*, determinado a partir do valor de *k*. Considere o número total de instâncias *n*, o valor do passo *p* é determinado pela razão *n/k*. Considerando o valor de *k* igual a seis e o valor de *n* para o *template* A&B igual a 24, o passo linear é calculado pela razão 24/6 = 4. Assim, as instâncias selecionadas para submissão são: *A=a1&B=b1, A=a1&B=b5, A=a3&B=b3, A=a3&B=b1, A=a3&B=b5, A=a4&B=b3*, de acordo com a figura 3c.

As estratégias anteriores apresentam várias restrições. Uma restrição existente em todas as estratégias apresentadas é que nenhuma delas garante, pelo menos uma vez, a seleção de todos os valores dos campos do formulário. Ainda, o mesmo valor pode ser selecionado várias vezes, o que pode aumentar o número de resultados duplicados assim como um determinado valor pode nunca ser selecionado.

Para evitar as restrições das estratégias anteriores, é proposta uma estratégia chamada *k-allValues* que tem por objetivo utilizar todos os valores dos campos pelo menos uma vez. Para garantir que todos os valores de todos os campos sejam usados pelo menos uma vez, o valor de *k* deve ser maior ou igual ao conjunto de valores possíveis para o campo que tenha o maior número de opções. A figura 3d mostra um exemplo da estratégia *k-allValues* para o valor de *k* igual a 6. A idéia é que a utilização de todos os valores, pelo menos uma vez, pode atingir uma cobertura maior dos dados.

5. EXPERIMENTOS

Esta seção descreve os experimentos executados para avaliar as estratégias de seleção de valores para preenchimento de formulários. Todos os experimentos foram realizados em formulários reais encontrados na Web. Para realização dos experimentos foram utilizados formulários de vários tamanhos e domínios, pois as estratégias de preenchimento propostas assim como a arquitetura implementada são independentes de domínio. O objetivo é avaliar as estratégias de maneira geral sem considerar a influência do domínio. A tabela I apresenta os detalhes das bases de dados usadas nos

experimentos. Para cada formulário é mostrado o número de URLs geradas para todas as estratégias conforme o valor *k* definido para as instâncias de *template*, o número de URLs do *baseline*, o número de registros encontrados por trás do formulário e os tipos de campos de cada formulário. Para análise dos experimentos, o *baseline* utilizado foi o resultado alcançado pelo produto cartesiano de todos os campos do formulário.

Tabela I. Características de cada coleção usada nos experimentos

Id	Formulário	URLs Submetidas	URLs Baseline	Tamanho do DB	# campos text	# campos select
1	http://www.global-standard.org/public-database/search.html	1609	2209	1571	1	3
2	http://www.hcareers.com/seeker/search	9197	1913840	6176	1	5
3	http://www.rtbookreviews.com/rt-search/books	3591	41299	42108	1	3
4	http://www.boston.com/jobs/	876	1375	60676	1	2
5	http://www.shadetrees.org/search.php	2041	77549	1756	0	4
6	http://pipa.gov.ps/results.asp	5474	3343199	409	0	4
7	http://www.usajobs.gov/	972	2600	5000*	2	0
8	http://formovies.com/search/combined.html	150	150	17357*	3	0
9	http://www.mymusic.com/advancedsearch.asp?curr=1	3357	8057	100000*	6	2

* Tamanho estimado

Para avaliar os resultados dos experimentos foram utilizadas duas métricas, a cobertura C_e e o número médio de linhas recuperadas por instância de *template* $numIT_e$ de cada estratégia. Seja N_f o número total de registros existentes na base de dados por trás do formulário *f* e nr_e o número total de registros distintos recuperados pela estratégia *e,* a cobertura C_e é dada por nr_e / N_f. Considere ns_e o número total de submissões para uma estratégia *e*, o numero médio de instâncias de *template* de uma estratégia $numIT_e$ é dado pela razão nr_e/ns_e.

O número de instâncias de *template k,* representadas pelas URLs geradas, foi estimado como uma função do número de entradas do formulário. Para formulários com dois ou três campos o valor de *k* foi definido para 800. Para formulários com quatro campos *k* foi definido como 600. Formulários com mais de cinco campos o valor de *k* foi ajustado para 400. Foram escolhidos formulários com até seis campos de entrada porque eles representam 99% de todos os formulários existentes na Web Oculta [Madhavan et al. 2008]. Para o número de *tokens t,* associados aos campos de domínio infinito, os experimentos realizados mostraram que o valor de 50 produziram resultados interessantes.

A tabela II apresenta os resultados alcançados. Para cada formulário são apresentadas as métricas e os respectivos valores para cada estratégia e para o *baseline.* Os valores de *numRec* são o número de registros distintos recuperados de cada base de dados. Os valores de *numIT* são o número médio de linhas recuperadas para cada submissão. Quanto maior este valor melhor a estratégia. A cobertura *C* varia de 0 a 100 e representa uma escala percentual do total de registros recuperados da base de dados.

Tabela II. Resultados para os formulários

Id	*k-first*			*k-allValues*			*k-linear*			*k-random*			*Baseline*		
	numRec	*numIt*	C_e	*numRec*	*numIt*	C_e	*numRec*	*numIt*	C_e	*numRec*	*numIt*	C_e	*numRec*	*numIt*	C_e
1	1061	0,65	65,95	1077	0,66	66,94	1076	0,66	66,88	1130	0,70	70,23	1139	0,51	70,78
2	3709	0,40	60,06	4021	0,44	65,12	3665	0,39	59,34	4136	0,44	66,97	6176	0,003	100
3	10751	2,99	25,54	11130	3,10	26,44	10309	2,87	24,49	12144	3,38	28,85	32765	0,79	77,82
4	29379	33,51	48,82	28714	32,78	47,33	35603	40,64	58,68	32592	37,23	53,71	56481	41,08	93,08
5	1756	0,86	100	1756	0,86	100	1756	0,86	100	1756	0,86	100	1756	0,02	100
6	408	0,07	99,75	408	0,07	99,75	389	0,06	95,22	404	0,07	98,77	409	10^{-5}	100
7	2295	2,36	45,90	1158	1,20	23,16	1592	1,63	31,84	2639	2,71	52,78	2713	1,04	54,26
8	1198	7,98	0,06	993	6,75	0,05	1134	7,56	0,06	1163	7,75	0,06	886	5,91	0,05
9	3847	1,14	0,03	5567	1,65	0,05	5230	1,55	0,05	5135	1,52	0,05	11294	1,40	0,11

A avaliação conjunta das tabelas I e II fornece alguns *insights* interessantes. Para bases de dados pequenas todas as estratégias obtiveram um comportamento melhor que o *baseline*. A cobertura (*C*)

foi acima de 95% em todas as estratégias e o número médio de registros por instâncias de *template* (*numIT)* das quatro estratégias foi muito superior ao do *baseline*. Estas características se repetiram para aqueles formulários que possuem somente campos com domínio finito. Nos formulários que possuem os dois tipos de campos a cobertura alcançada pelo *baseline* foi superior, embora algumas estratégias obtiveram resultados similares ao *baseline*.

6. CONCLUSÕES

Neste artigo foi examinado o problema de seleção de valores para preenchimento automático de formulários na Web. Foram propostas quatro estratégias automáticas para seleção de valores. As estratégias foram eficientes, alcançando uma alta cobertura dos dados. Além disso, as mudanças propostas na arquitetura permitiram a manipulação de campos de domínio infinito por meio de um processo de retroalimentação das submissões anteriores.

Os resultados alcançados com os experimentos preliminares demonstraram que as estratégias propostas possuem potencial. Novos experimentos devem ser realizados por meio da utilização de mais formulários. Os valores selecionados para todos os campos de domínio infinito que pertencem a um *template* são únicos. Os experimentos comprovaram que esta abordagem de seleção funciona bem para campos de domínio infinito baseado em palavras chaves. Considerações sobre a semântica dos campos podem melhorar ainda mais os resultados alcançados.

Embora as estratégias abordadas sejam independentes de domínio, trabalhos futuros devem investigar o comportamento dessas estratégias dentro de domínios específicos, bem como verificar se as características de cada domínio podem afetar a seleção de valores de preenchimento. Experimentos posteriores podem ser realizados para analisar os valores selecionados visto que cada domínio possui propriedades intrínsecas que podem influenciar na seleção dos valores.

AGRADECIMENTOS

Este trabalho foi parcialmente apoiado pelo CNPq (processo 480283/2010-9). Os autores expressam seus sinceros agradecimentos a CAPES pela concessão de bolsa para Gustavo Z. Kantorski e ao CNPq pela bolsa de estudos a Tiago G. Moraes.

REFERÊNCIAS

ALVAREZ,M., RAPOSO, J., PAN, A., CACHEDA, F. BELLAS,F. CARNEIRO, V. Crawling the Content Hidden Behind Web Forms, Computational Science and Its Applications. ICCSA. 2007. P.322-333.

BARBOSA, L. AND FREIRE, J. Siphoning Hidden Web Data through Keyword-Based Interfaces. SBBD 2004: 309-321.

BERGMAN, M.K. 2001. The Deep web: Surfacing Hidden Value. White Paper. University of Michigan.

CHANG,K.C., HE,B., AND ZHANG,Z. 2005. Toward large scale integration: Building a MetaQuerier over Databases on the Web. In Proc. Of 2nd Conference on Innovative Data Systems Research (Asilomar, CA, Jan., 4-7, 2005) CIDR'05. ACM Press, New York, NY. 44-55.

HE, B., PATEL,M. ZHANG,Z. AND CHANG, K.C. 2007. Accessing the DeepWeb. Communications of the ACM, 50, 5 (Oct. 2008), 94-101.

KHARE,R., AN, Y., AND SONG, II-Y. Understanding Deep Web Search Interfaces: A Survey. 2010. ACM SIGMOD Record, Volume 39. March 2010. New York, NY. 33-40.

LIDDLE, S.W., EMBLEY, D.W., SCOTT, D.T., YAU, S.H. 2003. Extracting Data Behind Web Forms. In: Proceeding of Advanced Conceptual Modeling Techniques. Springer Berlin.

LIU, B., GROSSMAN, R., ZHAO, Y. Mining data records in web pages. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-2003), pages 601–606, Washington, D.C, 2003.

MADHAVAN, J.,KO, D., KOT, L., GANAPATHY, V., RASMUSSEN, A., HALEVY, A. Google's Deep Web crawl. Proceedings of the VLDB Endowment. Volume 1 Issue 2, August 2008. ISSN: 2150-8097.

MADHAVAN, J. JEFRREY, S.R., COHEN, S.I., DONG, X., KO, D., YU, C. 2007. Web-Scale data integration: you can only afford to pay as you go. In Proceeding of Conference on Innovative Data Systems Research. CIDR'07. ACM Press, New York, NY-40-48.

RAGHAVAN,S. GARCIA-MOLINA,H. 2001. Crawling the Hidden Web. In Proc. of the 27th International Conference on Very Large Databases (Rome, Italy, September 11-14, 2001) VLDB'01, Morgan Kaufmann Publishers Inc, Sao Francisco, CA, 129-138.

SALTON,G. MCGILL, M.J. Introduction to Modern Information Retrieval. 1983.

TODA, G., CORTEZ,E., SILVA, A.S., MOURA, E.S. A Probabilistic Approach for Automatically Filling Form-Based Web Interfaces. PVLDB 4(3): 151-160 (2010).

WU, P., WEN, J.R., LIU, H., MA, W. Query Selection Techniques for Efficient Crawling of Structured Web Sources. 2006. In Proceedings if the 22nd International Conference on Data Engineering.

ZHAI, Y. AND LIU, B. Mining Data Records in Web Pages. 2005. In proceedings of the 14th International Conference on World Wide Web Chiba, ACM.