Estratégias de Persistência de *Clusters* em uma Técnica de Casamento por Similaridade para *Web Forms*[1]

Filipe Roberto Silva, Ronaldo dos Santos Mello

Departamento de Informática e Estatística (INE) – Centro Tecnológico (CTC)
Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC – Brasil

{filipesilva.sc,ronaldo}@inf.ufsc.br

Abstract. *Most data available currently in Web are just accessible by queries in Web forms. These hidden data bases that are just revealed in response to those queries are called Deep Web. One way to search contents in the Deep Web is to provide to the user relevant Web forms to his/her interests (car dealers, airfare, etc) and the own user can formulate his/her queries. This paper presents and evaluates strategies for clusters' persistence in a Web forms' similarity search system called WF-Sim with the purpose of identifying similar Web forms efficiently and effectively.*

Resumo. *Grande parte dos dados disponíveis atualmente na Web só é acessível através de consultas formuladas sobre Web forms. Estes bancos de dados "escondidos" e revelados apenas como resposta a estas consultas são denominados de Deep Web. Uma das formas de busca a conteúdos na Deep Web é fornecer ao usuário Web forms relevantes para os seus interesses (revendas de automóveis, passagens aéreas, etc) e o próprio usuário realiza suas buscas. Este trabalho apresenta e avalia estratégias de persistência de clusters de afinidade em um mecanismo de casamento (matching) de Web forms por similaridade chamado WF-Sim, de forma que seja possível identificar Web forms semelhantes entre si de forma eficiente e eficaz.*

1. Introdução

A Internet está presente na vida de muitas pessoas. Qualquer dúvida, pesquisa, notícia e etc, pode ser alcançada através da Internet utilizando as populares máquinas de busca, como *Google* ou *Bing*. Porém, o que poucos sabem é que essas buscas atingem apenas a superfície da Web. Grande parte do conteúdo na *Web* encontra-se escondido em bancos de dados que não são acessados por tais máquinas de busca. A esse conteúdo é dado o nome de *Deep Web* ou Web oculta (*Hidden Web*)[2]. O conteúdo da *Deep Web* é revelado em páginas dinâmicas, geradas automaticamente conforme requisições do usuário realizadas sobre formulários na *Web* (*Web forms*). Assim sendo, tais páginas não podem ser indexadas pelas máquinas de busca.

[1] Este trabalho é parcialmente financiado pelo CNPq através do projeto WF-Sim (Nro. processo: 481569/2010-3).

[2] http://brightplanet.com/images/uploads/12550176481-deepWebwhitepaper.pdf

Dada a dificuldade para a descoberta e indexação de todo o conteúdo da *Deep Web*, uma abordagem possível de busca e que é o foco deste trabalho, é pesquisar somente por campos de *Web form*s, porém indexando os campos destes *Web form*s e permitindo a busca por outros *Web form*s similares, ou seja, considerando um processo de *matching* de *Web forms*. Para entender melhor esse processo, é interessante entender melhor como é formado um *Web form*.

Um *Web form* possui diversos campos (campos de texto, *check boxes*, *combo boxes* e etc), que permitem ao usuário preencher digitando ou selecionando um valor. Como é possível observar na Figura 1, os *Web form*s possuem rótulos, que são utilizados para que o usuário saiba o que colocar em cada campo. Esses campos podem conter os já citados valores para seleção, como é possível ver na Figura 1 para o campo '*Select what you have*:', que possui duas possibilidades de preenchimento: '*I have a bike e 'I have a car*'.

Assim, esses valores podem ser utilizados para a comparação de um *Web form* com outro, tanto os valores de rótulo quanto os valores de preenchimento.

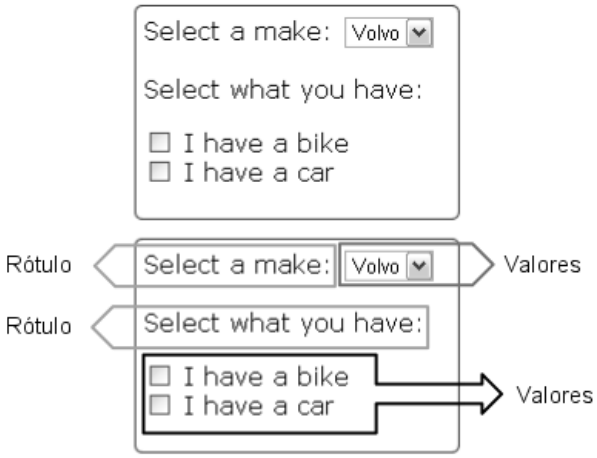

Figura 1. *Web form* e suas características

Este trabalho é uma continuação dos resultados obtidos com o projeto *WF-Sim*, projeto este que está em andamento e visa tornar possível buscas por similaridade sobre *Web form*s. Com o projeto já é possível agrupar por semelhança *Web form*s previamente extraídos da Web, indexar e a partir de um *Web form* de entrada, encontrar outros semelhantes. Experimentos foram realizados sobre o *WF-Sim* e o mesmo já tem resultados satisfatórios de busca em termos de acurácia (Gonçalves et. al. 2011). Entretanto, existem algumas melhorias a serem feitas e este trabalho visa implementar algumas dessas melhorias.

Especificamente, o objetivo desse trabalho é apresentar otimizações no algoritmo de clusterização de dados de *Web form*s utilizado pelo *WF-Sim* em termos de persistência desses *clusters*. O projeto, como foi implementado, realiza agrupamentos dos *Web form*s a cada vez que é executado. Assim, ele somente guarda os *clusters* em memória, não persistindo os mesmos. A contribuição deste trabalho é a implementação e teste de duas soluções para a persistência desses dados: uma em bancos de dados e outra em arquivos. Cada alternativa foi testada visando verificar a melhor solução para o problema. Resultados de experimentos preliminares são relatados, demonstrando qual das duas estratégias se mostrou mais promissora.

Os demais capítulos detalham o desenvolvimento deste trabalho. O capítulo 2 aborda o projeto *WF-Sim*, com foco na atividade de clusterização. O capítulo 3 mostra como foram

desenvolvidas as persistências dos *clusters* e o capítulo 4 descreve os experimentos preliminares. Por fim, no capítulo 5 são apresentadas as conclusões e atividades futuras.

2. WF-Sim

A descoberta de similaridades entre *Web form*s a partir de *strings* que representam valores em um campo é um grande desafio, já que muitas vezes duas palavras totalmente diferentes significam a mesma coisa, ou ainda duas palavras iguais em contextos diferentes possuem significados diferentes. Um exemplo disso seriam 2 *Web form*s com campos '*Manufacturer*' e '*Year of Manufacture*'. Apesar dos nomes semelhantes, eles possuem significados totalmente diferentes.

O trabalho em questão é uma extensão do projeto *WF-Sim* (*Web form Similarity*), um projeto em desenvolvimento pelo Grupo de Banco de Dados da UFSC[3] que visa realizar buscas por *Web form*s similares, tentando resolver parte desta problemática. A seguir são abordados alguns detalhes sobre o funcionamento do *WF-Sim*.

2.1 Funcionamento

O *WF-Sim* trata *Web form*s dividindo-os em elementos. Cada elemento representa um campo contendo um rótulo e opcionalmente um conjunto de valores. Essa divisão é exemplificada pela Figura 2 onde, no primeiro elemento, temos o rótulo "*Select a make*" e uma relação de valores iniciada por "Volvo".

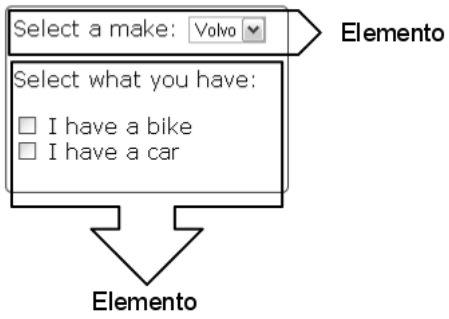

Figura 2. Divisão de um *Web form* em elementos

O projeto visa desenvolver um sistema de busca por similaridade para *Web form*s que é composto por 3 módulos principais. Estes módulos são: *clusterização*, *indexação* e *busca*. Neste sistema, os elementos são clusterizados de acordo com uma métrica de similaridade (ver Seção 2.2) e então indexados. A partir dessa indexação é possível realizar buscas, ou seja, a partir de um *Web form* de entrada são encontrados outros semelhantes. Essa busca, ao invés de comparar elemento por elemento de todos os *Web form*, compara somente os centróides, que são os elementos com maior afinidade com todos os demais elementos no *cluster* e que representam o *cluster* como um todo. Assim, a partir das comparações com os centróides, encontra-se os grupos de elementos semelhantes àquele *Web form* de entrada.

[3] http://www.gbd.inf.ufsc.br

2.2 Cálculo da Similaridade

O *WF-Sim* utiliza, para o cálculo de similaridade dos rótulos, a métrica *TF-IDF* (Cohen et. al. 2003) e para a similaridade dos valores, a métrica *SubSetSim* (Dorneles 2004), ambas escolhidas por possuírem os melhores resultados se comparados a outras métricas. A partir do cálculo das similaridades dos rótulos e dos valores separadamente, é feita uma média ponderada desses valores, como é possível ver na Equação 1, onde l_1 e l_2 são os rótulos, vl_1e vl_2 os valores, e *labelSim* e *valuesSim* são os cálculos de similaridade dos rótulos e valores, respectivamente. Por ser uma média ponderada, é possível aplicar pesos diferentes para cada atributo do elemento (*labelWeight* e *valuesWeight*), sendo a soma dos pesos igual a um (1). Vale lembrar que para elementos pertencentes a um mesmo *Web form* não é calculada a similaridade, visto que o objetivo é encontrar similaridades entre *Web forms* diferentes.

$$ElementSim = labelSim(l_1, l_2) * labelWeight + valuesSim(vl_1, vl_2) * valuesWeight \quad (1)$$

2.3 Clusterização

A *clusterização* dos elementos é feita utilizando uma extensão do algoritmo *Median Shift* (Jouili, Tabbone & Lacroix 2010), onde os elementos medianos são definidos como centróides dos *clusters*. Primeiramente, um raio de clusterização é determinado pelo usuário. Assim, é verificado em cada elemento, quais outros elementos estão contidos dentro desse raio. Esse processo é chamado de cálculo de vizinhança.

Pelo fato dos elementos não estarem contidos em um espaço vetorial, é necessário calcular todas as similaridades entre os elementos e transformar estes valores em distâncias. Como os valores de similaridade estão no intervalo de '0' a '1', sendo que o valor '0' significa ser diferente e o valor '1' ser igual, a distância é calculada utilizando a Equação 2, sendo que para valores de similaridade igual a '1' é aplicada diretamente a distância zero.

$$Distância = \frac{1}{Similaridade} \quad (2)$$

Com isso, são calculadas as distâncias entre os elementos, estes são agrupados e os centróides definidos.

3. Persistência dos Clusters

Pelo fato do *WF-Sim* ainda estar em desenvolvimento, suas clusterizações não são persistidas, ou seja, a cada execução, todos os formulários são novamente processados e clusterizados. Isso foi feito devido à necessidade inicial de analisar os resultados obtidos, tendo mais relevância a qualidade dos resultados do que a velocidade para obtê-los.

O cálculo dos *clusters* é algo que demanda muito processamento e por isso é demorado. Portanto, uma das contribuições deste trabalho é a extensão do *WF-Sim* para permitir o armazenamento dos *clusters* calculados e, assim, executar o agrupamento somente quando novas *Web forms* forem consideradas.

Duas estratégias de persistência dos *clusters* foram implementadas: a persistência em diretórios de arquivos e a persistência em um banco de dados. As próximas seções detalham estas estratégias.

3.1 Persistência em arquivos

Uma das estratégias para guardar os dados dos *clusters* foi através de arquivos gerenciados diretamente pelo *WF-Sim*. Como existem alguns parâmetros que podem variar na criação dos *clusters*, cada nova clusterização foi armazenada em pastas de acordo com os parâmetros de entrada definidos. Com isso é possível armazenar várias clusterizações com diferentes configurações.

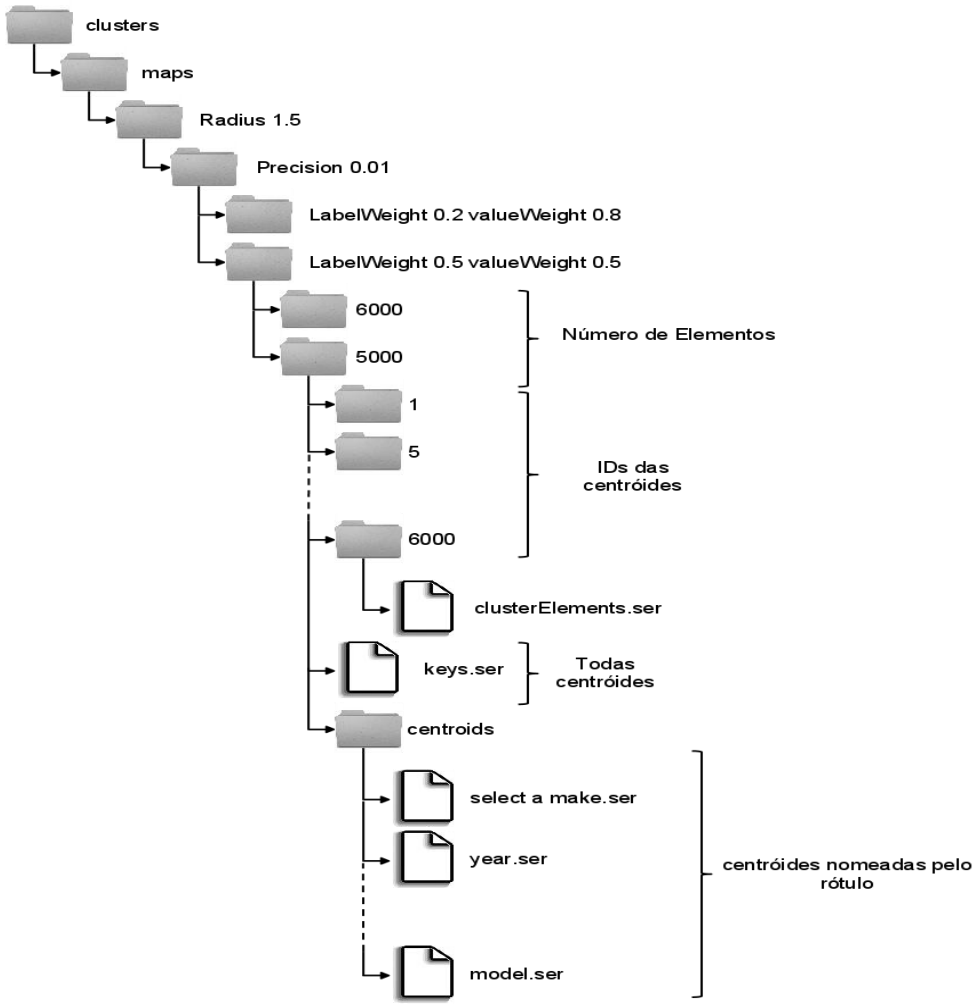

Figura 3. Organização dos *clusters* em pastas e arquivos

Como é possível ver na Figura 3, existem configurações de peso para rótulos e valores, precisão e raio de clusterização. Além disso, as clusterizações estão divididas pelo número de elementos utilizados na clusterização. Esta divisão considera outro parâmetro de configuração do *WF-Sim*, que é o número de elementos a ser carregado da base de dados de *Web forms*.

Cada clusterização gera vários *clusters* e, como visto anteriormente, cada *cluster* possui um centróide. Assim, os elementos de cada *cluster* foram armazenados em pastas nomeadas pelo identificador do centróide e os centróides foram armazenados separadamente em duas formas. A primeira forma foi um arquivo contendo todos os centróides identificados com um ID gerado pelo sistema. Desta forma, mantêm-se em memória somente as informações dos centróides e se

carrega os *clusters* desejados (mantidos em pastas específicas identificadas pelos IDs dos centróides), conforme a busca do usuário. A segunda forma foi armazenar cada centróide em um arquivo separado nomeado pelo próprio rótulo. Com isso, é possível fazer buscas acessando diretamente as pastas dos *clusters* desejados.

Para possibilitar a criação de arquivos nomeados pelos rótulos dos centróides, foi necessário um pré-processamento que retira caracteres especiais que não são aceitos pelo sistema de arquivos. Ainda, alguns rótulos possuíam muitos caracteres e por isso foram filtrados rótulos com mais de 250 caracteres.

3.2 Persistência em BD

Além da persistência em arquivos, também se decidiu testar o armazenamento dos *clusters* em um banco de dados relacional. Para isso, foi necessário o projeto do banco de dados. A modelagem conceitual definida pode ser vista na Figura 4.

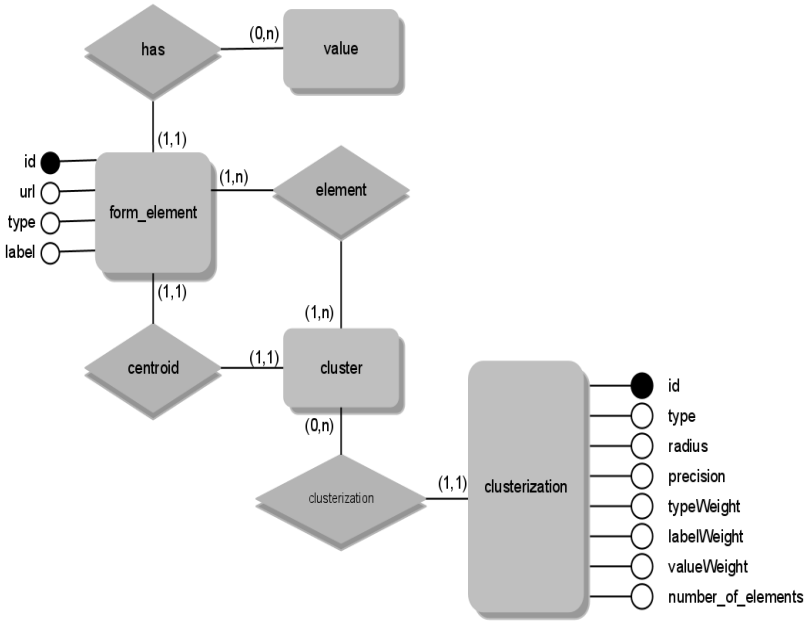

Figura 4. Modelagem conceitual da persistência dos *clusters*

Assim como no armazenamento em arquivos, encontrou-se o problema de como separar as clusterizações de acordo com cada configuração. Para resolver isso, uma entidade "*clusterization*" foi criada para manter estas configurações. Assim, cada *cluster* está relacionado a uma clusterização, podendo haver *clusters* iguais com configurações diferentes.

Como os centróides também são elementos, optou-se por não criar uma entidade centróides. Ao invés disso, a entidade '*cluster*' possui uma relação '*centroid*' com '*form_element*' e outra relação '*element'* também com '*form_element*'.

4. Experimentos

Para confirmar a melhoria de desempenho das estratégias propostas e compará-las, foram realizados alguns experimentos preliminares de busca sobre as clusterizações persistidas. Esses testes foram realizados em um computador com Intel Core2 Duo CPU T5800, 2 GHz, 3GB

RAM, Windows XP SP3, sendo 6157 o número de elementos clusterizados, divididos em 910 *clusters*.

Para a realização dos testes foi criado um formulário base (*template*) no domínio de automóveis com 10 elementos. Este é um formulário típico para busca de automóveis para venda ou aluguel. Este domínio foi escolhido pois é o que existe em maior quantidade na base de dados de *Web forms*. A Tabela 1 mostra os elementos considerados no *template* com suas restrições de valor, quando existentes.

Tabela 1. Web Form template considerado nos testes

ID	Rótulo	Valor	ID	Rótulo	Valor
1	select make	ford hyundai nissan Toyota	6	model	any model blazer caravan century civic cobalt enclave
2	min price	20,000 30,000 40,000 50,000 60,000 70,000 80,000	7	miliage	20,000 30,000 40,000 50,000 60,000 70,000 80,000
3	choose year	2000 2001 2002 2003 2004 2005 2006	8	used car	acura alfa romeo audi bmw fiat chevrolet cadillac
4	max price	200,000 300,000 400,000 500,000 600,000 700,000 800,000	9	sort order	by price by make by date by year
5	include	all vehicles new vehicles used vehicles certified vehicles	10	modified	

Para variar a quantidade de *clusters* retornados na busca, em alguns testes foram retirados elementos do *template* sendo que os mantidos foram aqueles que retornavam mais dados. Assim, testes foram feitos com 1, 5 e 10 elementos, para fins de avaliação de desempenho de um número crescente de elementos a ser considerado nas buscas.

Para encontrar os centróides desejados, tanto no banco de dados quanto nos arquivos, dividiu-se o *template* em seus elementos e, para cada elemento, foram encontrados os centróides cujos rótulos possuíssem os mesmos termos do elemento de entrada. Encontrados esses centróides, foi aplicado o cálculo de similaridade do *WF-Sim* sobre eles. Os centróides com similaridade abaixo de 0,7 foram desconsiderados para manter no resultado apenas dados com alta similaridade.

Além da busca retornando os *clusters* mais similares, também foram realizados testes utilizando o mecanismo de ranqueamento (*ranking*) do *WF-Sim*. Esse ranqueamento deve ser executado durante o processamento de cada consulta para verificar quais *clusters* são mais semelhantes à consulta. O *WF-Sim* originalmente utilizava todos os *clusters* calculados na clusterização para o ranqueamento, produzindo um *overhead* na geração do resultado da consulta. Para reduzir este *overhead*, o mecanismo de ranqueamento foi modificado neste trabalho para considerar somente os *clusters* recuperados das buscas em arquivos e no banco de dados.

Nos experimentos realizados, o foco foi a avaliação de desempenho em termos de tempo das duas estratégias. A avaliação da relevância dos *clusters* retornados não foi tratada nesse trabalho, visto que os resultados das clusterizações já são validados pelo *WF-sim*.

A Tabela 2 apresenta os resultados dos experimentos. Os tempos mostrados são a média de 3 execuções para cada busca. As consultas foram *disjuntivas*, ou seja, são retornados os *clusters* que casam com pelo menos um (1) dos elementos do *template*. Por isso, quanto mais elementos são considerados como entrada para a busca, maior é a quantidade de *clusters* recuperados.

Tabela 2. Resultados dos Experimentos

Tipo de teste	**#Elementos**	**#*Clusters* retornados**	**Tempo (seg.)**
Arquivos s/ ranking	1	9	0,968
Arquivos c/ ranking	1	9	1,135
BD s/ ranking	1	9	1,588
BD c/ ranking	1	9	1,098
Arquivos s/ ranking	5	38	2,162
Arquivos c/ ranking	5	38	2,338
BD s/ ranking	5	38	4,052
BD c/ ranking	5	38	5,031
Arquivos s/ ranking	10	65	2,807
Arquivos c/ ranking	10	65	3,135
BD s/ ranking	10	65	5,531
BD c/ ranking	10	65	5,766

Os testes com 1 elemento consideraram buscas envolvendo o elemento com ID 1 do *template* na Tabela 1. Os testes com 5 elementos consideraram os elementos com ID de 1 a 5 e os testes com 10 elementos consideraram todos os elementos do *template*.

De acordo com a Tabela 2, a estratégia de persistência em arquivos sem considerar ranqueamento do resultado foi a mais rápida, tornando-se 2,2 vezes mais lenta com 5 vezes mais elementos considerados na busca e 2,8 vezes mais lenta com 10 vezes mais elementos considerados. Isso mostra uma tendência de redução do crescimento do *overhead* à medida que o número de elementos de entrada aumenta. A estratégia de arquivos com ranqueamento ficou, em média, 1,1 vez mais lenta, representando um *overhead* pouco significativo.

A estratégia de persistência em banco de dados sem ranqueamento tornou-se 2,5 vezes mais lenta com 5 vezes mais elementos considerados na busca e 3,4 vezes mais lenta com 10 vezes mais elementos considerados. Há uma tendência semelhante à da estratégia de persistência de arquivos, apesar de haver um *overhead* maior. A estratégia de banco de dados com ranqueamento ficou, em média, para 5 e 10 elementos, 1,1 vez mais lenta, representando também um *overhead* pouco significativo. Ela só obteve desempenho superior à estratégia sem ranqueamento para a busca com 1 elemento de entrada, sendo isso provavelmente uma pequena anomalia em termos de acesso ao banco de dados.

Comparando as estratégias arquivo e banco de dados, ambas com ranqueamento, observa-se um desempenho praticamente equivalente para buscas com 1 elemento. Já para buscas com 5 e 10 elementos, a estratégia de banco de dados mostra-se, em média, 2 vezes mais lenta.

5. Conclusão

Este trabalho descreve duas estratégias para armazenamento de *clusters* de dados de *Web forms*, bem como uma avaliação de desempenho de acesso para ambas no contexto do projeto *WF-Sim*, que realiza buscas por similaridade em *Web forms*. Comparado com trabalhos relacionados (Chang and Cheng 2007; Silva 2007; Hao et. al. 2010; Hong et. al. 2010; Nguyen et. al. 2010), verifica-se que os mesmos não apresentam detalhes sobre a persistência da clusterização ou utilizam apenas bancos de dados para armazenar os *clusters* gerados. Além disso, nenhum desses trabalhos utiliza o algoritmo de clusterização proposto pelo *WF-Sim* para o contexto de *Web forms*. Este algoritmo foi escolhido por ser dirigido a dados representados na forma de grafo (estrutura de representação dos atributos das *Web forms* no *WF-Sim*) e por gerar automaticamente centróides de cada *cluster*. Essa facilidade se mostrou útil para fins de indexação, pois apenas os centróides são indexados e servem como base para buscas, ao invés de se indexar todos os elementos do *cluster*.

Analisando os resultados dos experimentos, verificou-se um bom desempenho das duas estratégias, considerando a quantidade de *clusters* a ser acessada e filtrada. Entretanto, para uma maior quantidade *clusters* retornados, a persistência em arquivos obteve melhor desempenho. Mesmo assim, é necessário avaliar melhor a escalabilidade das estratégias com um volume maior de dados, dado que existem milhões de *Web forms* disponíveis atualmente.

Quanto à utilização ou não de ranqueamento, não se notou grande diferença de desempenho, ou seja, os testes com e sem ranqueamento obtiveram tempos próximos. Conclui-se, assim, que organizar o resultado da busca não prejudica tanto o desempenho. Esse resultado

era de certa forma esperado, pois o ranqueamento trabalha com um universo de *clusters* reduzido. Como trabalho futuro, pretende-se realizar esses testes sobre uma base de dados maior e verificar se esse desempenho permanece aceitável.

Pelo fato de não existir persistência anteriormente, a velocidade de busca do *WF-Sim* melhorou consideravelmente. Com a adição dessa melhoria, o tempo de busca chegou a ter redução de horas (pois a clusterização era re-executada a cada sessão de utilização do sistema) para apenas alguns segundos, sendo tais resultados decisivos para tornar o *WF-Sim* um sistema de busca viável.

Vale lembrar que os testes realizados se referem apenas a atividades de busca. A clusterização no *WF-Sim* ainda é um processo demorado, porém, não é executado frequentemente. Mesmo assim, a persistência do resultado desse processamento melhorou a desempenho das buscas no *WF-Sim*. Conforme mencionado anteriormente, pretende-se melhorar o desempenho da clusterização e também avaliar o desempenho com uma implementação alternativa utilizando o *K-Means* (MacQueen 1967), um algoritmo popular para clusterização de dados.

Referências

Chang, K. C.-C and Cheng, T. (2007) "Entity Search Engine: Towards Agile Best-Effort Information Integration over the Web". In: Proceedings of the 2nd Conference on Innovative Data Systems Research (CIDR). p.108-113.

Cohen, W., Ravikumar, P. and Fienberg, S. (2003) "A Comparison of String Distance Metrics for Name-Matching Tasks". In: Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03). p.73-78.

Dorneles, C. F. et. al. (2004) "Measuring Similarity between Collection of Values". In: Proceedings of 6th ACM CIKM International Workshop on Web Information and Data Management (WIDM 2004). p.56-63.

Gonçalves, R. et. al. (2011). "A Similarity Search Approach for *Web forms*". In: Proceedings of the IADIS International Conference IADIS WWW/Internet.

Hao, L.; Wan-Li,Z. and Fei, R., (2010) "Describing the Semantic Relation of the Deep Web Query Interfaces Using Ontology Extended LAV". Journal of Software, v. 5, n. 1. p.89-98.

Hong, J., He, Z., and Bell, D. A. (2010). "An evidential approach to query interface matching on the Deep Web". Information Systems, v.35, n.2. p.140-148.

Jouili, S., Tabbone, S., and Lacroix, V. (2010). "Median graph shift: A new clustering algorithm for graph domain". In: Proceedings of the 20th International Conference on Pattern Recognition. p. 950-953.

MacQueen, J. B. (1967). "Some Methods for classification and Analysis of Multivariate Observations". In: Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. p. 281–297.

Nguyen, T., Nguyen, H. and Freire, J. (2010). "PruSM: A Prudent Schema Matching Approach for Web Forms". In: Proceedings of 19th ACM Conference on Information and Knowledge Management (CIKM). p.1385-1388.

Silva, A., Freire, J., and Barbosa, L. (2007). "Organizing Hidden-Web Databases by Clustering Visible Web Documents". In: Proceedings of the International Conference on Data Enginering (ICDE). p.326-335.