3 Estratégia para o enriquecimento de informações

Podemos resumir o processo de enriquecimento de informações em duas grandes etapas, a saber, busca e incorporação de dados, como ilustrado na Figura 10, a seguir.

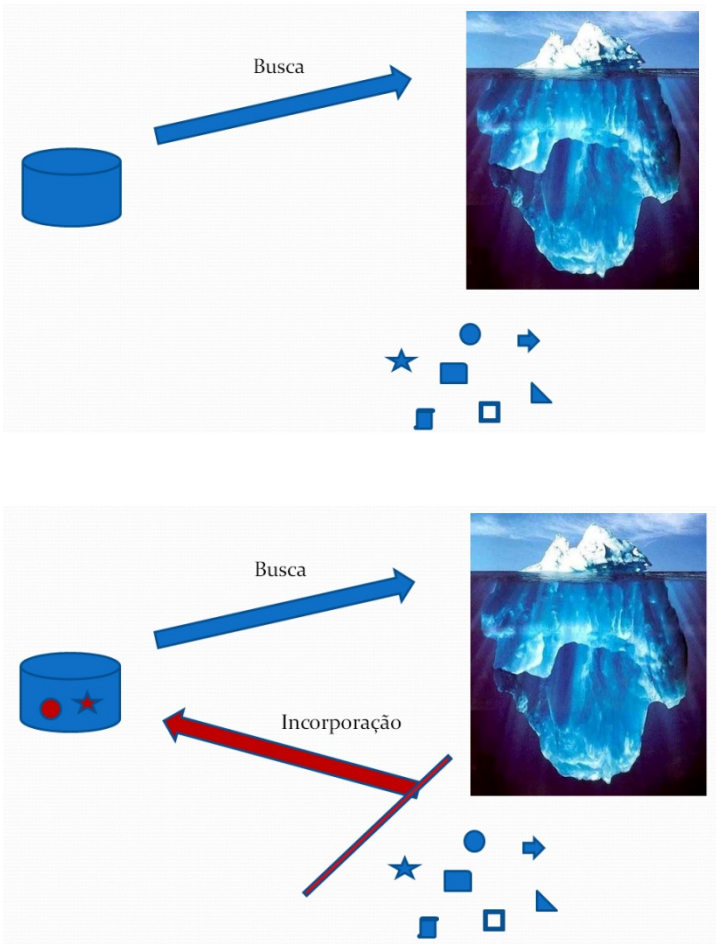

Figura 10 - Visão geral do processo de enriquecimento de informações proposto: (a) Busca, (b) Incorporação de dados

O processo de busca consiste na (1) escolha da fonte de dados, pois esta deve apresentar um formulário de entrada para realização de buscas, (2) seleção das palavras-chave adequadas ao domínio visando minimizar o número de consultas para uma determinada entidade. O processo de incorporação, por sua vez, consiste em selecionar aqueles que, com alguma segurança, representam a mesma entidade local e extrair informações adicionais que a entidade local não possui. No que se segue apresentamos as estratégias utilizadas pelos processos de busca e incorporação de informações, que constituem a estratégia de enriquecimento proposta neste trabalho. Daremos especial atenção à etapa de i-

dentificação de duplicatas, ou deduplicação, na Deep Web, contribuição principal do trabalho.

.

3.1 Busca de informações na Deep Web

A busca por informações na Deep Web, como demonstrada por trabalhos anteriores [3, 30, 32, 37], resume-se a escolher um termo (ou uma combinação de termos) e executar a consulta em um formulário de busca. Os links gerados pela busca são rastreados.

O funcionamento do nosso rastreador proposto é o mesmo do apresentado em [30]. O algoritmo contém algumas modificações de modo a receber como entrada um objeto modelo (instância da base), o qual desejamos buscar informações. Dada esta entrada é gerado um conjunto de termos para consulta. Para cada consulta é executada uma busca e os resultados são rastreados.

No nosso framework o rastreamento resulta na geração de um conjunto de objetos candidatos a duplicata. Para esse comportamento, cada página possui um *parser* específico capaz de retirar as informações que desejamos. Deste modo, o rastreador (*crawler*) só segue links para páginas que possuem um *parser* associado.

Apresentamos o algoritmo utilizado para rastreamento de links no framework:

Entrada: Instância de objeto m (modelo)

Saída: Conjunto P de produtos candidatos a duplicata

Q é um conjunto de consultas

q(m) é uma função geradora de consultas a partir do objeto modelo

Algoritmo:

```
1.  P = vazio
2.  Q = q(m)
3.  Para cada consulta de Q
4.      Executa a consulta
5.          Se a página resultado possui um parser associado
6.              Para cada produto p extraído pelo parser
7.                  P = P + p
8.                  Se p possui um link para outra página
9.                      Executa o link e vai para o passo 6.
10. Retorna P
```

Figura 11 - Algoritmo para a busca de informações na Deep Web

O objetivo principal deste algoritmo é retornar para a fase de incorporação o menor conjunto de objetos que apresentam alguma similaridade com o objeto modelo procurado. Para que isso ocorra é necessário escolher uma estratégia para a função que realiza a montagem de consultas.

Esta função é altamente dependente do domínio (significância dos termos) assim como das tecnologias envolvidas nos sites rastreados (qualidade dos mecanismos de busca). Por conta disso, a escolha da estratégia de geração de palavras-chave (termos) deve ser escolhida pelo utilizador do framework.

3.2 Incorporação de novas informações a partir da identificação de duplicatas

O foco desta etapa é encontrar uma solução para o problema de deduplicação em bancos de dados Web pertencentes a um mesmo domínio de forma a identificar informações que possam ser, posteriormente, incorporadas à base original.

Entendemos o problema de detecção de duplicatas da seguinte maneira: Dados dois registros distintos, podemos afirmar que estes representam a mesma entidade real se o resultado da aplicação de uma função de cálculo de similaridade, que tipicamente retorna um valor numérico, for maior que um limiar determinado.

Na prática há duas questões a serem resolvidas: eficiência e eficácia [8]. A eficiência refere-se à capacidade de resolver o problema utilizando os recursos computacionais de forma racional. É necessário ressaltar que já que lidamos com grandes quantidades de registros e os métodos de comparação muitas vezes têm ordem de complexidade elevada.

A eficácia refere-se à qualidade da medida de similaridade. Uma medida é eficaz se, no caso ideal, for capaz de reconhecer, em um conjunto de possíveis candidatos, todas as duplicatas e todas as não duplicatas. Central a esta questão é a determinação do limiar de detecção, que deve ser ajustado a cada domínio de aplicação [8]. Valores altos resultam em alta precisão, mas baixos *recalls*. Assim, a eficácia está ligada à capacidade de maximizar estes valores.

O resultado da detecção de duplicatas é a criação de uma identificação única para cada referência. Se duas ou mais entidades compartilham a mesma identificação, são ditas duplicatas.

A partir da revisão da literatura apresentada no Capítulo 2, da nossa experiência pregressa, e levando em consideração as necessidades apresentadas, propomos o algoritmo para identificação de duplicatas a seguir:

Entrada: Conjunto M de objetos modelo
Conjunto P de instâncias candidatas a duplicata
Saída: Conjunto D de duplicatas

C é um classificador.
f(s1,s2) é uma função de cálculo de similaridade entre strings s1 e s2

Algoritmo:

```
11. D = vazio.
12. Para cada m presente em M
13.     m é dito “similar” para C
14.     Para cada p presente em P
15.         s1 é a cadeia de caracteres que representa m
16.         s2 é a cadeia de caracteres que representa p
17.         sim = f(s1,s2)
18.         Se sim > Threshold
19.             p é dito “similar” para C
20.         Senão
21.             p é dito “diferente” para C
22. Para cada p presente em P
23.     Se p é classificado por p como “similar”
24.         D = D + p
25. Retorna D
```

Figura 12 - Algoritmo para o resolução de entidades

Para solucionar o problema de detecção de duplicatas online utilizamos utilizar um classificador (de 2 classes que aceita entradas de cadeias de caracteres) em conjunto com uma função de cálculo de similaridade entre cadeias de caracteres.

O classificar “aprende” sempre que o modelo m é um caso “similar” (positivo). Como não é possível utilizar uma base de dados confiável para treinamento, dadas as razões expostas no Capítulo 1, treinamos classificador com todos os candidatos utilizando uma função de cálculo de similaridade. Esta função pode ser vista como um clusterizador, que determina quais os candidatos podem ser vistos como exemplos “similar” (positivo) ou “diferente” (negativo).

Treinar o classificador desta forma não implica afirmar que o melhor objeto classificado é o que obteve maior similaridade, nem que somente os que têm similaridade maior que o valor limiar (*threshold)* definido serão classificados como positivos.

É preciso notar que alguns parâmetros não foram definidos no algoritmo. São o valor de corte do classificador (cutoff) e o valor limiar para a função de cálculo de similaridade (*threshold*). Estes devem ser definidos pelo utilizador do framework e é dependente do domínio escolhido.

As cadeias de caracteres s1 e s2 representam os objetos modelo e candidatos. O utilizador do framework deve implementar esta representação de acordo com o problema para que os objetos possam ter a similaridade definida.

O passo seguinte a detecção de duplicatas é a fusão dos dados. Para que haja a integração dos dados é preciso escolher uma estratégia que consiga atingir dois objetivos: completeza e concisão. A completeza é aumentada quando inserimos novos elementos ou novas fontes de dados ao processo de integração, já a concisão é alcançada quando conseguimos remover informações redundantes e mesclar informações comuns de objetos considerados duplicatas [8].

O problema central da fusão de dados é lidar com incertezas e contradições que podem ser apresentadas em dados de fontes diferentes. O processo de fusão de dados é definido como a mediação de conflitos tanto no nível dos esquemas de dados como no nível dos próprios dados [8].

3.3 Escopo do trabalho

É importante deixar claras as hipóteses sobre as quais repousa este trabalho, de modo a facilitar o entendimento do escopo da solução proposta. Em primeiro lugar assumimos a existência de um esquema global pré-definido para cada uma das fontes da Deep Web consultadas. Este esquema é compatível com o esquema utilizado pela base de origem, ou seja, conhece-se o tipo de objeto a ser retornado nas buscas e estes fazem parte do mesmo domínio dos objetos alvo cuja informação se deseja enriquecer.

Além disso, temos que todo resultado retornado por uma busca possui um *parser* específico capaz de retirar informações e criar uma instância do objeto que obedece ao esquema global. Logo, a estratégia proposta não é capaz de reconhecer, de forma totalmente automática, informações relativas ao objeto alvo.

Finalmente, também assumimos que todos os objetos retornados de uma mesma fonte são identificados univocamente pela sua URL e/ou nome, podendo assim guardar uma referência da duplicata apenas pelo seu link. Em outras palavras, cada objeto possui uma página própria com suas informações.

3.4 Resumo

Neste capítulo apresentamos uma visão geral da estratégia proposta neste trabalho para o enriquecimento de dados de uma base local a partir de informações obtidas na Deep Web.

Definimos que o processo de enriquecimento pode ser dividido em dois macro processos: (1) busca e (2) incorporação de dados. Para cada um desses processos propomos um algoritmo. O algoritmo de busca envolve o conhecimento da construção de rastreadores para Deep Web, enquanto que o algoritmo de incorporação mescla conceitos de resolução de entidades e de Fusão de Dados (*Data Fusion*).

No final da exposição discutimos o escopo da solução proposta.