Extração Automática de Conteúdo Semi-Estruturado na Web: Estudo de Caso do Futebol Brasileiro

Alexandre S. de Melo[1], Hendrik T. Macedo[2]

[1]*Departamento de Ciência da Computação - Universidade Federal de Minas Gerais (UFMG), Belo Horizonte, Brasil*

[2]*Departamento de Computação – Universidade Federal de Sergipe (UFS), São Cristóvão – SE – Brasil*

asmelo@dcc.ufmg.br, hendrik@ufs.br

(Recebido em 25 de março de 2009; aceito em 21 de agosto de 2009)

Técnicas de Extração de Informação possibilitam geração automática de uma representação estruturada a partir de conteúdo não estruturado ou semi-estruturado. Informação estruturada possibilita ou facilita seu processamento por parte de aplicações Web diversas. Este trabalho descreve a implementação de um sistema de extração automática de informação semi-estruturada na Web orientada a domínio. O sistema utiliza regras de produção baseada em objetos que produzem instâncias de classes que representam o domínio considerado. O sistema faz uso da API JEOPS, um motor de inferência de primeira ordem com encadeamento progressivo integrado à linguagem Java. Como estudo de caso, foi definido classes que representam o Campeonato Brasileiro de Futebol. O sistema recebe como entrada o endereço eletrônico de um portal Web e, fazendo uso de fatos e regras de sua base de conhecimento relacionada ao Campeonato Brasileiro, identifica links relacionados e navega no portal a fim de localizar a tabela de classificação do campeonato e extrair dados da tabela, produzindo de forma automática instâncias das classes especificadas.

Palavras-chave: Extração de Informação, Regras de Produção, JEOPS, Wrapper, Crawler.

Information extraction techniques provide automated generation of a structured representation from unstructured or semi-structured content. Structured information enables or facilitates further processing by third-part Web applications. This work describes the implementation of a domain-oriented information extraction system. The system automatically converts semi-structured Web content into structured content, by means of object-oriented production rules that instantiate a specific domain classes provided. These rules are implemented in JEOPS, a Java-based first-order forward chaining inference engine. We have fully specified classes modeling the Brazilian Soccer Championship to show the feasibility of the proposal. Taking as input a Web site address, the system uses facts and rules defined in its knowledge base in order to identify related links, find the championship classification table and extract table data. As a result, it automatically fulfills domain classes' instances.

Keywords: Information Extraction, Production Rules, JEOPS, Wrapper, Crawler.

1. INTRODUÇÃO

Nos últimos anos, o número de informações disponíveis na Internet tem crescido bastante. Através de uma máquina conectada à Internet uma pessoa pode ter acesso a jornais eletrônicos, revistas, anúncios de emprego, artigos científicos, imagens das obras de uma famosa galeria de artes ou às informações de um campeonato de futebol. Entretanto, as informações presentes na Internet usualmente encontram-se desorganizadas, sem qualquer padronização. Seria conveniente que estes dados estivessem estruturados para servir como entrada para uma infinidade de sistemas que realizam processamento automático.

Devido a esta ausência de estruturação das informações na Web, a pesquisa na área de extração de informação na Web tem se desenvolvido bastante na última década. O objetivo dos sistemas de extração de informação é a organização de dados. A extração de informação na Web consiste fundamentalmente em transformar uma determinada informação presente na Internet em um formato estruturado, gerando assim a possibilidade do processamento desses dados por sistemas tradicionais.

Um programa que visa realizar a extração de informação de um texto é chamado de wrapper [3]. A entrada para um wrapper pode ser um texto estruturado, semi-estruturado ou livre. Textos estruturados possuem formato pré-definido. Nesse tipo de texto pode-se prever exatamente onde buscar uma determinada informação. Já o texto livre, ao contrário do estruturado, não possui formato pré-definido, apenas segue as regras gramaticais de um determinado idioma. A extração de informação sobre esse tipo de texto geralmente baseia-se em técnicas de linguagem natural, que usa a relação sintática entre as palavras para extrair o significado de uma expressão. O texto semi-estruturado, por sua vez, possui marcadores que buscam explicitar e identificar estruturas no documento. Contudo, textos desse tipo apresentam irregularidades como campos ausentes ou com valor nulo, variações na ordem dos dados e ausência de delimitadores entre as informações a serem extraídas.

Existem duas estratégias para construção de wrappers que irão extrair informações destes textos. A primeira é baseada na engenharia de conhecimento e outra na geração automática de regras.

Regras dos sistemas baseados em engenharia de conhecimento podem ser definidas através de regras de produção, ao invés de uma simples especificação ad hoc. As regras de produção são utilizadas predominantemente para representar um conhecimento heurístico sobre o mundo, especificando um conjunto de ações que devem ser realizadas para uma dada situação. Uma regra de produção é composta por uma parte de condições e uma parte de ações. A ação da regra somente será executada se a condição for verdadeira. As regras de produção permitem uma representação modular do conhecimento, ou seja, cada regra representa uma parte do conhecimento de forma independente. Além disso, são fáceis de compreender e de modificar. Novas regras podem ser inseridas, enriquecendo assim a base de conhecimento do sistema.

Já nos sistemas com geração automática de regras, diversos autores aplicam algoritmos de aprendizagem de máquina baseados em casamento de padrões [14], [2] ou baseados em autômatos finitos, como [9], [7], [8] e [6]. O uso das Cadeias Escondidas de Markov também têm apresentado bons resultados [10], [1], [13]. Essas técnicas utilizam uma base de treinamento para identificar padrões e gerar regras de forma automática. Outras técnicas de aprendizagem de máquina como aprendizado simbólico, métodos estatísticos, gramática indutiva e programação lógica indutiva também têm sido aplicadas no problema de extração de informação [3].

A proposta deste trabalho é construir um sistema de extração de informações da Web capaz de estruturar, de forma automática, as informações extraídas. A entrada do sistema consiste de um endereço eletrônico referente a um portal Web, onde o sistema deverá identificar links relacionados com o domínio em questão, até que encontre uma página com as informações que se deseja extrair. Em seguida estas informações devem ser extraídas para serem estruturadas em instâncias de classes que representam o domínio. A estruturação das informações consiste em primeiramente definir classes para representar o domínio da aplicação, e então gerar instâncias dessas classes a partir das informações extraídas. Trata-se de um wrapper com um crawler incorporado ao seu código. Um crawler é uma aplicação que navega automaticamente por vários sites, através de links que os interligam, até acessar um site alvo. Alguns trabalhos como [16] e [17] desenvolveram crawlers para acessar informações específicas. Contudo, estes trabalhos apenas identificam o site que possui uma informação pré-definida. Neste presente trabalho nos propomos a não apenas identificar automaticamente um site específico, mas também a extrair as informações úteis do site identificado.

2. EXTRAÇÃO DE INFORMAÇÃO

A proposta deste trabalho é construir um sistema de extração de informações da Web capaz de estruturar, de forma automática, as informações extraídas. A entrada do sistema será um endereço eletrônico de um portal Web, onde o sistema deverá identificar links relacionados com o domínio em questão, até que encontre uma página com as informações que se deseja extrair. Em seguida estas informações devem ser extraídas para serem estruturadas em instâncias de classes que representam o domínio. A estruturação das informações consiste em primeiramente definir classes para representar o domínio da aplicação, e então gerar instâncias dessas classes a

partir das informações extraídas. A figura 1 ilustra as classes definidas para representação do domínio, expressas com a sintaxe de UML/OCL.

Foi definido como alvo da extração de informação os dados de uma tabela de classificação do Campeonato Brasileiro de futebol do ano corrente. O processo de extração de informação dos dados da tabela consiste em primeiramente analisar a página Web passada como parâmetro de entrada, a fim de localizar a tabela de classificação. Em seguida, deve-se mapear os seus dados como nome do time, pontos ganhos, saldo de gols, vitórias, etc. em instâncias das classes definidas.

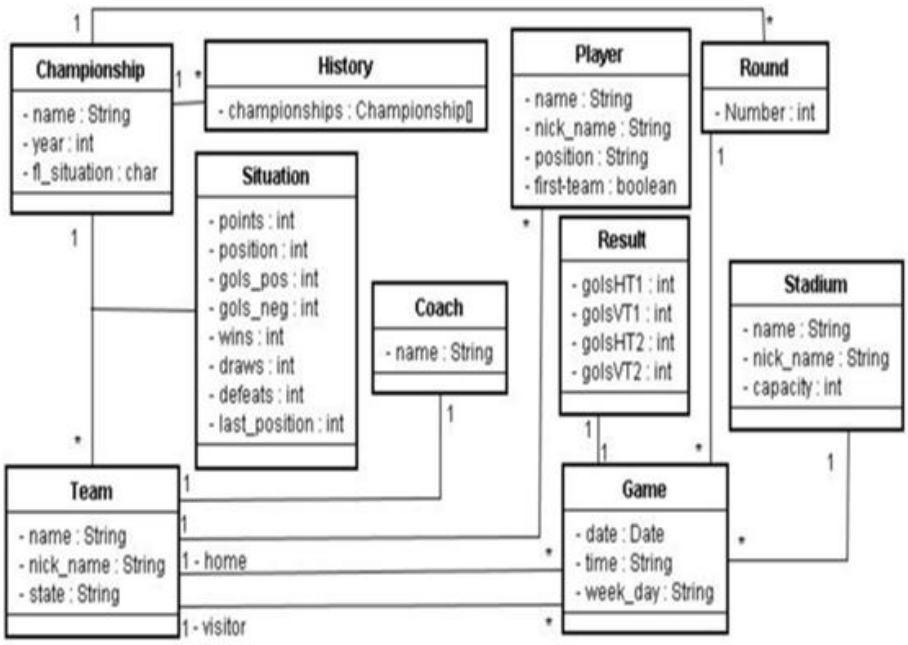

Figura 1: Modelagem do Campeonato Brasileiro de Futebol.

A principal dificuldade desta tarefa está no fato de que as informações em cada página são dispostas de forma aleatória, sem padronização alguma. A figura 2 contém dois exemplos de tabela de classificação retiradas da Web, onde se pode notar como os títulos das tabelas são diferentes, apesar de conterem, a princípio, as mesmas informações.

TIME		PG	J	V	E	D	GP	GC	SG
1	São Paulo	47	22	14	5	3	30	7	23
2	Cruzeiro	39	21	12	3	6	49	37	12
3	Vasco	38	21	11	5	5	39	22	17
4	Botafogo	38	22	10	8	4	42	30	12
	...								

PG - pontos ganhos; **J** - jogos; **V** - vitórias; **E** - empates; **D** - derrotas; **GP** - gols pró; **GC** - gols contra; **SG** - saldo de gols

P	Equipes	PTS	J	V	E	D	GP	GC	S
1	São Paulo	47	22	14	5	3	30	7	23
2	Cruzeiro	39	21	12	3	6	49	37	12
3	Vasco	38	21	11	5	5	39	22	17
4	Botafogo	38	22	10	8	4	42	30	12
5	Santos	36	22	11	3	8	36	28	8
6	Palmeiras	36	22	10	6	6	29	26	3
	...								

Figura 2: Exemplos de tabelas de classificação.

A primeira tabela utilizou o título PG para os pontos ganhos, enquanto que a segunda utilizou PTS. A primeira tabela apesar de também conter a posição dos times não colocou nenhum título para esta coluna, enquanto que a segunda tabela utilizou a abreviação P, referindo-se a posição.

Caso essas informações fossem padronizadas, para extrair suas informações automaticamente, bastaria identificar a estrutura da tabela padrão e mapear seus valores para instâncias de classes.

Para localizar a tabela de classificação o sistema deverá identificá-la na página Web passada como entrada. Caso não seja encontrada, o sistema deverá analisar os links do site, os quais poderão dar acesso a outras páginas Web que contenham a tabela de classificação. O número de links existentes necessários para se atingir a tabela pode variar de muito de um portal para outro.

Para estruturar as informações extraídas, foram criadas duas classes: Time e Situacao, que possuem atributos referentes às informações de uma tabela de classificação. No processo de estruturação das informações, para cada time da tabela de classificação será criado um objeto do tipo Time e outro do tipo Situacao, associado ao primeiro. A figura 3 ilustra um esquema para mapear as informações extraídas em classes estruturadas, e a figura 4 contém o diagrama de objetos UML de alguns dos objetos criados nesse esquema.

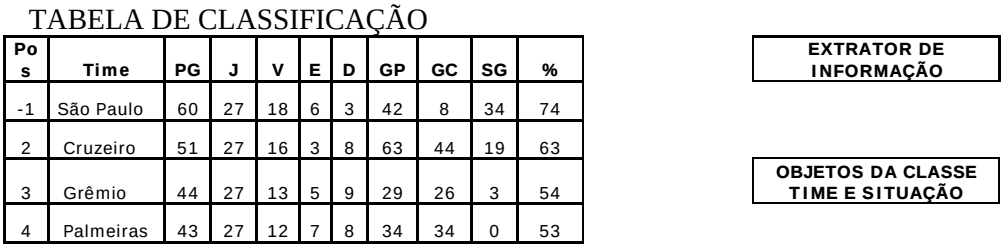

Figura 3: Mapeamento das informações extraídas em classes estruturadas.

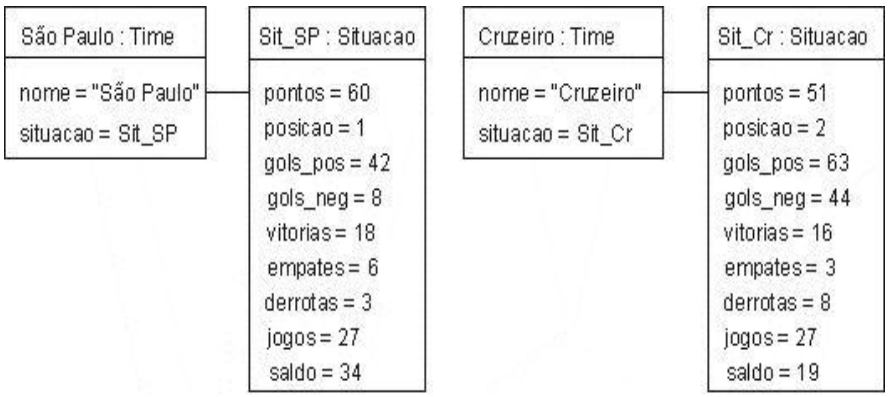

Figura 4: Parte das instâncias de classes preenchidas.

2.1. Metodologia

O processo proposto para a extração de informação consiste nos seguintes passos:

- Transformar o código HTML em XHTML (eXtensible Hypertext Markup Language). Esta é uma reformulação da linguagem de marcação HTML baseada em XML, que combina as tags de marcação HTML com regras da XML. Este processo visa corrigir possíveis erros no código HTML, padronizando assim o código das páginas que serão analisadas.
- Buscar por links relacionados com o domínio.
- Para cada link encontrado, o código associado será transformado em XHTML.
- Em cada página localizada é verificada a existência da tabela de classificação.
- Quando a tabela de classificação é identificada, é realizado o mapeamento das informações em objetos de classes estruturadas.

Este processo será interrompido quando a tabela da classificação for encontrada ou nenhum link seja identificado. A figura 5 ilustra todo o processo.

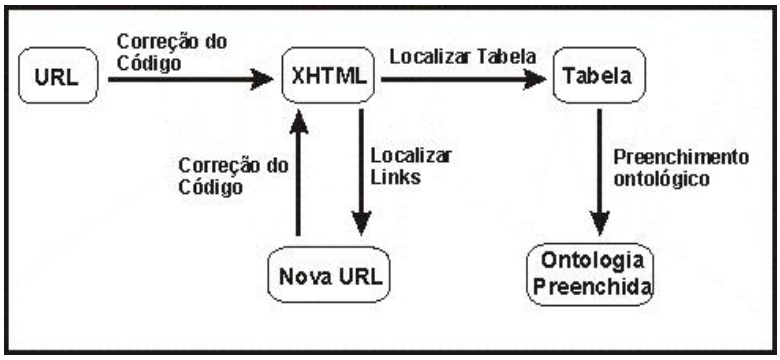

Figura 5: Processo de extração de informação proposto.

3. IMPLEMENTAÇÃO

Para a implementação do extrator de informações do campeonato brasileiro, foram utilizadas diversas ferramentas, assim, antes da descrição da implementação, será feita uma breve descrição destas ferramentas.

3.1. Descrição das Ferramentas

Todas as regras do sistema foram implementadas numa base de regras de produção. Para implementação das regras de produção foi utilizado o JEOPS (Java Embedded Object-Oriented Production Systems).

3.1.1 JEOPS

O JEOPS é um motor de inferência de primeira ordem com encadeamento progressivo integrado à linguagem Java [11], este faz parte da classe de sistemas EOOPS (Embedded Object-Oriented Production Systems) cujo termo é usado para caracterizar os sistemas que integram linguagens orientadas a objetos com sistemas de produção, ou seja, juntam as vantagens de sistemas inteligentes com as vantagens das linguagens orientadas a objetos, tais como modularidade, compreenssibilidade, legibilidade, manutenibilidade e reusabilidade. O JEOPS foi desenvolvido em 1998 na Universidade Federal de Pernambuco [4].

O JEOPS é um pré-compilador que traduz um arquivo de regras em uma classe Java que implementa o motor de inferência de acordo com as regras do arquivo original. Uma regra JEOPS é composta por declarações, condições e ações. As declarações especificam os objetos que serão utilizados nas condições ou ações. As condições são expressões Java que retornam um valor boleano. Finalmente as ações são comandos Java que serão executados se todas as condições forem verdadeiras. A regra será executada para cada instância dos objetos presentes na seção de condições.

No estudo de caso, as regras de produção, implementadas com o JEOPS, analisam várias vezes o código HTML das páginas. Para realizar esta análise foi utilizada a API DOM (Document Object Model) do Java.

3.1.2 API DOM

A API DOM fornece uma interface independente de plataforma e linguagem para estruturar o conteúdo de documentos HTML e XML [18]. Desde que o documento esteja bem formado, ele descreve uma linguagem neutra capaz de representar qualquer documento HTML ou XML em forma de uma árvore e tratar a informação armazenada nesses documentos como um modelo de objetos hierárquicos. Ou seja, o DOM possibilita estruturar as tags do código HTML das páginas Web que o sistema irá analisar, facilitando assim este processo. A figura 6 contém uma demonstração de como o DOM realiza esta estruturação para o exemplo em HTML ilustrado na mesma figura.

Uma vez que as tags do código HTML estão mapeadas no objeto do tipo Document da API DOM, através de uma simples chamada de método, é possível efetuar buscas de tag, filtrar uma determinada tag, ou um conjunto destas, excluir tags ou adiciona-las.

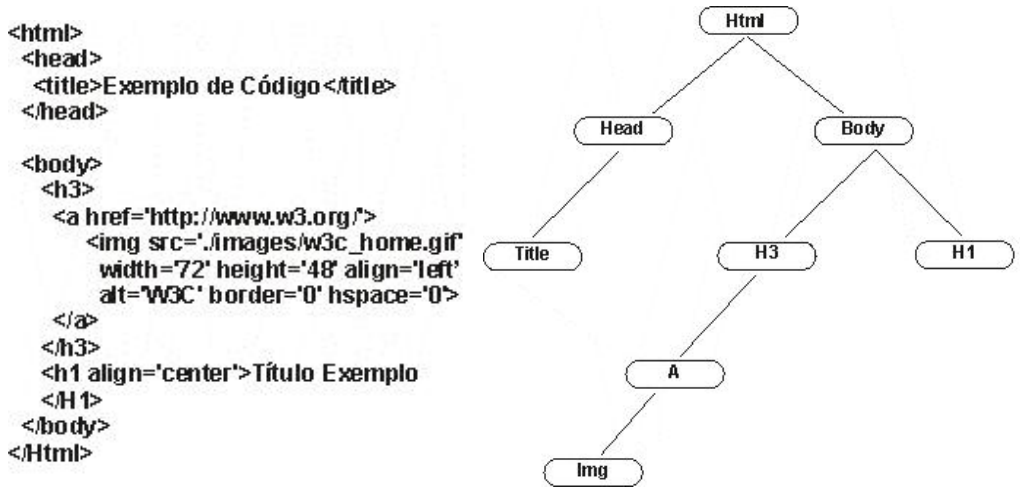

Figura 6: Mapeamento de tags de código HTML feito pelo DOM.

Contudo, para que ocorra este mapeando do DOM é preciso que o documento HTML esteja totalmente correto, uma simples tag sem fechamento, ou uma tag desconhecida pelo DOM, impede todo o mapeamento. Erros como estes são bastante freqüentes nas páginas Web, até porque os navegadores mais recentes são capazes de abstrair tais erros. Assim, antes de utilizar o DOM é preciso corrigir possíveis erros no código que será mapeado. Para isto foi utilizado o Tidy.

3.1.3 Tidy

O Tidy é uma ferramenta que corrige automaticamente os erros mais comuns que ocorrem com bastante freqüência na Web, como tags não fechadas, tags de fechamento na ordem errada, tags de fechamento sem o caractere "/" ou atributos sem aspas [19]. A figura 7 mostra um exemplo de correção efetuada pelo Tidy.

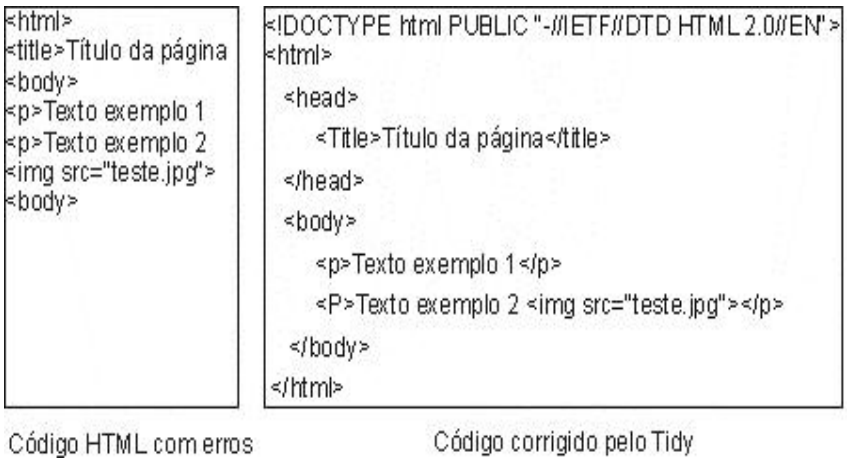

Figura 7: Exemplo de correção automática efetuada pelo Tidy.

Na correção do exemplo acima o Tidy inseriu o DOCTYPE correto no inicio, colocou a tag <head> que faltava, fechou as tags <p>, <html>, e corrigiu o fechamento da tag <body>. Neste processo de correção o Tidy também gerou um aviso alertando que na tag <img> está faltando o atributo "alt", o que não é um erro, mas é muito recomendado que todo "img" tenha um "alt".

3.2 Estrutura Geral do Sistema

O primeiro passo para extrair as informações da tabela de classificação do campeonato brasileiro de futebol é localizar a tabela a partir do endereço eletrônico passado como entrada do sistema. Para isto, analisando diversos portais e sites esportivos, foram definidas seis expressões (links) que podem dar acesso a páginas que contenham a tabela de classificação, estas expressões são: esporte, futebol, brasileirão, tabela, classificação e campeonato. Assim, o sistema deverá procurar na página de entrada a tabela de classificação, caso não seja encontrada busca-se pelas expressões listadas, caso seja identificada alguma das expressões associada a um hiperlink, o mesmo processo é realizado nesta nova página encontrada. Este processo é interrompido quando a tabela é encontrada ou quando todas as páginas já foram analisadas. Para executar a busca da tabela ou dos links é necessário, primeiramente, mapear o código da página utilizando a API DOM. Contudo, foram identificadas tags que, apesar de bem estruturadas, o DOM não reconhece, e para realizar o mapeamento com o DOM não pode existir erro algum. A fim de retirar estas tags do código das páginas foi criada uma base de regras de produção, implementada com o JEOPS. Depois da execução das regras desta base, os possíveis erros de sintaxe existentes no código são corrigir utilizando o Tidy.

A figura 8 mostra um diagrama de seqüência resumindo todos os passos do sistema.

3.3 Implementação das Regras de Produção

Para cada tag não reconhecida pelo DOM foi criada uma regra que analisa o código da página como uma String, a fim de buscar a tag e excluí-la, este procedimento é repetido até que a tag não seja mais encontrada no site. Essas regras compõem a base de filtragem. A figura 9 contém a regra implementada com o JEOPS, que foi utilizada para excluir uma destas tags.

A regra na Figura 9 possui na seção declarations uma instância da classe Site, que corresponde ao site que está sendo analisado. A condição desta regra é determinada pelo método LocalizaTermo(String termo) que busca no código do site a tag passada como parâmetro, caso esta seja encontrada o método retorna sua posição no código. Caso a condição da regra seja atendida, ou seja, a tag procurada seja localizada, os comandos da seção actions são executados. Estes capturam a posição inicial e final da tag identificada e a exclui do código utilizando o método ApagarCodigo(int indice_inicial, int indice_final). Também nesta seção, o comando modified(Object o) é executado para informar ao JEOPS que o objeto s, ou seja, o site em análise, foi modificado e portanto todas as regras desta base devem ser aplicadas a este objeto novamente, isto garante que todas as tags buscadas serão excluídas, mesmo se estas aparecem inúmeras vezes no código. Neste exemplo o JEOPS somente irá parar de executar esta regra quando não for mais encontrada a tag <script>, pois neste caso os comandos da seção actions não serão executados e portando o comando modified não será executado.

Na base de filtragem também foi criada uma regra para redirecionar o site. Comumente ao se colocar um endereço num browser, este é redirecionado para um outro endereço, isto é feito através da tag <script> que possui comandos em Java Script, por exemplo, para redirecionar site. Esta regra consiste em buscar estes comandos e caso encontre, modificar o endereço do site em analise e recarregar seu código.

Uma vez que o código da página já foi corrigido, filtrado e mapeado usando a API DOM, resta então localizar a tabela de classificação. Para isto foi implementada uma segunda base de regras, a base de busca, para localizar uma tabela de classificação e para buscar os links relacionados. A figura 10 contém a regra para identificar uma tabela de classificação.

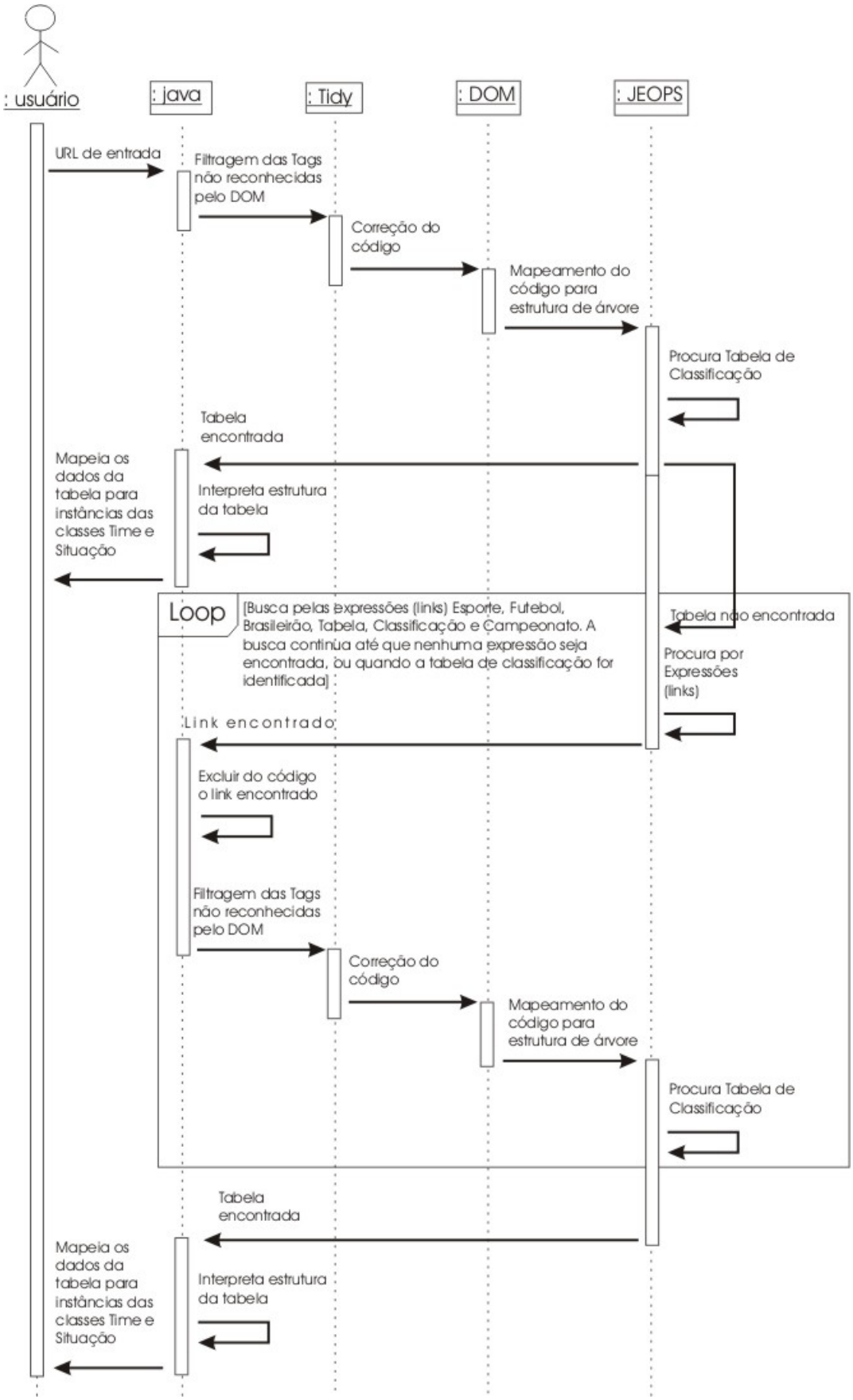

Figura 8: Diagrama de Seqüência do funcionamento do sistema.

.

Figura 9: Regra construída no JEOPS para retirar a tag <script>.

```
rule LocalizarTabela{
        declarations
                Site s;
                HistoricoSites historicoSites;
        conditions
                (!historicoSites.getAchouTabela()) &
                (s.AchouTabela(historicoSites));
        actions
                historicoSites.setAchouTabela();
                historicoSites.setSiteComTabela(s);
                modified(historicoSites);
}
```

Figura 10: Regra implementado no JEOPS para localizar uma tabela de classificação.

Nesta regra é declarado o site que está sendo analisado e um objeto da classe HistóricoSites, que possui informações do processo de busca como um todo. As condições desta regra são que a tabela de classificação ainda não tenha sido encontrada, esta informação estará contida no objeto do tipo HistoricoSites, e que a tabela de classificação seja localizada através da execução do método AchouTabela(). Este método utiliza-se do mapeamento feito com o DOM para recuperar todas as tags do tipo <table> existentes no código da página em análise, as quais possivelmente poderão conter uma tabela de classificação. Para cada tag desta, busca-se pelos possíveis títulos de uma tabela de classificação: times, posição, vitórias, etc., caso sejam identificados cinco destes títulos, este já é um forte indício de que se trata de uma tabela de classificação, assim o método AchouTabela() retorna verdadeiro. Para identificar os títulos da tabela estes são comparados com um arquivo que contém uma lista consideravelmente extensa de possíveis títulos. Por exemplo, neste arquivo contém todas as variações para “time”, como: time, Time, TIME, times, Times, Equipes, equipes etc. Estes possíveis títulos foram definidos a partir de uma análise exaustiva dos principais sites esportivos brasileiros. Uma vez que a tabela é identificada o método setAchouTabela() modifica um flag no objeto historicoSites, informando que a tabela foi encontrada. Isto é feito para que o processo de busca da tabela seja parado, pois todas as regras desta base possuem a condição de que a tabela de classificação ainda não tenha sido identificada. Assim nenhuma regra de busca será executada. Em seguida é registrado em qual site foi encontrada a tabela, e por fim o comando modified(historicoSite) é executado. Trata-se de um comando específico do JEOPS para informar que a instância historicoSite foi modificada, sendo assim todas as regras da base serão executadas novamente com esta instância modificada.

LocalizarTabela é a primeira regra da base de busca, caso a tabela não seja encontrada o JEOPS irá executar as demais regras, as quais buscam pelos links. Uma regra para buscar um link possui na sua declaração o site em análise, o histórico dos sites e um objeto do tipo FiltrarSite. Esta última é a classe que implementa as regras definidas na primeira base, a base de filtragem, utilizada para eliminar as tags não reconhecidas pelo DOM. Este objeto será utilizado para filtrar o código do site associado ao link encontrado. As condições desta regra são:

- A tabela de classificação ainda não tenha sido encontrada.
- O link buscado seja encontrado no código em análise.
- O link encontrado ainda não tenha sido encontrado, ou seja, esta é a primeira vez que este link está sendo identificado.

Uma vez que estas condições sejam satisfeitas as seguintes ações são tomadas:

- O método ExcluirLink será executado para excluir o link que foi encontrado, para que o sistema não entre em loop e permanece sempre encontrando um mesmo link.

- Após a exclusão é criada uma nova instância da Classe Site do site encontrado.
- O código do novo site será filtrado com o objeto do tipo FiltrarSite, para retirar as tags que a API DOM não reconhece.
- É executado o método GerarXHTML, que efetuará a correção do código em análise utilizando o Tidy e irá gerar um objeto do tipo Document, da API DOM, com o mapeamento do código.
- Este novo site é incluído no histórico dos sites, pra que não seja mais analisado.
- Finalmente cria-se uma nova instância da classe BaseBuscar que é a classe que implementa as regras definidas na base de busca, que será executada para o novo site encontrado. Ou seja, as mesmas regras de busca serão executadas, de forma recursiva, para o novo site. Será buscada a tabela de classificação e caso não encontre serão buscados os links relacionados.

A figura 11 contém a implementação de uma destas regras utilizadas para buscar um link.

Figura 11: Regra para buscar o link esporte.

A seção action da regra descrita acima contém os métodos tell e run, estes são métodos do JEOPS. O tell é usando para declarar os objetos que serão utilizados pela base de regras, já o método run é chamado para inicializar a execução das regras. Caso uma regra desta base possua na sua seção “declarations” um ou mais destes objetos declarados com o tell, esta regra será executada.

Nesta regra a busca do link é efetuada pelo método SearchLink, este recebe como parâmetro um vetor de strings contendo todos os sinônimos de um link, como por exemplo um vetor com as strings: Esporte, esporte, ESPORTE, Esportes, esportes, etc., ou seja, todas as variação que o termo esporte pode ter. O SearchLink percorre cada nó do código buscando uma tag que possua um atributo href, o qual referencia outros sites. Quando este é identificado verifica-se o texto associado à tag, ou seja, a descrição do link. Este texto é comparado com os termos do vetor de strings passado como parâmetro. Caso o texto coincida com um dos termos do vetor o valor do atributo href, que é o endereço associado ao link, é retornado.

Na base de busca foi criada também uma regra para localizar possíveis frames existentes na página. Quando são identificados o sistema recupera o endereço associado a este e cria uma nova instância de Site, onde será efetuado todo o processo de busca.

3.4 Parâmetro de Sensibilidade

No método SearchLink foi implementado o parâmetro de sensibilidade de busca. Um parâmetro para definir o tamanho dos links que serão analisados. Por exemplo, ao se buscar o link “futebol”, pode-se procurar um link exatamente com esta descrição: “futebol”, ou então buscar um link que contenha esta descrição. A vantagem que existe em buscar a palavra exata é o fato de que existirá uma certeza que aquele link está associado a uma página que contém informações sobre futebol. Caso seja buscado um link que contenha a palavra futebol, pode-se encontrar um link com uma descrição longa, como uma notícia com uma descrição do tipo: “…o jogador de futebol fez três gols…”, neste caso a palavra futebol seria identificada e este link seria analisado, contudo provavelmente este link não irá dar acesso à tabela de classificação, mas a uma reportagem sobre o jogador de futebol. Por outro lado pode-se encontrar o link “Campeonato de Futebol Brasileiro”, este apesar de não coincidir exatamente com a palavra buscada, é um link que vale à pena ser analisado em busca da tabela de classificação. Sendo assim foi definido como parâmetro de sensibilidade de busca o número de caracteres a mais que a palavra buscada, ou seja, se o parâmetro de sensibilidade é 20 e busca-se pelo termo esporte, somente serão analisados os links que contenham a palavra esporte e não possuam mais que 20 caracteres. Neste caso, por exemplo, o link “Tudo sobre esporte” seria analisado, mas o link “…o governo do estado tem investido bastante no esporte…” não seria, pois, apesar de conter a palavra esporte, possui mais que 20 caracteres. Assim, mantendo o parâmetro de sensibilidade baixo o sistema irá analisar sites que possuem uma maior

probabilidade de possuir a tabela de classificação, mas corre o risco de não analisar algumas páginas relevantes. Por outro lado, aumentando demais o parâmetro de sensibilidade o sistema poderá perder muito tempo analisando sites que não têm nada haver com o conteúdo buscado, por exemplo o link “Campeonato de Basquete” poderia ser acessado na busca de um link que contivesse o termo “campeonato”, esta certamente seria uma página analisada sem necessidade.

3.5 Mapeamento das Informações em Instâncias de Classes

Uma vez que a tabela de classificação foi identificada resta mapear as informações extraídas em instâncias das classes Time e Situação. Isto é feito através do método MapearDados que recebe como parâmetro a tag <table> que contém a tabela de classificação. O método irá percorrer os títulos das colunas da tabela para identificar a ordem em que os campos estão dispostos. Uma vez que esta ordem foi identificada pode-se prever em qual posição da tabela as informações estarão dispostas. Assim, para cada linha da tabela é criada uma instância da classe time associando ao nome do time da linha corrente e será criada também uma instância da classe Situação que possuirá as informações deste time.

4. RESULTADOS

A aplicação foi testada com sucesso em diversos sites. A figura 12 mostra um esquema da busca realizada pelo sistema passando como entrada o endereço do portal do Terra. Para o parâmetro de sensibilidade foi adotado o valor 10 como padrão, pois equivale, aproximadamente, a uma palavra a mais que o link procurado. Por exemplo, o parâmetro 10 permite capturar o link “Seção de Esportes”, mas não o link “Entrevista com o ministro do esporte”.

Nesta busca o sistema analisou seis páginas prováveis de encontrar a tabela de classificação. Na página 1, que foi o endereço inicial passado para o sistema, foi identificado o link Esportes que aponta para a página 2. Nesta foram encontrados os links Esportes Show e Brasileiro 2007, este último está associado à página 4, na qual foram encontrados os links Futebol e Classificação. O link Classificação deu acesso à página 6 onde foi identificada a tabela de classificação. Nas páginas 3 e 5 o sistema não identificou nenhum novo link.

Figura 12: Busca realizada no portal do Terra.

Esta mesma busca foi executada novamente, porém, o valor adotado para o parâmetro de sensibilidade foi 5. Nesta execução o sistema ignorou o link “Esporte Show”, que realmente foi um link desnecessário para localizar a tabela de classificação.

Para análise do parâmetro de sensibilidade, foi feito uma variação na busca realizada no site da UOL. Inicialmente foi realizada a busca no portal da UOL com parâmetro de sensibilidade 10. A figura 13 contém um esquema desta busca.

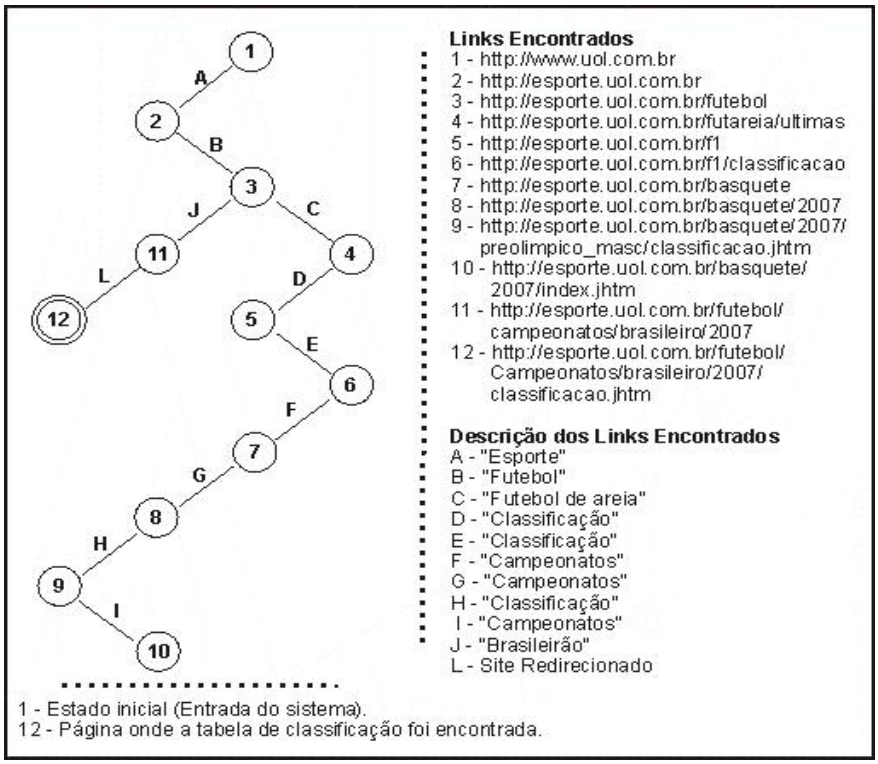

Figura 13:. Busca realizada no portal da UOL com sensibilidade 10.

Pode-se notar que na página 3 foi encontrado o link "Futebol de areia", como existem apenas nove caracteres a mais que a palavra buscada, que neste caso era Futebol, e a sensibilidade é 10, o sistema admitiu como sendo um link provável para encontrar a tabela de classificação. Contudo, apesar da tabela ter sido encontrada no final, a identificação deste link levou a uma análise de outras páginas irrelevantes como a página de classificação da fórmula 1 ou a página de classificação do campeonato de basquete. Esta mesma busca foi realizada com a sensibilidade igual a seis. A figura 14 contém o esquema deste resultado.

Reduzindo o parâmetro de sensibilidade o sistema apenas analisou cinco páginas, ao invés de 12, quando a sensibilidade era 10, e encontrou a mesma tabela de classificação. Pode-se notar que todas as páginas acessadas realmente referem-se ao estudo de caso do Campeonato Brasileiro de Futebol.

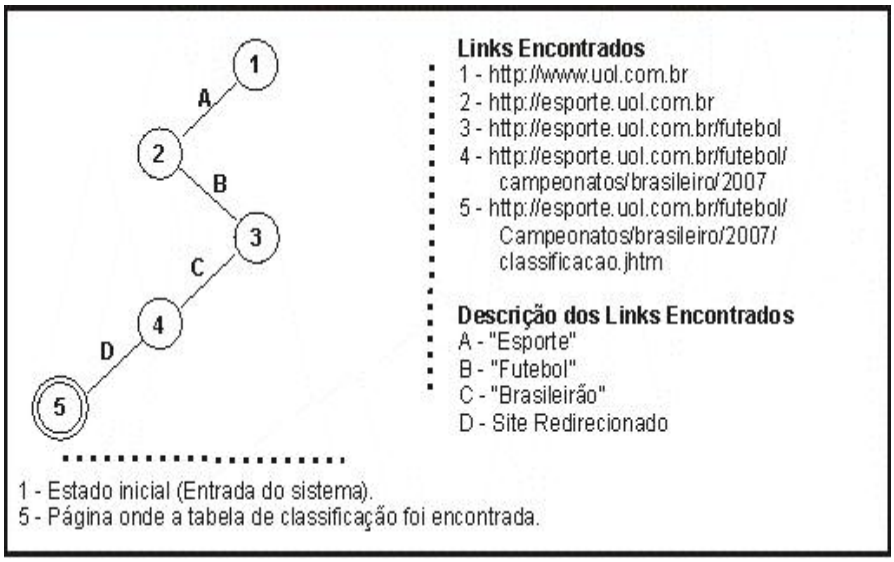

Figura 14: Busca realizada no portal da UOL com sensibilidade 6.

Apesar de o sistema ter funcionado bem em diversos sites, em alguns dos sites testados não foi possível realizar o processo de extração devido a limitação das regras implementadas, como o site Folha On-line (http://www.folha.uol.com.br/) que construiu sua tabela de classificação usando Java Script, ao invés da tag <table>. Neste caso, o sistema indica que não encontrou nenhuma tabela, pois não foi identificada nenhuma estrutura do tipo <table>. Problemas como este são resultado da não padronização dos sites, um dos principais problemas dos sistemas de extração de informação na Web. Por mais que sejam criadas regras para extração, sempre podem surgir casos que estas regras não resolvam.

5. APLICAÇÕES

Existem várias aplicações para o sistema construído neste trabalho ou sistemas do mesmo gênero. Aqui destacamos duas dessas aplicações.

São populares os sistemas para acompanhar as séries A e B do campeonato brasileiro de futebol. Nestes é necessário que o usuário informe ao sistema os resultados dos jogos para obter a classificação das equipes e gráficos de desempenho. Esta atualização pode ser feita diretamente por um palm. Com uma pequena modificação no sistema de extração de informação descrito neste trabalho, para recuperar a tabela com os resultados dos jogos, ao invés da tabela de classificação, estes sistemas populares poderiam ser alimentados automaticamente, bastando apenas que estivesse conectado à Internet.

Uma segunda aplicação é a utilização da saída do sistema em questão como entrada de um sistema de geração automática de textos. O trabalho descrito em [12], por exemplo, propõe uma nova abordagem de desenvolvimento que considera linguagens e padrões da engenharia de software moderna para impulsionar o uso prático de Sistemas de geração de linguagem natural (GLN). A proposta considera o pipeline de atividades característico de um processo de geração de textos em linguagem natural como um conjunto de bases de regras de transformação de modelos. O trabalho em questão considera a construção de um gerador de relatórios de jogos de campeonatos de futebol. Os dados de entrada do sistema foram obtidos através de um simulador que gera resultados de partidas do campeonato brasileiro de futebol. Para originar a entrada do gerador de relatórios, o simulador cria aleatoriamente instâncias de classes e mapeia os valores dessas instâncias para o formato XMI.

O trabalho descrito em [5] mostra os resultados obtidos com o processo inicial de geração de textos a partir do *wrapper* descrito neste artigo. O trabalho em questão descreve a determinação do conteúdo do texto final em linguagem natural substituindo o simulador utilizado em [12] pelo *wrapper* implementado e descrito neste trabalho. A figura 15 ilustra a adaptação feita nesse processo.

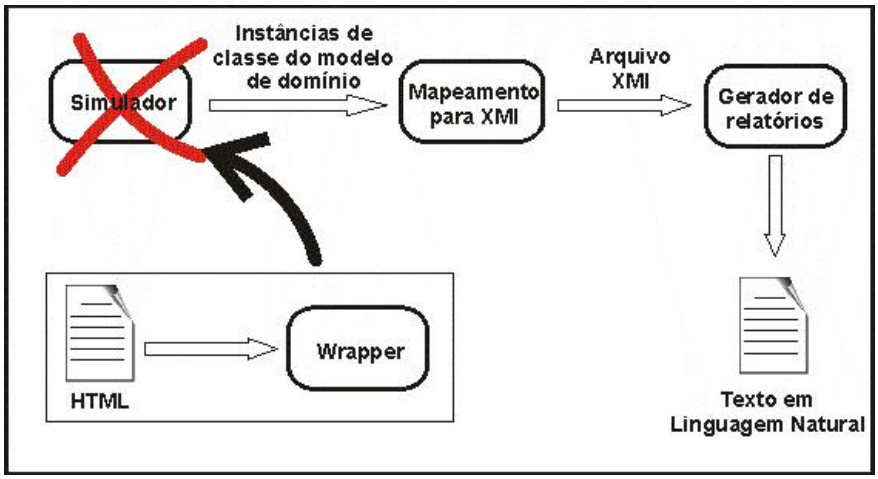

Figura 15: Wrapper como entrada para um sistema de geração de linguagem natural.

6. CONCLUSÃO

Este trabalho teve como finalidade analisar e aplicar técnicas de extração de informação a fim de estruturar automaticamente as informações extraídas. Assim, foram apresentados os principais conceitos dos sistemas de extração de informação e as principais técnicas utilizadas nestes. Como estudo de caso foi implementado um sistema de extração de informação para recuperar da Web as informações do atual campeonato brasileiro de futebol, a partir de uma URL de entrada. O sistema identifica links relacionados com o domínio e realiza a busca pela tabela de classificação do campeonato em cada uma das páginas localizadas. Uma vez que a tabela é encontrada, é realizado o mapeamento das informações em instâncias de classes estruturadas. O sistema foi construído utilizando regras de produção, que foram implementadas com o JEOPS, um motor de inferência de primeira ordem com encadeamento progressivo integrado à linguagem Java. O sistema foi testado em diversos portais e sites esportivos, como no portal do Terra (www.terra.com.br), no site do jornal O Povo (www.opovo.com.br) e no portal da UOL (www.uol.com.br). Nestes sites e em outros, o wrapper implementado identificou a tabela de classificação, e estruturou as informações com sucesso. O sistema apresentou resultado negativo em apenas alguns sites devido a não padronização das páginas na Web, um problema típico dos sistemas em geral de extração de informação na Web.

Apesar da limitação citada, o sistema atingiu seu objetivo que era de localizar e extrair os dados da tabela de classificação do campeonato brasileiro. Através de um conjunto de regras de produção, o sistema foi capaz de prever diversas variações que possam existir no código das páginas HTML.

Como trabalhos futuros estão previstos alguns aprimoramentos para o sistema implementado, como a captura de tabelas de classificação que não estejam numa tag <table>, além da captura das demais informações do campeonato brasileiro, como a tabela de resultados dos jogos. É importante notar que a implementação descrita aqui é específica para extrair informações das tabelas de classificação do campeonato brasileiro. Contudo, esta pode ser facilmente modificada para casos de uso semelhantes, como a extração de informações de qualquer esporte que possua tabela de classificação, como basquete, vôlei, Fórmula 1 e muitos outros. Para adaptar o sistema a outros domínios, basta redefinir as expressões (links) a serem buscadas. Para recuperar outras informações além da tabela de classificação, basta acrescentar novas regras no JEOPS. Além disso, como trabalho futuro, pretende-se implementar o mesmo sistema com outras técnicas de extração de informação e realizar uma comparação entre os trabalhos. Por fim, pretende-se também incorporar técnicas de aprendizagem ao sistema, possibilitando assim que o sistema seja capaz aumentar sua base de conhecimento de forma automática.

---

1. BORKAR, V. R.; DESHMUKH, K.; SARAWAGI, S. Automatic segmentation of text into structured records, Proceedings of the ACM-SIGMOD Int'l Conference on Management of Data, p. 175-186, 2001.
2. CALIFF, M.E.; MOONEY, R.J. Relational learning of pattern-match rules for information extraction, Proceedings of the 16th National Conf. on AI, 1999.
3. EIKVIL, LINE. Information Extraction from World Wide Web - A Survey. Norwegian Computing Center, 1999.
4. FILHO, CARLOS S. F. JEOPS – Integração entre Objetos e Regras de Produção. 2000. 130 f. Dissertação de Mestrado em Ciência da Computação, Universidade Federal de Pernambuco, Pernambuco, 2000.
5. FONSECA, M.; JUNIOR, L.; MELO, A.; MACEDO, H. T. Innovative Approach for Engineering NLG Systems: The Content Determination Case Study. Lecture Notes in Computer Science, Springer-Verlag, v. 4919, p. 478-487, 2008.
6. FREITAG, D. Information extraction from HTML: Application of general machine learning approach. In Proc. of the AAAI, 1998.
7. HSU, C.; DUNG, M. Generating finite-state transducers for semi-structured data extraction from the Web, Journal of Information Systems, Vol. 23, p. 521–538, 1998.
8. KOSALA, R.J.; DEN BUSSCHE. V.; BRUYNOOGHE, M.; BLOCKEEL, H. Information extraction in structured documents using tree automata induction, Proc of the 6th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD), p. 299-310, 2002.

9. KUSHMERICK, N.; WELD, D.S.; DOORENBOS, R. Wrapper induction for information extraction, Proc. of the 15th Int'l Joint Conference on AI, 1997.

10. KUSHMERICK, N.; JOHNSTON, E.; MCGUINNESS, S. Information extraction by text classification, IJCAI Workshop on Adaptive Text Extraction and Mining, Seattle, WA, 2001.
11. MARQUES, TIAGO C.; PINHEIRO, VLÁDIA CÉLIA E FURTADO, JOÃO JOSÉ V. P. Modelagem de conhecimento integrando regras de produção e ontologias. Encontro de Iniciação à Pesquisa da Unifor, 2004.
12. MELO, A.; MACEDO, H.; FONSECA, M.; JUNIOR, L. Uma Nova Abordagem para Engenharia de Sistemas de Geração de Textos em Linguagem Natural: Aplicação na Determinação do Conteúdo. I Simpósio Brasileiro de Inteligência Computacional, 2007.
13. RAY, S.; CRAVEN, M. Representing sentence structure in hidden markov models for information extraction, In Proc. of the Int. Joint Conf. on Artificial Intelligence, 2001.
14. SODERLAND, S. Learning information extraction rules for semi-structured and free text, Machine Learnig, Vol. 34, p. 233-272, 1999.
15. TIMÓTEO, M. Extraindo informações de referências bibliográficas automaticamente. 2006. 31 f. Trabalho de Conclusão de Curso (Graduação) – Curso de Ciência da Computação, Universidade Federal de Pernambuco, Pernambuco, 2006.
16. BERTOLI, CLAUDIO; CRESCENZI, VALTER; MERIALDO, PAOLO. Crawling programs for wrapper-based applications, Information Reuse and Integration, 2008. IRI 2008. IEEE International Conference, pp.160-165, 13-15, Las Vegas, Nevada, USA July 2008.
17. S. CHAKRABARTI, M. VAN DEN BERG, AND B. DOM. Focused crawling: a new approach to topic-specific Web resource discovery. Computer Networks (Amsterdam, Netherlands), 31(11–16):1623–1640, 1999
18. Package org.w3c.dom. Sun Microsystems, Inc. Acesso em 16 de Agosto de 2009. Disponível em http://java.sun.com/j2se/1.4.2/docs/api/org/w3c/dom/package-summary.html
19. Dave Raggett. Clean up your Web pages with HTML Tidy. W3C® (MIT, ERCIM, Keio). Acesso em 16 de Agosto de 2009. Disponível em http://www.w3.org/People/Raggett/tidy/