Um Framework para Extração de Dados em Documentos Científicos: Uma Abordagem baseada em XML

Davi Medeiros Cabral, Roberto Souto Maior de Barros

Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7.851 – 50.732-970 – Recife – PE – Brazil

{dmc3,roberto}@cin.ufpe.br

Abstract. *Nowadays, it is very easy to publish documents in the Internet. Thus, a large number of documents become available every day, making the management and manipulation of these documents progressively more difficult. Moreover, the existence of several incompatible formats makes the development of efficient tools to deal with such documents a very complex task. This paper proposes a semantic restructure framework for the conversion of documents to XML, to make it possible the consistent use of query and transformation languages.*

Resumo. *Hoje em dia, documentos são facilmente publicados através da Internet. Assim, um grande número de documentos torna-se avaliados diariamente, fazendo o gerenciamento e a manipulação desses progressivamente mais difícil. Além disso, a existência de formatos incompatíveis faz o desenvolvimento de ferramentas eficientes para o tratamento com documentos uma tarefa altamente complexa. Este artigo propõe um framework de reestruturação semântica para a conversão de documentos para XML, fazendo possível o uso consistente de linguagens de consulta e transformação.*

1. Introdução

Com a explosão da World Wide Web, uma grande quantidade de dados em diferentes assuntos e formatos tem se tornado disponível on-line. Porém, esses dados geralmente encontram-se de uma maneira semi-estruturada ou não-estruturada, tornando as consultas dos usuários limitadas a técnicas de *browsering* ou busca por palavras-chaves. Essas técnicas não são tão eficientes, pois podem retornar uma grande quantidade de documentos irrelevantes (*precision*) ou ainda documentos relevantes podem não ser retornados (*recall*) [Aldea 2003]. Além disso, a tarefa de extração das informações fica a cargo da atividade humana, o que inviabiliza a Web Semântica de [Berners-Lee 2001].

Muitos estudos tentam dar à manipulação em documentos o mesmo poder dos SGBDs, dentre as quais pode-se citar: (1) a criação de uma linguagem de consulta para documentos Web (ex. [Arocena 1998]) e (2) a popularização de um banco de dados a partir de dados extraídos de fontes da Internet, possibilitando tratamento posterior.

Uma solução comum para essa última estratégia é o desenvolvimento de programas especializados, denominados *wrappers*, criados para identificar dados de fontes da *Web* e mapeá-los para um formato apropriado.

Os *wrappers* fornecem o suporte necessário para alcançar consultas mais precisas e eficientes em documentos. Porém, a maioria deles sofre por confiar na ordem dos dados ou na estrutura dos documentos (ex. HTML) da Internet, acarretando em grande esforço de manutenção quando características estruturais são modificadas ([Embley 1999]). Além disso, as abordagens estudadas realizam extração de dados apenas em documentos nos formatos HTML, XML ([Bray 2004]) ou texto, esquecendo-se que a Internet é muito mais heterogênea, abrangendo formatos como PDF, DOC, LaTeX, etc. Um outro ponto diz respeito aos núcleos dos *wrappers* que em um mesmo sistema normalmente são semelhantes para diferentes fontes de informação, ocasionando duplicações de código e, conseqüentemente, de esforço de programação.

Este artigo busca solucionar os problemas citados propondo um *framework* para extração e reestruturação de dados que otimize reuso e extensão para diferentes formatos de documentos científicos na Web. A flexibilidade da ferramenta é alcançada a partir da modularização e da comunicação de componentes através de XML.

O restante deste documento está organizado da seguinte forma: a seção dois apresenta uma breve discussão sobre técnicas e modelos de sistemas para extração de dados; a seção três apresenta o modelo proposto e sua implementação. Por último, a seção quatro fala sobre as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

O desenvolvimento manual de *wrappers* possui barreiras bem conhecidas. Atualmente, muitas propostas têm surgido para solucionar essas dificuldades, sendo classificadas aqui de acordo com o tipo de técnica de extração e com sua utilização.

2.1. Mecanismos de Extração

Segundo [Laender 2002], uma ferramenta de geração de *wrapper* pode ser classificada de acordo com seis diferentes técnicas de extração utilizadas. Porém, por questão de espaço e escopo de trabalho apenas as técnicas extração NLP, Indução e Ontologia voltadas para documentos livres e semi-estruturados serão apresentadas.

Ferramentas baseadas em Processamento de Linguagem Natural (NLP) buscam obter regras de extração de dados em documentos escritos em linguagem natural, aplicando técnicas de filtragem, tipos de palavras e classes semânticas. Essas técnicas são utilizadas na construção de relacionamentos entre frases e sentenças de elementos utilizados na derivação de regras de extração. Como exemplos de ferramentas podem ser citados o SRV [Freitag 2000] e o WHISK [Soderlan 1999].

Ferramentas de indução de *Wrapper* geram regras de extração baseadas em delimitadores (prefixos e sufixos) derivados de um dado conjunto de exemplos de treinamento. Diferentemente das ferramentas NLP, essa abordagem confia em características de formatação que implicitamente descrevem a estrutura de pedaços de dados encontrados — ferramentas representativas são o WIEN [Kushmerick 2000] e o STALKER [Muslea 2001].

Ferramentas baseadas em ontologia utilizam uma ontologia de domínio específico para localizar constantes presentes em um documento e assim construir objetos com elas. Uma ontologia é "uma especificação explícita e formal de conceitos" ([Gruber 1993]), um meio de representação do conhecimento. Essa abordagem difere das demais, por não confiar na estrutura de apresentação dos dados nos documentos. Assim, ela é mais flexível e abrangente que as demais abordagens, porém possui uma aprendizagem de regras menos automatizada. Um exemplo dessa abordagem é o sistema apresentado em [Embley 1999].

2.2. Utilização de Wrappers

Essa seção traz uma visão geral da utilização de *wrappers* em sistemas de extração de dados em documentos. Aqui serão definidas três estruturas evolutivas de utilização de *wrappers* (vê figura 1), as quais podem ser desenvolvidas utilizando, por exemplo, agentes ou aplicações *stand-alone*.

A abordagem simples (figura 1a) é a mais elementar, preocupando-se apenas com a conversão direta entre dois formatos diferentes de dados de um documento HTML para tuplas de um SGBD, por exemplo. Uma ferramenta representativa dessa abordagem é [Embley 1999].

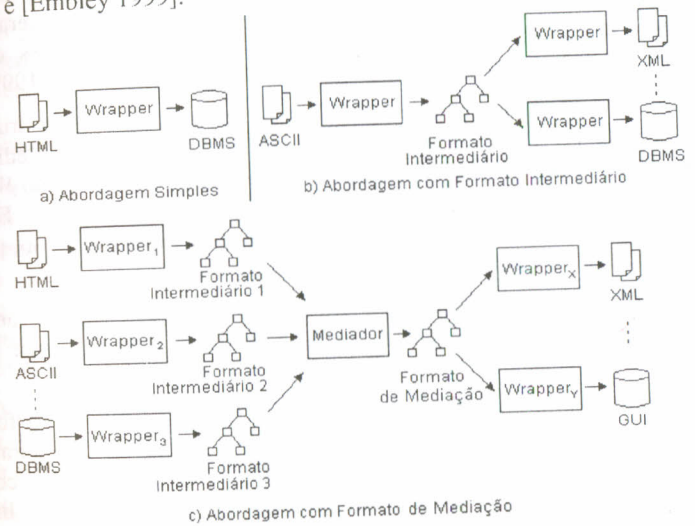

Figura 1 — Abordagens de utilização de *wrappers*

Para tornar a estrutura anterior mais flexível, é adicionado a ela um formato de representação intermediário para os dados extraídos (figura 1b). Esse formato pode ser uma interface de manipulação de objetos, um arquivo (ex. XML, OEM [Hammer 1997]) etc. Essa modificação possibilita estender facilmente o sistema de extração com formatos diferentes de saídas de dados, a partir de *wrappers* simples, que convertem o formato intermediário interno em um outro.

A estratégia da figura 1c amplia a abordagem anterior para diferentes tipos de entradas. Nesse caso, surge um componente especializado, denominado mediador, responsável por intermediar consultas e dados entre os formatos gerados pelos *wrappers* de entrada e a interface de entrada padrão para os *wrappers* de saída. Portanto, para adicionar outro *wrapper* de entrada, o módulo mediador deve ser alterado ou empilhado a um outro, desenvolvido para o reconhecimento desse novo formato intermediário. Porém, isso pode se tornar difícil à medida que novas entradas são inseridas.

Ainda com relação a essa estratégia, pode-se notar que os formatos intermediários possuem em comum a forma de representação dos dados extraídos (ex. XML, OEM), mas diferem na estrutura de como esses dados são modelados (ex. granularidade, relacionamento entre dados etc.). Essa peculiaridade ocasiona um aumento de esforço significativo no desenvolvimento dos mediadores, obrigando-os a solucionar heterogeneidades estruturais e semânticas dos dados ([Wache 2001]). Um exemplo de ferramenta de extração desse tipo é o TSIMMIS [Hammer 1997].

3. Modelo Proposto

Essa seção apresenta o *framework* proposto e para tanto é preciso entender o que esse termo significa. Um *framework* tem por objetivo auxiliar a construção de aplicações complexas pertencentes a um determinado domínio, através da definição de um modelo que possibilite a reutilização de código. Um dos pontos essenciais de um *framework* é a definição de suas funcionalidades *frozen spots* e *hot spots*. As *frozen spots* são as funcionalidades fixas, comuns ao domínio, constituindo a parte inalterada de um *framework*. Por outro lado, as funcionalidades *hot spots* são aqueles em aberto, específicas de cada domínio, e que necessitam ser implementadas [Fayad 1999].

O *framework* proposto (vê figura 2) é baseado na abordagem de estruturação de *wrappers* da figura 1c, sendo diferenciada pela ausência do módulo Mediador, pela adição do módulo Alimentador de Conhecimento e pela quebra do módulo *Wrapper* de Entrada em três outros: *Wrapper* Simples de Entrada, Reestruturador e Extrator de Dados. Essa estratégia modular torna mais flexível o problema de extração, permitindo assim a reutilização do módulo de extração de dados.

Na abordagem da figura 2, os *Wrappers* de Entradas são mecanismos de encapsulamento e reuso de ferramentas existentes para conversão entre diferentes formatos.

Os módulos Reestruturadores são utilizados na padronização do formato de apresentação dos documentos, solucionando heterogeneidades estruturais. Esses módulos diferem-se do módulo mediador da figura 1c por não gerenciar consultas a dados em diferentes fontes (*Wrappers* de Entrada). Um Reestruturador limita-se a converter o que é retornado por um *Wrappers* de Entrada em arquivos (XML Unificado) usando conceitos semanticamente pobres como parágrafo, figura, tabela etc., tornando essa estratégia menos complexa do que as de desenvolver ou modificar um mediador.

O módulo Extrator de Dados implementa uma das técnicas apresentadas na seção 2.1, sendo o responsável pela identificação e obtenção de informações semanticamente ricas como autor, orientador, título, ano de publicação e referência bibliográfica, entre outras. Os metadados necessários a esse processo da extração são retirados da base externa de conhecimento, escrita em XML. Esse é o módulo mais complexo do modelo e, por isso, a preocupação em projetá-lo de maneira centralizada.

Por último, o Alimentador de Conhecimento é o responsável pela alimentação semi-automática da base de conhecimento do sistema e os *wrappers* de saída são os mesmos apresentados na figura 1c.

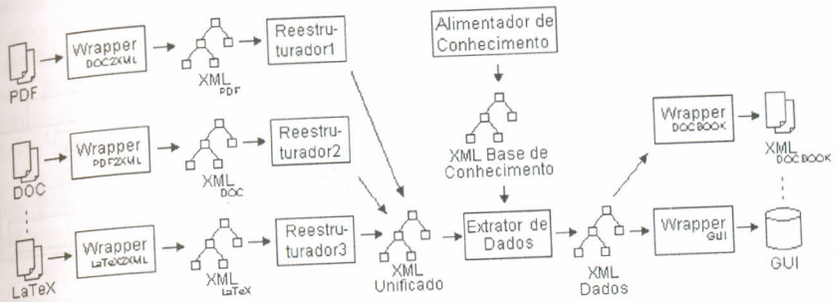

Figura 2 — Modelo de sistema de extração para documentos científicos

Por serem gerais e estáveis, os módulos Extrator de Dados e Alimentador de Conhecimento constituem a parte *frozen spots*, enquanto que os demais módulos são responsáveis pelas funcionalidades *hot spots* do *framework*.

3.1. Implementação dos Módulos

A implementação foi utilizada como módulo de conversão de documentos científicos para o formato XML, necessário ao sistema de gerenciamento e publicação de teses e dissertações proposto por [Santos 2002]. A escolha por esse tipo de documento foi devido a sua riqueza de informações e formato não-estruturado, que acarreta na dispersão das informações dentro do texto. Essa característica é presente na maioria dos documentos Web, possibilitando a utilização do conhecimento desenvolvido na conversão de documentos com outros tipos de conteúdo.

O protótipo foi implementado como um *framework* de caixa branca, ou seja, a partir da definição de classes abstratas que devem ser especializadas para cada novo caso. Em geral, essas classes são definidas utilizando o padrão *Template Method*.

Para cada módulo do modelo proposto, com exceção do módulo Alimentador de Conhecimento, foi implementado um protótipo. Os módulos foram desenvolvidos utilizando a linguagem Java, dando origem aos seguintes pacotes (figura 3): WrapperEntrada, Reestruturador, MaquinaExtracao e WrapperSaida. A manipulação (escrita e leitura) dos documentos XMLs entre os módulos foi realizada utilizando o *data binding* Castor [Gilmartin 2002], cujos modelos de objetos para a representação dos dados e os respectivos *parsers* foram definidos nos subpacotes ModeloUnificado, ModeloDados, ModeloEntrada e ModeloSaida[1]. Adicionalmente, esses dois últimos subpacotes devem ser expandidos para cada novo formato de arquivo a ser utilizado pelo sistema.

[1] Os modelos de objetos foram definidos e analisados utilizando, respectivamente, os padrões *Composite* e *Visitor*.

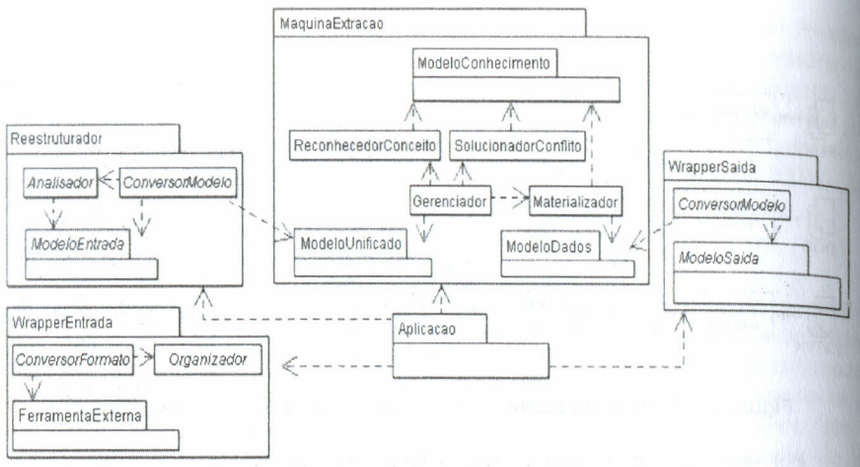

Figura 3 — Diagrama de classes do sistema

O módulo WrapperEntrada é composto basicamente pelas classes abstratas ConversorFormato e Organizador, que deverão ser estendidas para cada novo formato de entrada suportado. A classe ConversorFormato é a responsável por encapsular uma ferramenta de conversão, pacote FerramentaExterna, que transforme arquivos em formato específico para XML. Baseada na idéia do HTML *Tidy* [Raggett 1994] para correção de arquivos HTML, a classe Organizador deve ser implementada para a correção dos possíveis XMLs mal formados — gerados pelo ConversorFormato. Os formatos de arquivos de entrada implementados para o sistema foram DOC e PDF, através do encapsulamento das ferramentas de conversão Majix[2] e Pdftohtml[3], respectivamente.

Dando continuidade aos módulos, o pacote Reestruturador é dividido nas classes abstratas Analisador e ConversorModelo. A primeira classe é responsável por analisar o arquivo XML, gerado por um *wrapper* de entrada, e por quantificar a ocorrência de certos atributos como distância entre linhas e valores para a margem esquerda, por exemplo. O ConversorModelo utiliza os valores retornados pelo Analisador e aplica regras heurísticas para reestruturar o documento em um XML Unificado definido na seção 3. Isso foi necessário porque, por exemplo, a ferramenta Pdftohtml retornava um documento XML sem o conceito de parágrafos, dividindo uma possível informação em linhas diferentes.

Com relação ao pacote MaquinaExtracao, a estratégia utilizada foi baseada no uso de ontologia proposto em [Embley 1999]. O módulo MaquinaExtracao é composto pelas classes ReconhecedorConceito, SolucionadorConflito, Gerenciador e Materializador, além dos subpacotes ModeloConhecimento, ModeloUnificado e ModeloDados.

A abordagem de extração de dados consiste de cinco etapas: (1) criação de um modelo ontológico sobre uma área de interesse, subpacote ConversorModelo; (2) obtenção do texto não-estruturado, mas com parágrafos, através de uma solicitação recursiva ao método getText[4]; (3) invocação da classe ReconhecedorConceito que usa as regras de combinação informadas na ontologia (constantes e palavras-chaves) para extrair de cada documento não-estruturado os objetos esperados; (4) análise heurística (classe SolucionadorConflito) dos resultados da etapa 3, correlacionando palavras-chaves com as constantes extraídas, usando o conjunto de relacionamentos e restrições de cardinalidade da ontologia — para se determinar quais dados serão considerados ou não; (5) análise dos resultados do SolucionadorConflito para composição do XML de Dados pelo Materializador. Adicionalmente, os quatro últimos passos são orquestrados pelo Gerenciador.

Por último, o pacote WrapperSaida é o responsável pela conversão do modelo de dados em XML para um formato de saída específico (subpacote ModeloSaida) como, por exemplo, registros em um banco de dados. Essa operação é realizada por uma classe abstrata, nomeada ConversorModelo, semelhante a do pacote Reestruturador, definida através da utilização do padrão *Builder*.

4. Conclusões e Trabalhos Futuros

Nesse artigo, foi realizado um estudo sobre *wrappers*, culminando na apresentação e implementação de um novo modelo de extração de dados baseado em XML. Esse sistema dá o suporte necessário ao desenvolvimento de ferramentas mais eficientes para o gerenciamento, consulta e publicação de documentos. Além disso, o modelo proposto permite ser adaptado para sistemas de classificação independentes (DocBook[Walsh 2003]), bem como modelos nacionais e internacionais (ex. ABNT).

O desenvolvimento desse sistema mostrou a flexibilidade trazida pelo uso de XML como linguagem de representação de metadados e de comunicação entre os módulos do protótipo. Além disso, a representação em XML dos dados extraídos, juntamente com a utilização de ferramentas XSLT e XSL-FO, possibilita a criação rápida de *wrappers* de saída para uma considerável variedade de documentos em formatos distintos de apresentação.

Atualmente, o protótipo desenvolvido encontra-se implementado apenas para os formatos DOC e PDF. Futuramente pretende-se realizar a expansão do sistema para suportar outros tipos de entrada (ex. LaTeX, XHTML etc.) e publicações (ex. livros, artigos, etc.). Além disso, poderiam ser feitas melhorias como, por exemplo, a utilização de *parser* DOM *Level 3* na manipulação de arquivos XML e de *Web Services* no desenvolvimento dos módulos do sistema, contribuindo respectivamente na eliminação dos problemas trazidos com *data binding*[5] e no aumento da flexibilidade do modelo.

Com relação à base de conhecimento, poderia ser realizado um estudo sobre a viabilidade de representação do seu conhecimento em XMI, o que permitiria a alimentação da mesma a partir de uma ferramenta de modelagem UML (ex. ArgoUML) ou de um módulo alimentador de conhecimento utilizando técnicas de aprendizagem da Inteligência Artificial. Por último, seria interessante tornar possível a utilização de outras técnicas de extração de dados como, por exemplo, NLP ou indução.

[2] Outros dados em http://www-900.ibm.com/developerWorks/cn/xml/tips/x-tipword/index_eng.shtml
[3] O projeto Pdftohtml encontra-se hospedado em http://sourceforge.net/projects/pdftohtml
[4] Esse método é definido na classe base do padrão *Composite*, usado na implementação dos modelos.
[5] Pode-se destacar a incapacidade do Castor em lidar com *tags* mistas e hierarquias complexas de objetos.

References

Aldea, A., Bañares-Alcántara, R., Bocio, J., Gramajo, J., Isern, D., Kokossis, A., Jiménez, L., Moreno, A. and Riaño, D. (2003) "An Ontology-Based Knowledge Management Platform", In: 18th IJCAI. IIWeb-03.

Arocena, G. and Mendelzon, A. (1998) "WebOQL: Restructing Documents, Databases, and Webs", In: Proceedings of the 14th ICDE. Orlando, Flórida.

Berners-Lee, T., Hendler, J. and Lassila, O. (2001) "The Semantic Web", In: Scientific American.COM., http://www.sciam.com/print_version.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21.

Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E. and Yergeau, F. (2004) "Extensible Markup Language (XML) 1.0 (Third Edition)", In: W3C Recommendation, http://www.w3.org/TR/2004/REC-xml-20040204.

Embley, D., Campbell, D., Jiang, Y., Liddle, S., Lonsdale, D., Ng, Y. and Smith, R. (1999) "Conceptual-Model-Based Data Extraction from Multiple-Record Web Pages".

Fayad, M., Schmidt, D., Johnson, R. (1999) "Building Application Frameworks: Object-Oriented foundations of frameworks design". Wiley, Estados Unidos.

Freitag, D. (2000) "Machine Learning for Information Extraction in Informal Domains", In: Machine Learning 39.

Gilmartin, A., Horowitz, R. and Suez, E. (2002) "Castor: An Open Source Data Binding Framework for Java", In: ExoLab Group, http://www.castor.org, February.

Gruber, T. (1993) "A translation approach to portable ontology specifications", In: Kknowledge Acqui-sition, 5(2):199-220.

Hammer, J., McHugh, J. and Gracia-Molina H. (1997) "Semistructured Data: The TSIMMIS Experience", In: Proceedings of the 1st ADBIS. St. Petersburg, Russia.

Kushmerick, N. (2000) "Wrapper Induction: Efficiency and Expressiveness", In: Artificial Intelligence Journal 118.

Laender, A., Ribeiro-Neto, B., Silva, A. and Teixeira, J. (2002) "A Brief Survey of Web Data Extraction Tools".

Muslea, I., Minton, S. and Knoblock, C. (2001) "Hierarchical wrapper induction for semistructured information sources", In: Autonomous Agents and Multi-Agent 4.

Raggett, D. (1994) "Clean up your Web pages with HTML TIDY", http://www.w3.org/People/Raggett/tidy, January.

Santos, H., Batista, M. and Barros, R. (2002) "Publishing Theses and Dissertations: An Approach Using XML", In: The 4th IIWAS.

Soderlan, S. (1999) "Learning Information Extraction Rules for Semi-Structured and Free Text", In: Machine Learning 34.

Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H. and Hübner, S. (2001) "Ontology-Based Integration of Information — A Survey of Existing Approaches".

Walsh, N. and Muellner, L. (2003) "DocBook: The Definitive Guide". O'Reilly.