UNIVERSIDADE FEDERAL
DE SANTA CATARINA

UM ALGORITMO PARA EXTRAÇÃO DE LISTAS ESTRUTURADAS DA WEB BASEADO EM HEURÍSTICA

Fernando Luís Amorim Agostinho

Florianópolis, SC
2015

UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE CIÊNCIA DA COMPUTAÇÃO

UM ALGORITMO PARA EXTRAÇÃO DE LISTAS ESTRUTURADAS DA WEB BASEADO EM HEURÍSTICA

Fernando Luís Amorim Agostinho

Trabalho de conclusão de
curso apresentado como parte
dos requisitos para obtenção
do grau de Bacharel em
Ciência da Computação.

Orientador:
Ronaldo dos Santos Mello, Dr.

Banca Examinadora:
Profª. Carina Friedrich Dorneles
Prof. Frank Siqueira

Florianópolis, SC
2015

Fernando Luís Amorim Agostinho

UM ALGORITMO PARA EXTRAÇÃO DE LISTAS ESTRUTURADAS DA WEB BASEADO EM HEURÍSTICA

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Ronaldo dos Santos Mello, Dr.

Banca Examinadora

Profª. Dr. Carina Friedrich Dorneles

Prof. Frank Siqueira

Agradecimentos

Tenho uma imensa gratidão a todos os professores do departamento que, nos anos que por aqui passei, desempenharam um papel primordial na minha formação. Um agradecimento aos membros da banca, Carina e Frank e em especial ao professor Ronaldo, que teve muita paciência e dedicação para orientar este trabalho.

Agradeço a minha mãe, Maria de Fátima Amorim, por todo apoio psicológico para enfrentar essa longa etapa da graduação. Agradeço a Daniele Freitas pelo carinho e contribuição direta para o desenvolvimento desse trabalho.

Resumo

Este trabalho apresenta o SWLE, um mecanismo para descoberta e extração de listas da Web que possuam dados estruturados. A técnica utilizada busca identificar elementos textuais sequenciais que possuam um padrão quanto à presença, quantidade e ordenação de caracteres especiais, julgando que isso seja uma heurística suficiente para concluir que o conteúdo textual entre estes caracteres representam os mesmos atributos, pareados entre os demais registros da listagem.

Palavras chave: Web Lists, dados estruturados, extração de dados

Sumário

1. Introdução

1.1. Visão Geral

Um dado isolado no meio de um contexto não emite grande significado. Ele é apenas um conteúdo de algum atributo de um domínio qualquer. Entretanto, quando ele se junta a outros dados através de uma relação semântica, dizemos que essa junção forma um conjunto de dados estruturados ou uma informação estruturada. Eles podem ser encontrados em textos planos. Um humano pode encontrá-los no decorrer de uma leitura. Porém, eles são mais evidentes em estruturas que, de alguma forma, apresentem estes dados separados por alguma identidade visual, em que facilmente podemos identificar o domínio de conhecimento da entidade a que se trata e seus atributos.

Dados estruturados são a base do conhecimento. Eles estão presentes em toda parte no mundo digital. Quando eles estão vinculados a uma base de dados relacional, eles estão rigidamente organizados sobre uma estrutura semântica previamente formalizada. Por outro lado, grande parte dos dados digitais está localizada em ambientes em que a estrutura semântica não possui essa estrutura formal, ou muitas vezes apresentam uma semi-estruturação ou mesmo ausência de estruturação, como é o caso da Web. Os elementos das páginas da Web são focados primordialmente para a leitura por humanos. A Web é o maior repositório de dados hoje em dia e nela pode-se encontrar muita informação estruturada. Porém, a web ainda não tem um contexto semântico, apesar de existirem muitos esforços para que isso ocorra (BERNERS-LEE, 2009).

As principais ferramentas de busca da web hoje funcionam através de palavras-chave, previamente indexadas de acordo com a importância dessa palavra num determinado contexto. Entretanto, essa forma de busca é muito imprecisa. Uma determinada palavra-chave pode conter inúmeros significados, totalmente distintos entre si, porém o usuário geralmente procura somente por algum significado específico. Ao procurar por “pacífico”, ele está querendo

saber sobre o oceano ou sobre uma situação de paz? Isso obriga o usuário a ter que reformular a sua busca até encontrar o que ele realmente deseja.

PLATINUM é um projeto de pesquisa com o objetivo de tratar essa situação (MELLO, 2012). Ele propõe uma plataforma para a busca integrada de dados da Web e da nuvem. Essa busca integrada seria realizada sobre o mar de informações existentes nesses ambientes através de uma linguagem similar a SQL, como se estivéssemos acessando apenas uma única base de dados. Como exemplo, poderíamos formular buscas do tipo:

```
SELECT empresa, linha, horário_de_saída
FROM ônibus
WHERE origem='Florianópolis' AND destino='Joinville';
```

Esta consulta exemplo poderia retornar uma lista de diferentes linhas, de diferentes empresas, que partem ou passam por Florianópolis com parada ou destino em Joinville.

O desenvolvimento da plataforma precisa levar em conta uma série de questões como a descoberta dos dados, a modelagem da semântica destes dados para que seja possível uma forma de pesquisa eficiente por um sistema informatizado, a combinação dos dados de diferentes fontes e a disponibilização de uma linguagem para o acesso a eles. Cada uma dessas questões têm suas dificuldades e devem ser tratadas separadamente.

A descoberta e extração de dados estruturados na Web é o primeiro desafio para o desenvolvimento da plataforma. Podemos encontrar dados estruturados em páginas HTML no formato de tabelas, listas, formulários, entre outros. O algoritmo, aqui apresentado, busca encontrar e extrair dados a partir de listas, estrutura ainda pouco tratada na literatura.

1.2. Objetivo Geral

O objetivo deste trabalho é o desenvolvimento de um algoritmo para a extração de dados estruturados a partir de listas encontradas em páginas da web. O conteúdo extraído deve ser armazenado em alguma estrutura tabular semelhante a uma tabela relacional, tal que os valores de um mesmo atributo para os registros coletados de uma lista devem se encontrar na mesma coluna.

1.2. Objetivos Específicos

Para que o objetivo geral deste trabalho seja atendido, os seguintes objetivos específicos devem ser considerados:

- Construir o algoritmo de extração
- Construir um crawler para a coleta aleatória de links na Web.
- Fazer testes para avaliar a precisão e revocação do algoritmo criado.

1.3. Metodologia

As principais etapas de desenvolvimento deste trabalho foram:

- Primeiramente foi realizada uma pesquisa bibliográfica para a descoberta de trabalhos correlatos, levantando e analisando as principais técnicas existentes para a coleta e extração de dados estruturados a partir de listas na web.
- Com base nesta análise, foi desenvolvido o cerne deste trabalho, que é um algoritmo de extração de listas na Web.
- Foi construído um crawler para coletar, de forma aleatória, uma grande quantidade endereços da web. Sua execução resgatou cerca de 1.5 milhão de novos links. A partir de um subconjunto destes endereços, foram realizados experimentos para a verificar o desempenho do algoritmo.
- Por fim, a partir da utilização das métricas clássicas de recuperação de informação (precisão e revocação), foi possível avaliar a qualidade do algoritmo proposto.

1.4. Estrutura do Trabalho

1. Fundamentação Teórica: este capítulo apresenta os conceitos de dados estruturados, web estruturada, extração de dados e algumas das etapas do projeto PLATINUM;

2. Trabalhos Relacionados: este capítulo descreve sucintamente os principais trabalhos relacionados com a extração de listas da web;
3. Proposta: este capítulo apresenta em detalhes o algoritmo proposto para extração de listas na Web;
4. Experimentos: este capítulo descreve dois experimentos realizados para a verificação da precisão e revocação a partir de conteúdo extraído da Web através do algoritmo proposto;
5. Conclusão: este capítulo conclui o trabalho realizado, mostrando as principais vantagens do extrator proposto e sugestões para trabalhos futuros.

2. Fundamentação Teórica

2.1 Dados Estruturados

O conceito primordial deste trabalho é a definição do que são dados estruturados. Comumente, dados estruturados estão relacionados a dados similares a uma instância de um banco de dados relacional, em que seus dados possuem uma estrutura rígida e previamente definida através de um esquema, que os categoriza de acordo com seus significados. A forma oposta, os dados não-estruturados, representa os dados que não possuem essa estrutura formal definida. Textos em geral, imagens e vídeos se encaixam nessa classificação. Adicionalmente, existem os dados que possuem uma estrutura irregular e muitas vezes implícita nos próprios dados, como os documentos XML. Estes são conhecidos como dados semi-estruturados.

Michael J. Cafarella (2009) formaliza os conceitos citados acima de uma maneira genérica, dizendo que os termos dados estruturados, semi-estruturados ou não-estruturados descrevem o grau em que tal conjunto de dados oferece suporte a consultas relacionadas a um domínio específico. Em uma base de dados relacional, por exemplo, o formalismo da sua estrutura nos permite realizar pesquisas do tipo “SELECT nome FROM funcionários WHERE salario > 2000”, onde nome, funcionários e salário são conceitos de um domínio específico.

A importância dos dados estruturados está na própria definição sugerida por Michael J. Cafarella (2009), ou seja, a partir de um conjunto de dados com a estrutura semântica bem definida, conseguimos utilizá-los de uma maneira mais precisa através de consultas específicas.

no.[2]	Name	Starting year of Office	Ending year of Office
1	Thomas Willett	1665	1666
2	Thomas Delavall	1666	1667
3	Thomas Willett	1667	1668
4	Cornelius Van Steenwyk	1668	1671
5	Thomas Delavall	1671	1672
⋮			
56	William Paulding, Jr.	1825	1826
57	Philip Hone	1826	1827
58	William Paulding Jr.	1827	1829
59	Walter Bowne	1829	1832
60	Gideon Lee	1833	1834

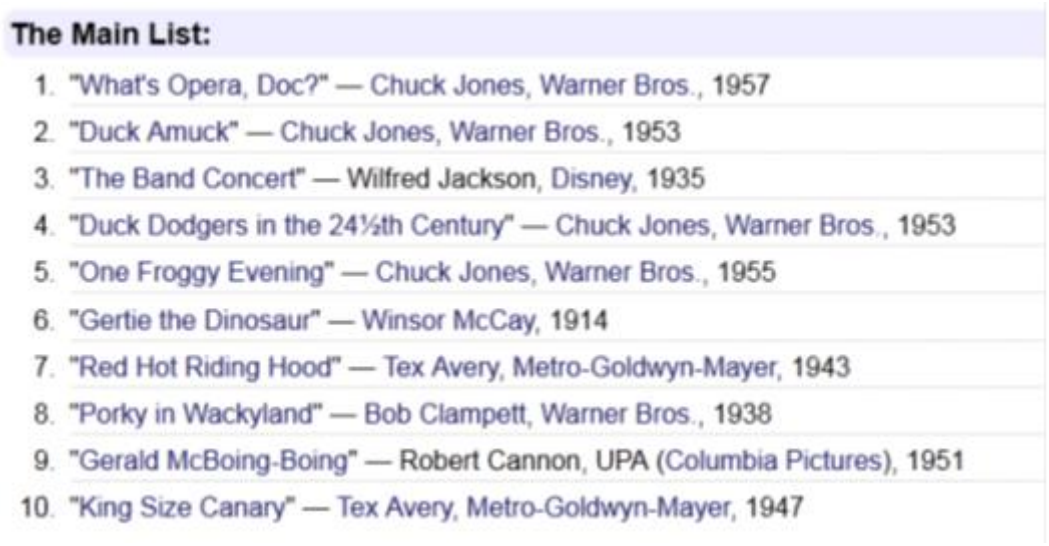
Figura 1: Lista dos 60 primeiros prefeitos de Nova York

A Figura 1 mostra uma tabela retirada da Wikipédia. Ela lista os 60 primeiros prefeitos da cidade de Nova York. Essa tabela é uma boa representação de informação estruturada. Visualmente, podemos rapidamente estabelecer relações entre os elementos de uma mesma linha e colunas da tabela. Ainda, podemos elaborar uma consulta relacionada com o domínio apresentado, como por exemplo, “quais os prefeitos que governaram a cidade de Nova York no período entre 1825 e 1832?”, justificando sua representação como informação estruturada.

Dados estruturados no formato de tabelas são muito comuns em bancos de dados. Porém, como visto no exemplo anterior, elas podem também ser encontradas na web. Michael J. Cafarella (2009) estimou em 2009 que, apenas na língua inglesa, existem cerca de 154 milhões de tabelas estruturadas na Web. Este número deve ser muito superior nos dias de hoje.

Entretanto, tabelas não são as únicas fontes de dados estruturados na Web. Dados estruturados podem ser apresentados em outros formatos como listas e formulários. Muitas vezes uma tabela apresenta dados em um formato mais próximo a um esquema relacional de banco de dados, com os nomes de seus atributos na parte superior da tabela e cada linha sendo representada por um registro da base de dados. Porém, em outros casos, como o exemplo mostrado na Figura 2, podemos verificar a presença de informação estruturada

mesmo sem conseguirmos identificar, em uma primeira análise, o domínio em questão e os atributos de interesse.



The Main List:

1. "What's Opera, Doc?" — Chuck Jones, Warner Bros., 1957
2. "Duck Amuck" — Chuck Jones, Warner Bros., 1953
3. "The Band Concert" — Wilfred Jackson, Disney, 1935
4. "Duck Dodgers in the 24½th Century" — Chuck Jones, Warner Bros., 1953
5. "One Froggy Evening" — Chuck Jones, Warner Bros., 1955
6. "Gertie the Dinosaur" — Winsor McCay, 1914
7. "Red Hot Riding Hood" — Tex Avery, Metro-Goldwyn-Mayer, 1943
8. "Porky in Wackyland" — Bob Clampett, Warner Bros., 1938
9. "Gerald McBoing-Boing" — Robert Cannon, UPA (Columbia Pictures), 1951
10. "King Size Canary" — Tex Avery, Metro-Goldwyn-Mayer, 1947

Figura 2: Lista dos 10 maiores desenhos animados.

O exemplo da Figura 2 mostra uma lista de desenhos animados produzidos. Após uma breve leitura deste exemplo, podemos perceber que existe um padrão entre seus itens. De maneira intuitiva, podemos descrever o padrão dos itens como sendo: [Número. Título – Nome, Produtor, Ano]. Um leigo no assunto ou na língua utilizada poderia ter dificuldades em extrair essa informação da lista por não ter uma descrição mais detalhada como uma introdução, ou atributos explícitos, como temos nos cabeçalhos das tabelas.

A separação de cada possível valor de atributo em um item da lista é outro ponto que não visualizamos de forma explícita. Precisamos olhar diversos itens para conseguirmos identificar alguns separadores comuns, tais como traços, vírgulas, pontos, etc. Em alguns casos, um mesmo caractere utilizado como separador pode fazer parte do texto em outro local da mesma lista.

Elmeleegy (2011) estima que a quantidade de listas com informação estruturada é semelhante à quantidade de informações estruturadas representadas através de tabelas.

Os dois recursos da web para representação de dados estruturados comentados até o momento, tabelas e listas, são organizados de acordo com a semântica do domínio a que se propõem. Porém, essa estrutura de

organização é apenas um modelo visual. O código HTML por trás desta organização não traz nenhuma especificação formal do domínio do assunto apresentado. Para um computador acessar uma base de dados em busca de respostas para consultas de uma forma eficiente, é necessário que o modelo de organização semântica desses dados seja expresso numa forma previamente definida, como é o caso da organização dos dados em tabelas relacionais com um esquema. A ausência dessa esquematização na Web faz com que um mecanismo de busca neste ambiente com inferência na estrutura e da sua semântica seja muito complexo.

Como solução parcial para esse problema, ou seja, para tratar a questão da inferência de estrutura, poderíamos criar um esquema para cada elemento da web que possua informação estruturada e adequar todos os seus dados a ele, podendo inclusive realizar esse processo com o objetivo de ter como produto final uma base de dados relacional. Essa tarefa seria feita em um momento anterior à execução de consultas, e apenas uma vez.

Esta proposta de solução é o foco deste trabalho. Para cada tipo de elemento, sejam tabelas, listas ou formulários, existem diferentes desafios associados à descoberta da organização dos seus dados em busca da definição de uma estrutura padrão e homogênea, como é o caso de bases de dados relacionais. Esse trabalho define um mecanismo que realiza essa tarefa para listas encontradas na web.

2.2. Projeto PLATINUM

O projeto PLATINUM visa desenvolver uma plataforma para a busca integrada a dados presentes em fontes da web e na nuvem (MELLO, 2012). A figura 3 mostra a arquitetura de alto nível do projeto.

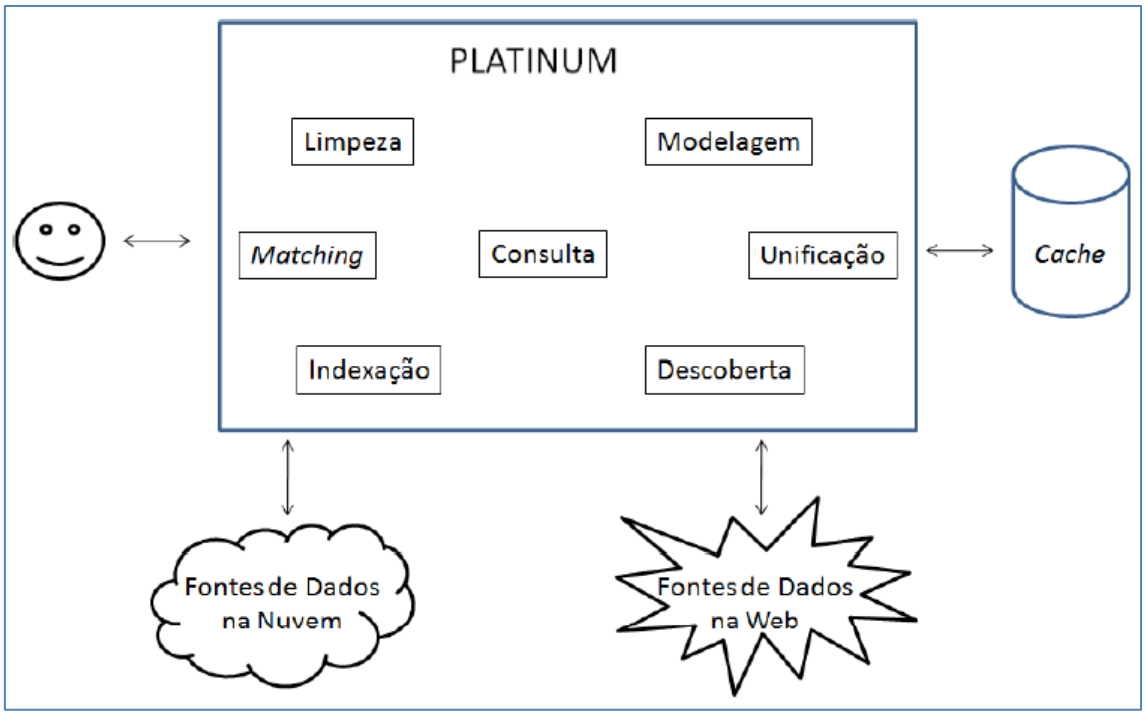


Figura 3: Arquitetura de alto nível do projeto PLATINUM

Os principais módulos do PLATINUM são os seguintes:

- **Descoberta**: Este primeiro item corresponde à fase da busca de elementos (tabelas, lista, formulários) com informação estruturada contidos na web. Essas informações, como explicado anteriormente, estão no formato de um documento HTML. A estrutura semântica dos dados está implícita na estrutura visual que apresenta o elemento. Um *crawler* faz o papel de buscar, de forma contínua, inúmeras páginas pela web afora. Cada uma dessas páginas é analisada em busca de prováveis elementos com informação estruturada, que são resgatados caso estes elementos apresentem uma estrutura semântica implícita ou um algum padrão de representação recorrente, que pode dar indícios de uma coleção de dados relevantes.
- **Limpeza**: Eliminam-se ou alteram-se algumas informações dos elementos, já resgatados, que não são relevantes para o domínio em questão. É comum que dados presentes na web sejam redigidos com alguma ortografia incorreta, ou mesmo atributos sem muito significado. No meio de uma tabela, por exemplo, podem

haver atributos que estejam redigidos de uma forma resumida, ou com erros. Podem também existir tags, como algum campo oculto, no meio das estruturas para guardar informações que facilitam no processo de dinamização de uma página da internet.

- **Modelagem**: Essa parte é responsável por criar a estrutura semântica para os dados estruturados que no momento estão representados de acordo com a estrutura visual do código HTML. As principais dificuldades estão relacionadas com as diversas formas possíveis que um elemento com informação estruturada pode assumir em nível de domínio, tipo de elemento e itens de dados.
- **Matching e Unificação**: Matching é o processo de mesclagem dos diversos esquemas de dados extraídos da web. Unificação é a adequação dos dados dos outros modelos para um modelo em comum.

Este trabalho contribui para a etapa de descoberta de dados do projeto PLATINUM, em especial, a extração de dados estruturados em listas presentes na Web. O capítulo 4 detalha o algoritmo proposto para tal tarefa.

3. Trabalhos Relacionados

Este capítulo apresenta alguns artigos na literatura relacionados à extração de dados presentes em listas estruturadas na Web.

3.1. ListExtract, Halevy(2011)

A técnica apresentada neste trabalho recebe como fonte de entrada uma lista de registros, e como saída uma tabela com estes registros separados e realocados nas colunas que representam os respectivos atributos extraídos (Halevy et al, 2011).

A técnica conta com as fases de Separação, Alinhamento e Refinamento. Na fase de Separação, o ponto chave da solução utilizada é a função FQ, que utiliza uma base de conhecimento para identificar corretamente os atributos. Essa função é aplicada sobre todos os arranjos de palavras em sequência que um registro pode ter. Se um registro possui as palavras A, B e C, os arranjos possíveis são A, B, C, AB, BC e ABC. De acordo com a pontuação que cada arranjo atinge, o registro contendo todas as palavras é definido.

A função FQ contém três níveis de pontuação. O primeiro deles é o *Type Score*, que pontua o atributo candidato levando em conta o pareamento dele com algum tipo de dado específico, como valores numéricos, e-mail, data, URLs, códigos postais, entre outros. O segundo é *Language Model Score*, que computa a probabilidade de tal sequência de palavras ser um atributo com base no aparecimento de tal sequência em outras fontes recolhidas por um *web crawler*. O terceiro é o *Table Corpus Support Score*, que computa o número de vezes que o atributo candidato apareceu como valor de alguma célula em tabelas coletadas da web.

A fase de Separação é feita em cada registro de forma independente, podendo gerar um número de atributos diferentes entre os demais registros. As fases seguintes, Alinhamento e Refinamento, operam sobre os conjuntos de registros como um todo, transformando o resultado parcial em uma tabela, em que o número de atributos é o mesmo para todos os registros.

3.2. WWT, Gupta & Sarawagi, 2009.

Este trabalho apresenta o WWT, um mecanismo para extrair dados estruturados da web a partir de uma consulta em um formato de tabela. Esta consulta representa exemplos de dados estruturados, em um determinado domínio, em que cada linha da tabela representa um registro e cada coluna representa um determinado atributo, semelhante a uma tabela relacional. Dado a consulta de entrada, o algoritmo retorna uma tabela resposta com outros registros com o mesmo esquema definido na consulta.

Inicialmente, um grande número de listas foi inicialmente coletado e indexado com o software *Lucene*, da Apache. Nesse momento, nenhuma extração de dados estruturados é realizada. A heurística utilizada para coleta leva em consideração apenas o número de registros de cada lista, o tamanho médio dos registros e o número de delimitadores presentes.

A extração dos dados ocorre quando a consulta é submetida ao algoritmo. O processo inicia retornando exemplos de listas já coletadas que possuem registros com valores de atributos iguais aos valores de atributos dos registros da consulta, pareando os atributos da consulta com os atributos das listas já coletadas. A partir disso, infere-se que os demais valores dos atributos que foram pareados pertencem à relação proposta pela consulta e são alocados como um novo registro na tabela resposta.

3.3. Collie, Machanavajjhala (2011).

Este trabalho apresenta o Collie, um mecanismo para extração de dados estruturados de listas da Web baseado na homogeneidade de separadores entre seus registros internos e em registros já coletados em outras listas anteriormente.

Primeiramente o Collie extrai os dados de uma lista através de um algoritmo simples, que verifica se existe algum padrão de separadores entre os diversos registros da lista de entrada. Com base nos valores de atributos extraídos, outras fontes já extraídas anteriormente são resgatadas, juntamente com um conjunto de atributos pertinentes ao tipo de domínio identificado e entidades reais associadas aos registros coletados.

A partir disso, o algoritmo itera sobre quatro etapas. A primeira busca rotular e segmentar novamente a lista recém-coletada, a segunda atualiza as fontes de dados já extraídas anteriormente, a terceira rotula e segmenta outra vez a lista recém-coletada e a quarta atualiza os atributos pertinentes e as entidades reais. Essa iteração dura até haver uma convergência nas atualizações da lista coletada, atributos e entidades já existentes.

3.4. Análise dos trabalhos

Os três trabalhos relacionados mostrados utilizam grande quantidade de informações prévias para o processo de extração. Enquanto o ListExtract tira grande proveito de base de conhecimento, o WWT e o Collie exploram a comparação de diversas fontes coletadas em outros momentos para tal.

Motivando-se na busca por um mecanismo que utiliza pouco recurso computacional durante a extração, o SWLE procura realizar este processo a partir de cada fonte de forma independente e sem utilizar nenhuma base de conhecimento.

4. O Algoritmo de Extração SWLE

Este capítulo apresenta o algoritmo proposto neste trabalho, denominado SWLE. SWLE é um acrônimo para os termos *Structured Web List Extractor*. O objetivo do algoritmo é extrair dados de listas presentes em páginas na web que possuem, de forma implícita, múltiplos atributos.

A estratégia do SWLE é encontrar sequências de registros textuais em um documento HTML, adjacentes entre si, que contenham conteúdos alfanuméricos, e adicionalmente possuem conjuntos de caracteres, chamados de separadores, que separam o conteúdo de cada registro em duas ou mais partes. Para a lista ser considerada, para fins de extração dos seus dados, esses separadores precisam se repetir entre os demais registros e estarem presentes em cada um deles em uma mesma ordem. Cada conteúdo resultante dessa separação torna-se um valor de atributo para o esquema da lista extraído. Qualquer elemento textual no código pode se tornar um registro.

O algoritmo não se limita a extrair apenas registros associados às tags <ul> , como o nome pode sugerir, sendo a sua estratégia de extração genérica, conforme ilustra a figura 4. De acordo com a figura, dada uma estrutura HTML com uma sequência de dados que se repete (figura 4a), verifica-se que esta sequência apresenta um padrão de separadores que delimita itens de registros (figura 4b) e, a partir desta identificação, extrai-se e persiste-se os atributos desses registros (figura 4c).

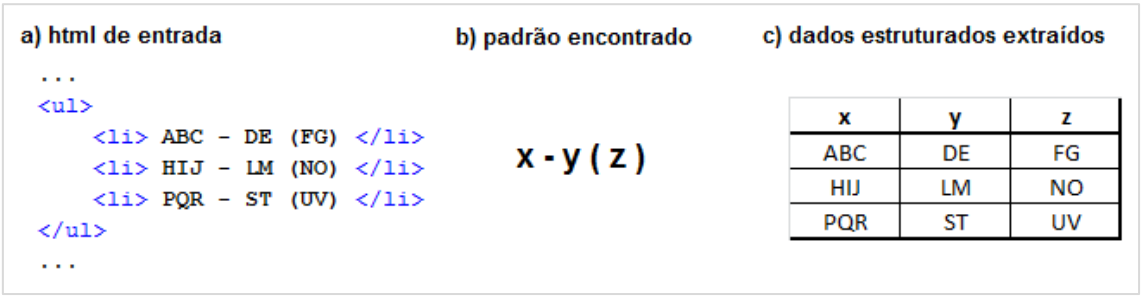


Figura 4: Exemplo de extração realizada pelo algoritmo SWLE

Diferentemente de trabalhos relacionados como Halevy (2009), o SWLE não utiliza bases de conhecimento que relacionam semanticamente um valor de atributo de uma lista na web com uma instância de um conceito de um determinado domínio de conhecimento. O esquema produzido pelo SWLE não possui atributos rotulados. As decisões para a coleta e separação são baseadas somente na disposição dos conteúdos textuais encontrados nos registros das listas.

Algorithm 1 Parser

```
1: procedure PARSE(STRING HTML)
2:     Document tree = JSoup.parse(html);
3:     Element body = tree.getElementByTag("body");
4:     parse(body);
5: end procedure
6:
7: procedure PARSE(NODE NODE)
8:     Registro reg = null;
9:     if node instanceof TextNode then
10:        String texto = node.text();
11:        reg = new Registro(texto);
12:    else if node instanceof Element then
13:        Node[ ] childNodes = node.childNodes();
14:        Registro[ ] childRegistros;
15:        for Node child : childNodes do
16:            Registro childRegistro = parse(child);
17:            if childRegistro ≠ null then childRegistros.add(childRegistro);
18:            end if
19:        end for
20:        if childRegistros.size > 0 then
21:            reg = agrupar(childRegistros);
22:        end if
23:    end if
24: end procedure
```

A entrada do SWLE é uma página HTML e sua saída é um conjunto com múltiplos objetos JSON, um para cada lista coletada da página. JSON é um formato leve, herdado da linguagem Java Script, para o intercâmbio de dados computacionais. Ele foi escolhido para manter o conteúdo extraído por ser um formato compacto e suficiente para representar uma estrutura tabular. Cada JSON possui seu próprio esquema, sendo definido independentemente dos demais esquemas que podem ser extraídos em uma mesma página. Cada registro coletado é separado, de acordo com o esquema extraído, em um

formato que pode ser diretamente mapeado para uma tabela, semelhante ao modelo de dados relacional.

O algoritmo SWLE está dividido em duas etapas principais: a etapa de *Parser* e a etapa de *Agrupamento*. Na etapa de *Parser,* o código HTML de entrada é transformado em uma estrutura de árvore em que cada nodo, a partir da raiz, recursivamente é visitado em busca de elementos textuais. Em cada visita, o algoritmo executa a etapa de *Agrupamento*, que faz com que os elementos textuais abaixo do novo visitado sejam avaliados em conjunto, podendo, a partir deles, serem extraídos um ou mais esquemas de listas e seus conteúdos. Estas duas etapas são detalhadas a seguir.

4.1. Parsing

A etapa de Parsing transforma o documento HTML de entrada em uma estrutura de árvore para facilitar a navegação entre seus elementos. Essa transformação é feita com o auxílio da API externa JSoup. Esta API foi escolhida por se tratar de algo já conhecido, ser utilizado em larga escala para estruturar o código HTML e por ser escrita na mesma linguagem utilizada para o desenvolvimento do SWLE, Java.

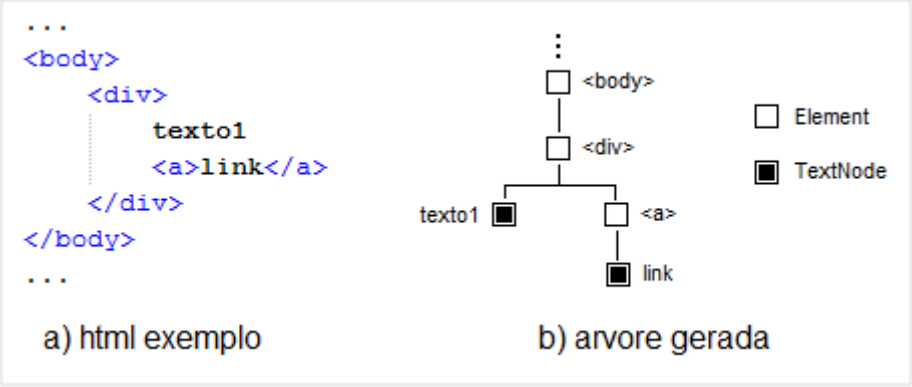


Figura 5: Exemplos de nodos *TextNode* e *Element*

Uma árvore HTML possui dois tipos de nodos. O primeiro são os nodos que representam textos, denominados neste trabalho de *TextNode.* Eles estão sempre localizados nas folhas da árvore. O segundo tipo são os nodos que representam as *tags*, aqui chamados de *Element*. Nessa árvore, o nodo raiz

sempre é representado pela tag <*html*>. Um exemplo desses dois tipos de nodos é mostrado na Figura 5.

Cada nodo da árvore HTML é visitado, com exceção dos metadados da página localizados na tag <*head*>. A visita é realizada de maneira recursiva e inicia a partir da tag <*body*>. Para cada visita, é retornado um objeto do tipo *Registro*. Esse objeto assume dois formatos diferentes, ou ele é nulo ou traz consigo elementos textuais coletados a partir de *TextNodes* localizados abaixo dele na hierarquia da árvore.

Se o nodo visitado for um *TextNode*, além do texto representado pelo nodo ser inserido no objeto Registro, este texto passa por um processo chamado de *tokenização*. Esse processo busca transformar um texto plano em um vetor de objetos do tipo *token*. Um *token* é a menor unidade de texto dentro de um Registro. Após um pedaço de texto ser tokenizado, ele não pode ser dividido em nenhuma das etapas futuras do algoritmo.

O processo é realizado em duas partes. A primeira delas separa o texto em substrings a cada espaço em branco entre dois caracteres não nulos. Em seguida, para cada substring gerada, todo caractere especial em suas extremidades são separados. Todo caractere que não for uma letra ou um número é considerado um caractere especial.

Cada pedaço de texto não nulo dessa separação torna-se um token. Um exemplo de execução deste processo é mostrado na Figura 6.

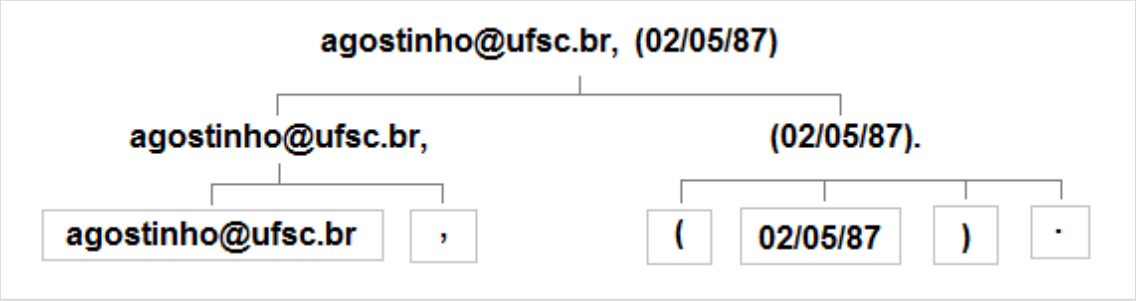


Figura 6: Exemplo de tokenização

Se o nodo visitado for um *Element*, é criado um vetor de objetos do tipo *Registro*, que é preenchido com os *Registros* resultantes da visita de cada um

de seus nodos filhos. Em seguida, esse vetor passa pela etapa de *Agrupamento*, responsável por identificar se existem sequências de registros dentro desse vetor, adjacentes uns aos outros, que possam ser caracterizados como uma lista contendo dados estruturados.

A etapa de agrupamento, detalhada a seguir, também retorna um objeto do tipo *Registro*. Se a partir do vetor de Registros recebido como entrada, a etapa de Agrupamento encontrar uma ou mais listas com dados estruturados, essas listas são extraídas, armazenadas em um objeto temporário, e é retornado um objeto *Registro* nulo. Caso não for encontrada nenhuma lista estruturada, é retornado um objeto Registro com todos os elementos textuais dos registros do vetor de entrada concatenados uns aos outros.

4.2. Agrupamento

A etapa de Agrupamento possui as principais decisões heurísticas do algoritmo SWLE. Ela recebe como entrada um vetor de objetos do tipo Registro e busca encontrar uma ou mais sequências destes objetos com alguma semelhança entre seus *tokens*, que os faça serem considerados uma lista com dados estruturados.

Ela inicia com o processo denominado *Derivação*, responsável por encontrar *tokens* em comum entre um par de registros adjacentes. Em seguida, ocorre o processo de clusterização, que agrupa os registros semelhantes a fim de torná-los candidatos a uma lista de dados estruturados. Caso estes grupos obtenham as condições mínimas para serem separados, inicia-se então o processo de separação dos atributos implícitos nos registros e sua transformação em uma estrutura tabular para fins de persistência. Cada um destes processos é detalhado a seguir.

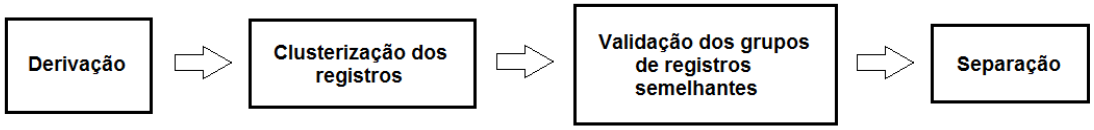


Figura 7: Fluxograma da etapa de agrupamento

4.2.1 Derivação

O processo de derivação é responsável por descobrir quais *tokens* são comuns entre registros adjacentes. Infere-se que estes *tokens* em comum são os separadores que delimitam os valores de atributos de cada registro presente em uma lista. Para cada par de registros, no final do processo, é gerado um objeto *Nodo de Derivação*, que contém essa informação. O *nodo de derivação* indica cada Registro que ele derivou.

O processo de derivação pode ser executado múltiplas vezes, a partir do resultado de uma derivação anterior, ou seja, em uma segunda etapa, ou posterior, os dados de entrada para o processo são os próprios *nodos de derivação* gerados anteriormente. O número de derivações que o algoritmo executa é definido em um objeto de configurações, explicado em detalhes mais adiante.

A figura 8 a seguir demonstra o processo com duas derivações.

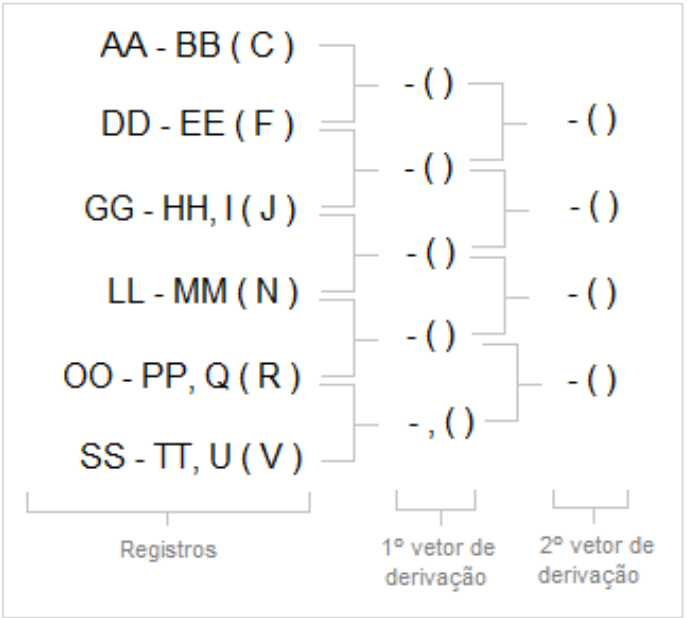


Figura 8: Exemplo de execução do processo de Derivação

A partir dos registros de entrada, é gerado o primeiro vetor de derivação. Nele podemos observar que cada nodo do primeiro vetor possui apenas os *tokens* que são em comum entre os registros adjacentes a partir dos quais ele foi derivado. No segundo vetor podemos perceber o mesmo comportamento.

Apesar da entrada para a segunda derivação ser um vetor com uma grande homogeneidade em relação ao conteúdo de seus nodos, o vetor gerado consegue eliminar mais um *token* que difere dos demais nodos anteriores.

4.2.2. Clusterização de Registros

Após a execução de uma ou mais derivações, o último vetor de *nodo de derivação* gerado é entrada para o processo seguinte, a clusterização de Registros. Um vetor de derivação pode conter nodos nulos ou nodos com um ou mais *tokens*. Na figura 9, o nodo derivado a partir do quarto e quinto registro é nulo, pois entre estes dois registros não há nenhum *token* em comum. O mesmo ocorre em nodos do segundo vetor, em que a derivação a partir de um nodo nulo sempre vai gerar um nodo derivado nulo. Para cada subsequência de nodos não nulos do último vetor de derivação é criado um *grupo de registros semelhantes*, que mantém todos os registros derivados pelos Nodos de Derivação da subsequência que os contém. A Figura 9 exemplifica esse processo.

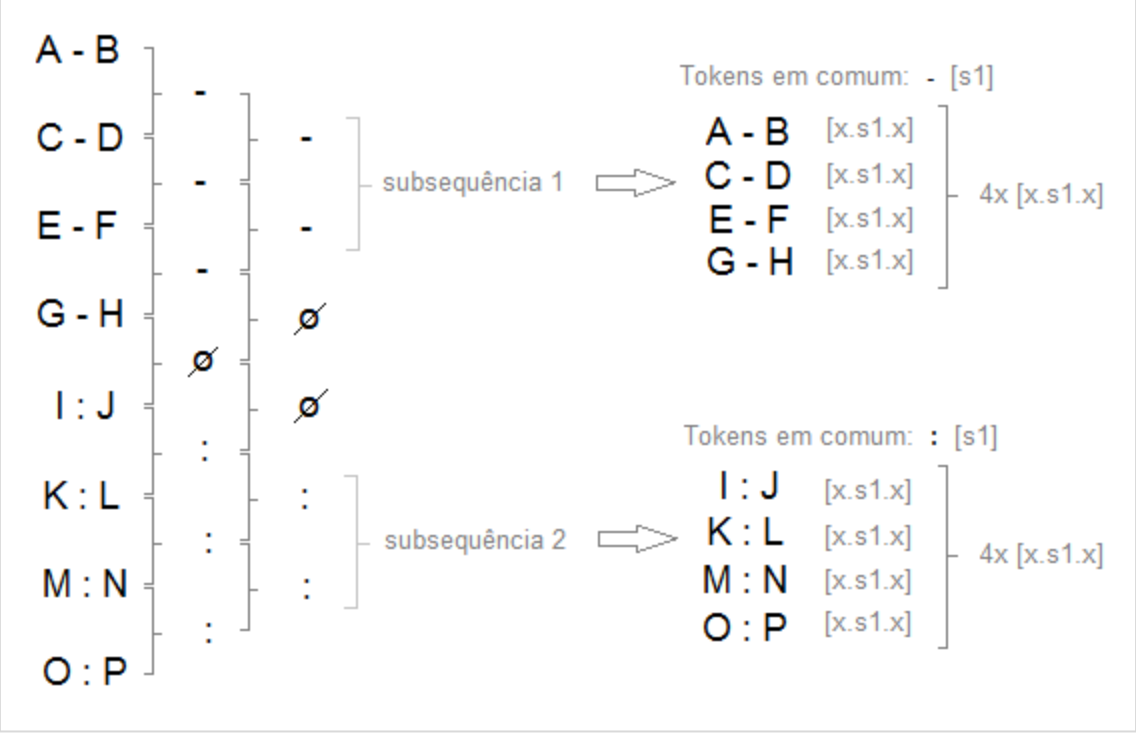


Figura 9: Exemplo do Processo de Clusterização de Registros

Os tokens contidos nos nodos de derivação da subsequência, comuns entre os Registros dos quais foram derivados, são associados ao *grupo de registros semelhantes* gerado. Para cada um desses *tokens* é associada uma identificação. Na subsequência 1 da Figura 8, existe apenas um *token* presente nos nodos de derivação, o traço. Na subsequência 2, o único *token* presente é o caractere dois pontos.

Para cada Registro contido no grupo, é gerada uma descrição que leva em consideração os *tokens* derivados e os demais *tokens* dos registros, que representam os valores de atributos. Essa descrição é chamada de *Identidade*. Ela é descrita por uma *string* que espelha o registro original, trocando os *tokens* derivados pela sua respectiva identificação e os demais *tokens*, referentes aos conteúdos dos atributos dos registros, aglutinados por um caractere genérico. Na Figura 9, tanto para a primeira e segunda subsequência, todos os seus registros possuem dois conteúdos textuais separados por um *token* contido nos nodos de derivação. Por isso, cada subsequência possui registros com as mesmas identidades.

4.2.3. Validação dos Grupos de Registros Semelhantes

Um grupo de registros semelhantes sempre vai conter ao menos uma identidade que descreve os seus registros, mas é possível que ele possua mais. A Identidade que descreve a maior quantidade de registros no grupo é denominada *Identidade Padrão*.

A primeira validação feita sobre o grupo visa verificar se a quantidade de registros descritos pela identidade padrão ultrapassa a quantidade mínima requerida, dado o número de registros totais desse grupo, ou seja, é definido previamente um percentual mínimo de registros que a identidade padrão deve descrever. Caso esse percentual não atinja esse mínimo, a lista é descartada. Essa validação é feita para eliminar listas com registros muito poluídos, que atrapalhem no processo de separação.

A segunda validação verifica se a identidade com maior percentual de descritos possui ao menos dois caracteres genéricos separados por *token*

derivado, ou seja, se os registros descritos pela identidade padrão possuem ao menos dois valores de atributos que são delimitados por um separador.

Por fim, a terceira validação verifica se o número de registros do grupo é maior ou igual à quantidade mínima de registros por lista coletada.

4.2.4. Separação

Um grupo de registros que tiver sucesso nas três validações anteriores é submetido a um próximo processo denominado *Separação*. Este processo utiliza a identidade padrão como parâmetro para a definição do esquema de saída a ser persistido.

A princípio, os dados são organizados em uma matriz com o número de colunas igual ao número de separadores da identidade padrão, mais um. Todos os registros, mesmo aqueles que não são descritos pela identidade padrão, sofrem o processo de separação. Entretanto, nem todos são inseridos na matriz resultante.

Se ao percorrer o vetor de tokens de um registro, todos os tokens separadores da identidade padrão estiverem presentes, o registro é armazenado na matriz, tendo seus tokens agrupados nas colunas de acordo com o posicionamento dos seus separadores no vetor de tokens do registro. O Algoritmo 2 apresenta o processo de separação.

O processo de separação gera uma matriz com um número de linhas maior ou igual à quantidade de registros descritos pela identidade padrão. Alguns registros, que possuem a identidade diferente da identidade padrão, também podem completar o processo de separação bastando que seja possível, apenas com a remoção de alguns separadores, igualar o vetor de tokens da identidade do registro candidato com o vetor de tokens da identidade padrão. Supondo que a identidade padrão possua os tokens *x* e *y*, nesta ordem. Caso um registro possua a identidade contendo os separadores *a, x, b, y, c,* ele vai completar o processo de separação, pois é possível remover os separadores *a, b, c,* da identidade do registro, igualando com os separadores da identidade padrão.

Algorithm 2 Separador

```
1: procedure SEPARAR(TOKEN[ ] TOKENSDOREGISTRO, TOKEN[ ] SEPARADORES)
2:     Atributo[ ] atributos;
3:     int indexSeparadores = 0;
4:     Atributo att = new Atributo();
5:     boolean completo = false;
6:     for int i = 0 ; i < tokensDoRegistro.size ; i++ do
7:         if tokensDoRegistro[i] = separadores[indexSeparadores] then
8:             Atributo attLast = new Atributo();
9:             for int j = i+1 ; j < tokensDoRegistro.size ; j++  do
10:                attLast.add(tokensDoRegistro[j]);
11:            end for
12:            atributos.add(attLast);
13:            completo = true
14:            break;
15:        else
16:            atributo.add(tokensDoRegistro[i]);
17:        end if
18:    end for
19:    if completo then
20:        return atributos;
21:    else
22:        return null;
23:    end if
24: end procedure
```

As colunas da matriz gerada pelo processo de separação não necessariamente estarão todas preenchidas. Se todos os registros tiverem uma identidade que termina com um separador, por exemplo, a última coluna da matriz será vazia. Caso for encontrada uma coluna inteira vazia, ela é removida da matriz.

É muito comum encontrar construções na Web que possuem uma listagem de itens associados a um pedaço de texto que se repete em todos os seus registros. Caso for encontrada uma coluna em que seus valores possuam uma repetição superior a uma taxa pré-definida, essa coluna também é removida, por não ser considerada relevante.

Por fim, após o processo de separação e de remoção de colunas, caso a matriz, que guarda registros de apenas uma lista, ainda possua o número mínimo de registros definidos previamente e o número de colunas igual ou superior a dois, a extração é considerada satisfatória. Assim sendo, a matriz resultante é transformada em um objeto JSON, que imita uma estrutura tabular.

4.3. Variáveis de configuração

Em diversos pontos do algoritmo SWLE, algumas decisões são baseadas em um conjunto de variáveis de configuração, como por exemplo, o número mínimo de registros que uma lista deve conter para que ela seja coletada. Dependendo do valor atribuído a esse critério, o resultado da coleta para uma mesma página HTML de entrada pode ser diferente.

Essas variáveis nos permitem parametrizar os diversos processos que fazem parte do SWLE, fazendo com que possamos executar o algoritmo diversas vezes para uma mesma entrada, cada uma contendo um conjunto de diferentes valores para as variáveis de configuração. Observando o resultado da execução em diversas páginas HTML de entrada, utilizando diversas combinações de valores para as variáveis de configuração, é possível tirar conclusões a respeito das decisões tomadas nos processos realizados pelo algoritmo.

Cinco são as variáveis de configuração que o algoritmo SWLE utiliza. As seções a seguir, explicam a motivação para cada uma delas, os possíveis valores e como a escolha de um determinado valor afeta na execução do algoritmo.

4.3.1. Quantidade Mínima de Registros por Lista – RL

A primeira variável de configuração diz respeito ao número mínimo de registros que uma lista coletada deve conter. Isso significa que nenhum esquema de lista extraído de uma página de entrada vai conter um número de registros, já separados, menor do que o valor definido na variável. Um número baixo para esse valor aumenta as chances de que sejam coletadas sequências de elementos textuais que não representam registros com dados estruturados. Isso porque é alta a probabilidade de se obter um pequeno número de supostos registros com pelo menos um caractere especial em comum entre eles, que não tenham sido criados com a intenção de definir uma lista.

Essa variável tem efeito em várias decisões ao longo do algoritmo. Na etapa de agrupamento, o processo de derivação se inicia somente se o vetor de registros tiver uma quantidade igual ou superior ao valor desta variável. Um grupo de registros semelhantes, criado após as derivações, só passa pelo processo de separação se o número de registros desse grupo for maior ou igual ao valor da variável. Por fim, se o número de registros que completam o processo de separação for menor que esse valor, o esquema não é extraído.

4.3.2. Quantidade Máxima de Caracteres por Registro - CR

A segunda variável refere-se à quantidade máxima de caracteres que cada registro da lista pode conter. Valores altos para essa variável tornam o algoritmo propenso a resgatar pedaços de textos que não representam algo estruturado.

Na etapa de Agrupamento, quando o número de registros não atinge o mínimo definido, os registros do vetor possuem seus tokens compilados em apenas um objeto registro. Caso este registro compilado possua um número de caracteres superior ao número definido nesta variável, então o objeto registro de retorno passa a ser um objeto nulo.

4.3.3. Quantidade de Derivações - ND

No processo de derivações seguido pelo SWLE, dado um vetor de registros, uma única derivação faz com que o vetor resultante contenha o tamanho do vetor de registros menos um. O processo poderia continuar, derivando-se o vetor resultante em novos vetores até que restasse um vetor com apenas um nodo de derivação. Porém, o algoritmo também parametriza este critério, tornando-o um número fixo, mesmo que um número menor de derivações gere um vetor com todos os nodos contendo os mesmo tokens.

Com uma quantidade de derivações pequena, podem ser selecionados tokens indesejados para a geração da identidade de cada registro resultante. Isso faz com que o número de identidades para um grupo de registros semelhantes cresça, fazendo com que a identidade padrão do grupo não contenha o percentual mínimo de registros que ela descreve.

4.3.4. Percentual Mínimo para Identidade Predominante - IP

A relação entre a quantidade de registros que são descritos pela identidade padrão e a quantidade total de registros precisa atingir um valor mínimo para um grupo de registros semelhantes ser validado. Esse valor é definido nesta quarta variável.

Um valor mínimo muito baixo para o percentual faz com que sejam aceitos grupos que contenham muitas identidades diferentes, aumentando a chance de serem extraídas listas que não apresentem registros com uma estrutura adequada.

4.3.5. Percentual Máximo de Valores Repetidos em uma Coluna - RC

Após a execução do processo de separação pelo SWLE, além de serem eliminadas da matriz colunas vazias, são também removidas colunas que apresentam um valor textual que se repete por um percentual maior do que o definido por esta quinta variável.

Colunas com textos que apresentam uma alta taxa de repetição podem indicar elementos textuais que figuram no meio de uma lista por motivos adversos, como por exemplo, uma lista que contenha junto a cada registro, um link de mesmo valor textual para alguma página ou um anúncio.

4.4. Considerações finais do capítulo

Este capítulo apresentou em detalhes o algoritmo proposto neste trabalho para a extração de listas de dados estruturados na Web. O próximo capítulo descreve avaliações experimentais realizadas com ele.

5. Experimentos

Dois experimentos foram realizados para avaliar a qualidade do algoritmo SWLE para a extração de listas estruturadas da Web. O primeiro deles testa a precisão do algoritmo para diversas combinações de variáveis de configurações definidas no capítulo anterior. As páginas Web de entrada utilizadas nesse primeiro experimento foram determinadas de forma aleatória, através de um conjunto de *links* recolhidos por um *crawler*. O segundo experimento testa a precisão e revocação do algoritmo a partir de um conjunto de links recolhidos manualmente tais que, visualmente em sua página, identificam-se listas com dados estruturados. Para cada página de entrada também foi executado o algoritmo para as mesmas combinações de variáveis de configuração utilizadas no primeiro experimento.

Para cada variável de configuração é possível atribuir um valor que torne a extração mais liberal, tornando-a mais propensa a aceitar listas que não representem dados estruturados, ou um valor que a deixe mais conservadora, podendo deixar de fora listas que representam dados estruturados. A partir disso, para cada uma destas variáveis, foram definidos valores que retratam esses dois aspectos. As valorações das cinco variáveis de configuração foram resumidas em três grupos, possuindo cada qual um valor V1 (liberal) e um valor V2 (conservador).

Variável	**V1**	**V2**
Quantidade mínima de registros por lista – **RL**	3	6
Quantidade de derivações - **ND**	2	4
Quantidade máxima de caracteres por registro - **CR**	160	80
Percentual mínimo para Identidade predominante - **IP**	0,5	0,9

Tabela 1: Valores definidos para as variáveis de configuração nos experimentos

Para a variável que indica o percentual máximo de valores repetidos em uma coluna foi atribuído um valor fixo de 0,8 para todos os experimentos. Esse valor foi fixado para diminuir a complexidade dos experimentos.

Para uma quantidade de derivações igual a 4, é necessário que o número mínimo de registros por lista seja 5, pois a cada derivação o vetor de saída

sempre possui o tamanho diminuído em uma unidade do vetor de entrada. Em função disso, a quantidade de derivações definida com o valor 4 sempre estará associada ao valor 6 para a quantidade mínima de registros por lista. Já o número de derivações com valor 2 sempre estará associado ao valor 3 para a quantidade mínima de registros por lista.

Como temos 3 grupos de variáveis, cada um com possíveis valores atribuídos às variáveis, foram definidas 8 combinações possíveis de parâmetros de configuração para o SWLE. Cada combinação, denominada *settings*, pode ser conferida na Figura 10.

Settings 1
RL: 3
ND: 2
CR: 160
IP: 0,5

Settings 2
RL: 3
ND: 2
CR: 80
IP: 0,5

Settings 3
RL: 6
ND: 4
CR: 160
IP: 0,5

Settings 4
RL: 6
ND: 4
CR: 80
IP: 0,5

Settings 5
RL: 3
ND: 2
CR: 160
IP: 0,9

Settings 6
RL: 3
ND: 2
CR: 80
IP: 0,9

Settings 7
RL: 6
ND: 4
CR: 160
IP: 0,9

Settings 8
RL: 6
ND: 4
CR: 80
IP: 0,9

Figura 10: Grupo de valores das variáveis de configuração definidos para os experimentos

5.1. Experimento I

O primeiro experimento avalia a precisão do algoritmo para cada configuração estabelecida. Para isso, um *crawler* foi construído para coletar aleatoriamente uma grande quantidade de páginas na web. O *crawler* recebeu como sementes iniciais 11 endereços de portais de notícias e conteúdos na língua portuguesa:

- http://www.pmf.sc.gov.br
- http://www.brasil.gov.br
- http://www.portalinformacao.net

- http://g1.globo.com/index.html
- http://noticias.uol.com.br
- http://www.jn.pt/paginainicial
- http://www.folha.uol.com.br
- http://noticias.terra.com.br
- https://br.noticias.yahoo.com
- http://www.dn.pt/inicio/default.aspx
- http://www.msn.com/pt-br/noticias

Na tentativa de diversificar os links coletados, a cada 100 visitas feitas pelo *crawler* para a coleta de novos links, elas foram realizadas em domínios que não se repetiam. Em 10 mil visitas, foram coletados cerca de 1,5 milhão de novos *links*. A partir destes, um subconjunto de 10 mil deles foi selecionado, também de forma aleatória, como fonte de entrada para o algoritmo. Cada *link* selecionado foi submetido ao algoritmo nas 8 combinações propostas de configuração. Cada lista estruturada extraída é associada ao *link* de entrada e à configuração utilizada.

Em seguida, para cada configuração, foram selecionadas 50 páginas em que foram extraídas uma ou mais listas. Nesse experimento cada configuração é avaliada separadamente das demais. As páginas avaliadas por uma configuração não correspondem necessariamente às páginas de outra configuração.

Para avaliar o algoritmo SWLE com uma determinada configuração, em cada página selecionada foi verificado se as listas coletadas representam ou não dados estruturados. Para validar se a extração teve sucesso para uma determinada lista, foi verificado se cada registro coletado apresenta dois ou mais atributos implícitos, que são compartilhados entre os demais, na mesma ordem de apresentação.

O valor da precisão é a razão entre o número de listas estruturadas relevantes que foram extraídas e o número total de listas extraídas. O valor para a precisão de cada combinação pode ser conferido na Figura 11. Nela podemos facilmente verificar que, quando as variáveis possuem um valor mais

conservador, a precisão, em cinza escuro no gráfico, tende a ser maior. Em cinza claro temos o complemento do valor da precisão, ou seja, a razão entre o número de listas coletadas que não representam informação estruturada e o número total de listas coletadas.

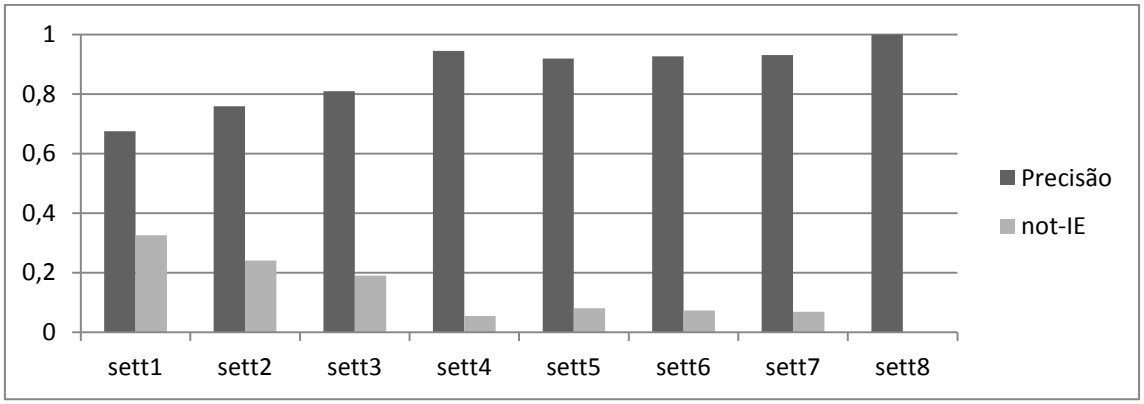


Figura 11: Resultados do experimento I

5.2. Experimento II

No segundo experimento, o conjunto de entrada utilizado foi o mesmo para cada uma das 8 configurações previamente estabelecidas. Esse conjunto possui um total de 52 páginas, escolhidas manualmente, contendo uma ou mais listas com dados estruturados. Cada uma dessas páginas é submetida ao algoritmo SWLE para cada configuração e as listas extraídas, assim como no primeiro experimento, são associadas à página e à configuração utilizada.

Nesse experimento são verificadas a precisão e a revocação para o algoritmo SWLE. Para cada página, se sabe a quantidade e quais são as listas com dados estruturados nela existentes. Esse conjunto é definido como *Informações Estruturadas Relevantes (IE-R)*. As listas extraídas, para uma determinada configuração, são comparadas com os elementos da IE-R e são atribuídos a eles dois possíveis rótulos. O primeiro rótulo indica uma *Informação Estruturada Relevante Coletada (IE-RC)*, caso a lista extraída faça referência a um elemento de IE-R e contenha pelo menos 50% dos registros que o elemento de IE-R apresenta. O segundo rótulo indica uma Informação

Estruturada Não-Relevante Coletada (IE-nRC), caso não haja nenhuma referência a algum elemento de IE-R.

A tabela 2 detalha a execução do algoritmo em cada uma das 52 páginas consideradas. Cada linha da tabela faz referência a uma das páginas em que se verificou manualmente a existência de listas com dados estruturados. Na última coluna é mostrado o valor de IE-R associado à página, ou seja, o número de listas com dados estruturados relevantes existentes na página. Para cada um dos *settings* é atribuído o valor para a quantidade de listas relevantes e não relevantes coletadas pelo algoritmo. No rodapé da tabela temos a soma desses valores para cada um dos *settings*, que são suficientes para computarmos os seus valores de precisão e revocação. Esses valores são mostrados também no gráfico da Figura 12.

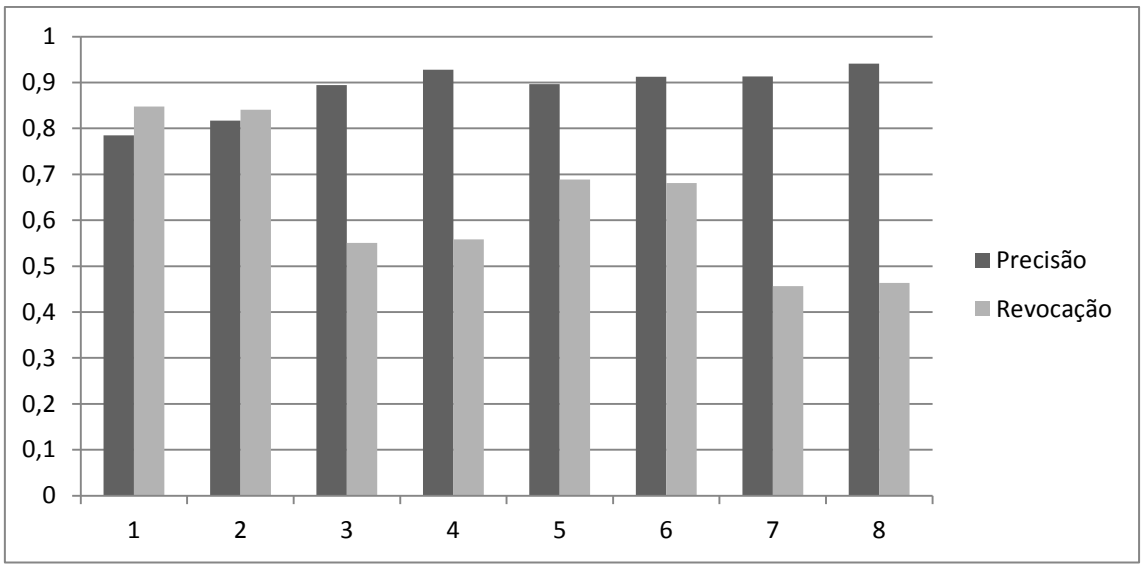


Figura 12: Resultados do experimento II

A precisão, assim como no experimento anterior, é contabilizada como a razão entre o número de listas relevantes extraídas e o total de listas extraídas, ou seja, a razão entre |IE-RC| e |IE-RC| +|IE-nRC|. Já a revocação é computada como a razão entre o número de listas relevantes coletadas e o número total de listas com informação estruturada relevante nas páginas, ou seja, é a razão entre |IE-RC| e |IE-R|.

Apesar dos 2 experimentos apresentarem valores absolutos diferentes para a precisão em cada configuração, conseguimos tirar conclusões importantes dos resultados apresentados. Ambos os gráficos apresentam uma

curva similar, apresentando uma subida da primeira até a quarta configuração, e em seguida uma leve concavidade entre a quarta e oitava.

A precisão se mostrou mais baixa nas duas primeiras configurações, que apresentam um número mínimo de registros (RL) pequeno, um número de derivações (ND) pequeno e um percentual para identidade predominante (IP) pequeno. Para as configurações restantes, pelo menos uma dessas variáveis apresenta um valor mais conservador.

5.3. Falhas na extração que independem das configurações

Na tabela 2 podemos verificar que alguns dos endereços visitados (49-52) não coletaram em nenhuma das configurações propostas nenhuma das listas com informação estruturada que se esperava coletar. Isso ocorreu devido a um problema intrínseco ao algoritmo, em que os valores das variáveis não dizem respeito ao correto funcionamento.

O SWLE agrupa os registros candidatos que estão localizados em um nodo *Element* em comum. Devido à forma como são organizadas algumas listas estruturadas no código HTML das páginas web, partes dos registros podem ficar divididas entre diversos nodos diferentes e o resultado da extração pode apresentar diversas listas diferentes, quando na verdade é uma mesma lista dividida em colunas, sessões ou subtítulos diferentes. Uma das possibilidades de melhorias no SWLE é uma maneira de aglutinar essas listas subsequentes que apresentam o mesmo valor para identidade padrão, inferindo-se que elas representam dados de um mesmo domínio, com os mesmos atributos.

Na etapa de Parsing, todos os registros acumulados a partir dos nodos filhos de um nodo *Element* são colocados em um vetor e enviados para a etapa de Agrupamento, que decide se tais registros apresentam algum padrão quanto à presença de caracteres especiais. Caso os diferentes registros apresentarem um texto plano, sem nenhuma tag interna, e o que separa um registro do outro for uma *tag
,* nada de errado ocorre. Caso os diferentes registros fiquem concentrados cada qual em uma tag, que por sua vez engloba internamente

múltiplas tags, este cenário também não gera problemas. Porém, se temos uma lista com registros que são separados por uma *tag* <*br*> e que, mesmo assim, eles possuem *tags* internas que identificam algum atributo em específico, isso gera um erro na extração. A Figura 13 ilustra os três cenários descritos.

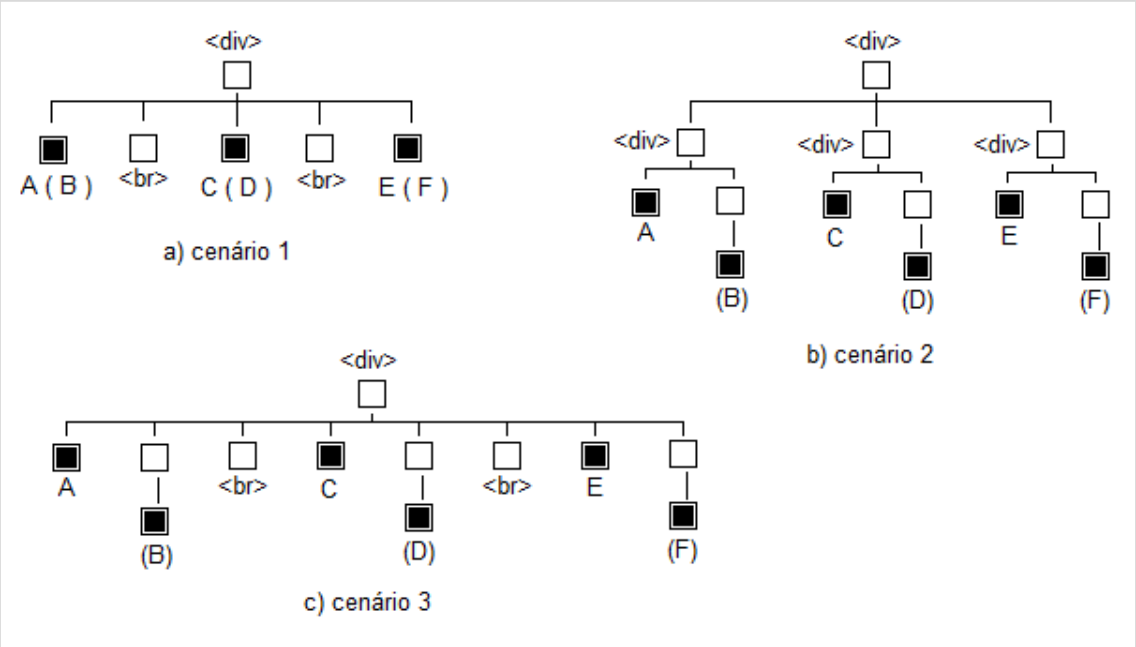


Figura 13: Cenários possíveis na etapa de Agrupamento do SWLE

Nos dois primeiros cenários, durante a etapa de Parsing, quando o elemento raiz é visitado, ele vai conter um vetor com três registros, cada um deles contendo um texto semelhante a X (Y). Na etapa de Agrupamento, o algoritmo não encontra problemas para identificar o padrão dos registros e consegue extrair a lista.

No terceiro cenário, entretanto, vão existir seis registros. Três deles vão ser do tipo X, e os outros três, intercalados com os demais, vão ver do tipo (Y). Na etapa de Agrupamento, não é encontrado um padrão para estes registros, já que o padrão não se repete a cada registro adjacente. No processo de Derivação, o primeiro vetor de nodos derivados gerados conterá nodos vazios, descartando, assim, uma lista relevante.

link	sett1		sett2		sett3		sett4		sett5		sett6		sett7		sett8		IE-R
	IE-RC	IE-nRC	IE-RC	IE-nRC	IE-RC	IE-nRC	IE-RC	IE-nRC	IE-RC	IE-nRC	IE-RC	IE-nRC	IE-RC	IE-nRC	IE-RC	IE-nRC	
1	3	2	3	2	2	1	2	1	2	2	2	1	1	1	1	1	4
2	7	0	7	0	2	0	2	0	7	0	7	0	2	0	2	0	7
3	1	4	1	2	1	1	1	1	1	3	1	2	1	1	1	1	1
4	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
5	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1
6	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
7	1	2	1	1	1	0	1	0	0	1	0	1	0	0	0	0	1
8	3	0	3	0	2	1	2	0	0	1	0	0	1	1	1	0	4
9	3	1	2	1	3	1	2	0	1	0	1	0	2	0	1	0	3
10	2	0	2	0	2	0	2	0	2	0	2	0	2	0	2	0	2
11	1	1	1	1	1	0	1	0	1	1	1	1	1	0	1	0	1
12	2	1	2	2	2	2	2	2	1	0	1	0	1	0	1	0	2
13	2	1	2	1	2	0	2	0	2	0	2	0	2	0	2	0	2
14	3	0	2	0	2	0	1	0	2	0	1	0	1	0	0	0	4
15	13	7	13	6	5	1	5	1	13	0	13	1	5	1	5	1	15
16	1	1	2	1	1	1	2	0	1	0	1	1	1	0	1	1	2
17	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
18	1	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
19	1	2	1	1	1	0	1	0	1	1	1	1	1	0	1	0	1
20	2	0	2	0	2	0	2	0	2	0	2	0	2	0	2	0	2
21	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
22	3	0	2	0	1	0	1	0	3	0	2	0	1	0	1	0	3
23	11	0	11	0	5	0	5	0	9	0	9	0	5	0	5	0	11
24	4	1	4	1	4	0	4	0	4	0	4	0	4	0	4	0	4
25	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
26	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
27	3	3	3	3	3	0	3	0	3	0	3	0	3	0	3	0	3
28	16	0	16	0	5	0	5	0	14	0	14	0	4	0	4	0	16
29	2	0	2	0	2	0	2	0	0	0	0	0	0	0	0	0	2
30	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
31	1	1	1	1	1	0	1	0	0	1	0	1	0	1	1	0	2
32	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
33	3	0	3	0	1	0	1	0	1	0	1	0	1	0	1	0	3
34	6	0	6	0	4	0	4	0	5	0	6	0	3	0	4	0	7
35	0	2	0	2	0	1	0	1	0	1	0	0	0	1	0	0	5
36	2	1	2	1	1	0	1	0	1	0	1	0	0	0	0	0	4
37	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1
38	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	1
39	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
40	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
41	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
42	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
43	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
44	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
45	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
46	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
47	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
48	1	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
49	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
51	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	117	32	116	26	76	9	77	6	95	11	94	9	63	6	64	4	138
Precisão	0,79		P 0,82		P 0,89		P 0,93		P 0,9		P 0,91		P 0,91		P 0,94		
Revocação	0,85		R 0,84		R 0,55		R 0,56		R 0,69		R 0,68		R 0,46		R 0,46		

Tabela 2: Detalhamento da execução do slagoritmo SWLE para as 52 páginas Web consideradas no segundo experimento

5.4. Análise dos Resultados

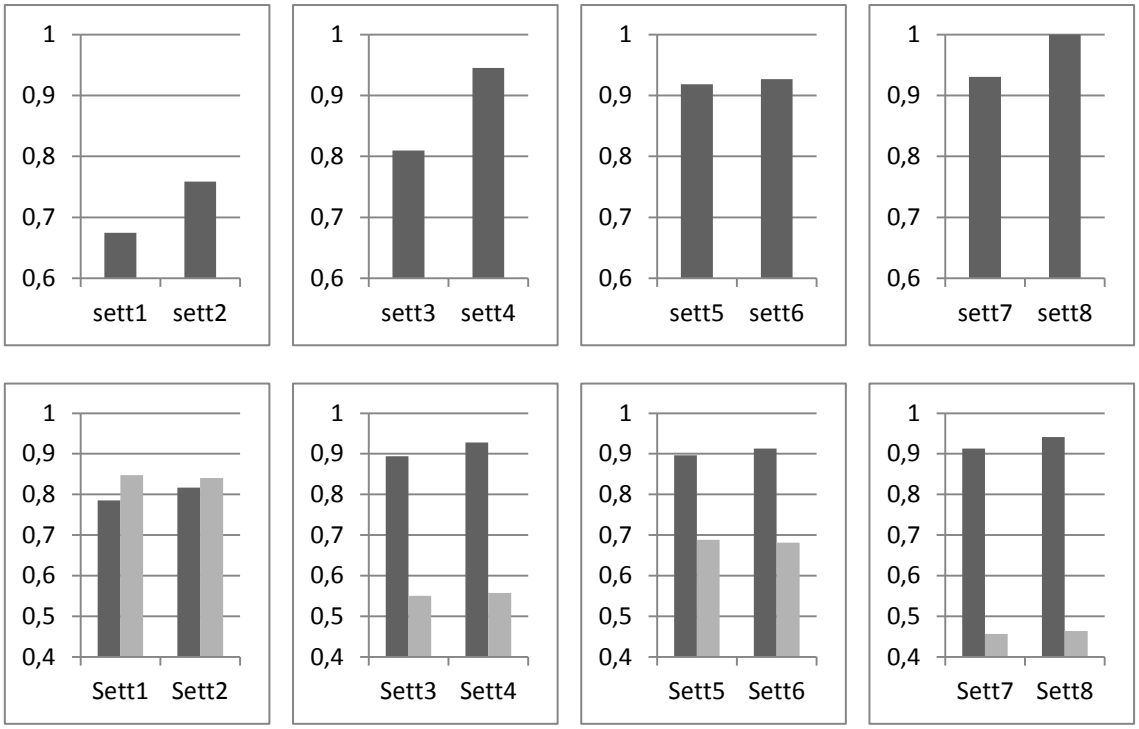
Conclusões a respeito das variáveis de configuração do SWLE podem ser obtidas de maneira isolada. Para cada uma delas, pode-se comparar o valor da precisão e revocação entre dois *settings* que apresentam valores diferentes para a variável que se deseja analisar e com as demais variáveis apresentando o mesmo valor.

Cada coluna, apresentada nas 3 figuras a seguir, mostra o resultado do primeiro e segundo experimento, respectivamente, para os valores de *settings* definidos na tabela da mesma coluna. Em uma mesma coluna, o primeiro gráfico mostra os valores da precisão do primeiro experimento e o segundo (abaixo) mostra os valores de precisão e revocação do segundo experimento. Na Figura, a variável que apresenta 2 valores é a variável que está sendo comparada entre o primeiro e segundo *settings* nos gráficos dos 2 experimentos acima.

5.4.1. Quantidade Máxima de Caracteres por Registro (CR)

Definir um valor alto para a variável CR diminui as chances de se obter listas que representem dados estruturados, conforme podemos verificar nas análises entre as configurações pareadas com as demais variáveis. Um valor baixo eleva essas chances porque é mais improvável que ele represente um pedaço de texto plano.

Outro ponto que torna o valor alto de CR mais suscetível à coleta de dados não estruturados é a lógica utilizada na etapa de agrupamento dos registros quando não foi extraída uma lista a partir dos registros de entrada. Quando isso ocorre, os dados são concatenados e retornados para a etapa de parser somente se o número de caracteres dessa concatenação for menor ou igual ao valor de CR. Quando CR está alto, o registro formado a partir do texto concatenado que é retornado para a etapa de parsing possui maiores chances de não possuir atributos implícitos.



IP:	0,5	
RL:	3	
ND:	2	
CR:	160	80

IP:	0,5	
RL:	6	
ND:	4	
CR:	160	80

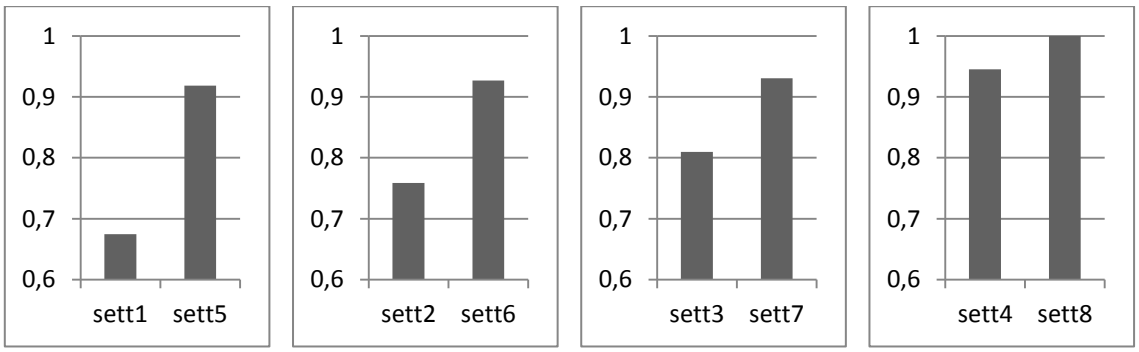
IP:	0,9	
RL:	3	
ND:	2	
CR:	160	80

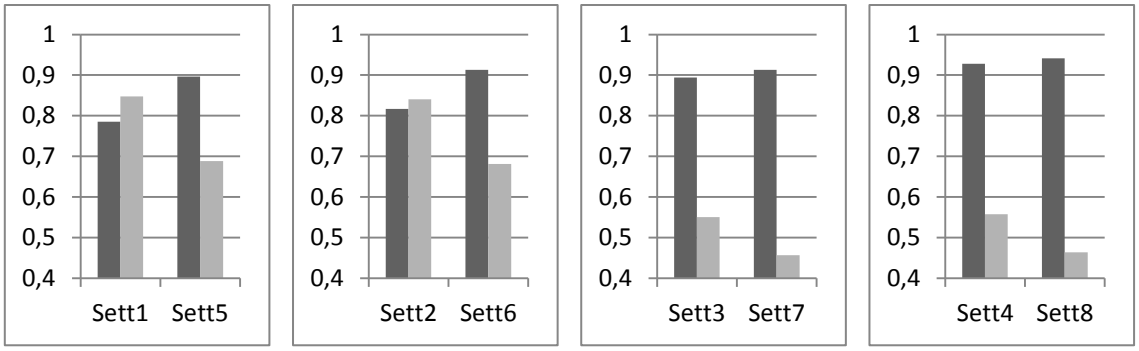
IP:	0,9	
RL:	6	
ND:	4	
CR:	160	80

Figura 14: Análise dos resultados da variável CR

Como as páginas escolhidas para o segundo experimento possuem listas com registros com menos de 80 caracteres, o valor de revocação para cada configuração pareada acima não possui grandes diferenças.

5.4.2. Percentual Mínimo para Identidade Predominante (IP)





CR:	160	
RL:	3	
ND:	2	
IP:	0,5	0,9

CR:	80	
RL:	3	
ND:	2	
IP:	0,5	0,9

CR:	160	
RL:	6	
ND:	4	
IP:	0,5	0,9

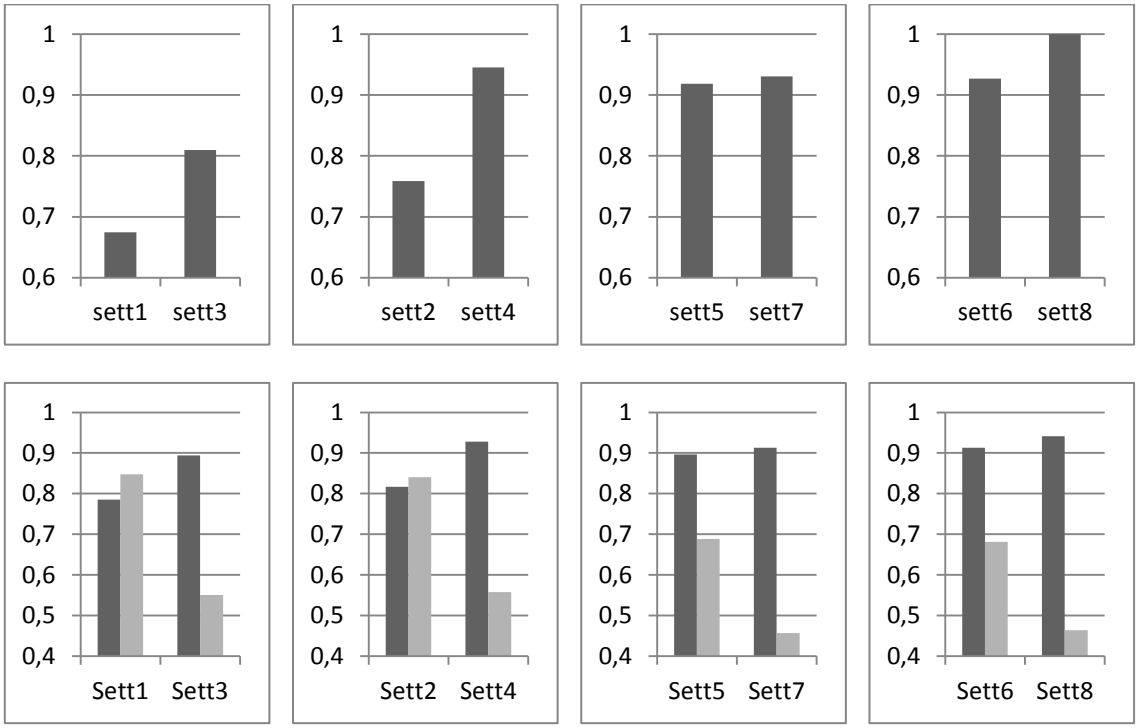
CR:	80	
RL:	6	
ND:	4	
IP:	0,5	0,9

Figura 15: Análise dos resultados da variável IP

Um valor baixo para IP sinaliza para o SWLE que sejam extraídas listas mesmo que não tenha sido encontrado um padrão de caracteres separadores que descreva um número maior de registros. Isso faz com que os esquemas coletados possuam maiores chances de não apresentar uma relação semântica entre cada coluna da matriz, ou seja, os atributos implícitos não representam o mesmo tipo de entidade. Por outro lado, um valor alto para IP, pelos mesmos motivos, pode impedir que listas que de fato possuam conteúdo estruturado sejam ignoradas pelo extrator.

Conforme a análise das configurações pareadas, pode-se concluir que a precisão aumenta conforme aumentamos o valor de IP. Porém, verifica-se que esse aumento, de acordo com o resultado da revocação no segundo experimento, leva a uma queda no valor de revocação para todas as comparações.

5.4.3. Número Mínimo de Registro por Lista (RL) e Número de Derivações (ND)



CR:	160	
IP:	0,5	
RL:	3	6
ND:	2	4

CR:	80	
IP:	0,5	
RL:	3	6
ND:	2	4

CR:	160	
IP:	0,9	
RL:	3	6
ND:	2	4

CR:	80	
IP:	0,9	
RL:	3	6
ND:	2	4

Figura 16: Análise dos resultados das variáveis RL e ND

Como essas duas variáveis foram testadas em conjunto, não podemos concluir nada sobre elas isoladamente. Entretanto pode-se verificar que, quanto maior for o número mínimo de registros e maior for o número de derivações, maior será a precisão da extração.

Com valores baixos para RL e ND, a chance de que sejam extraídos dados não estruturados é maior. Mesmo com um IP alto e um CR baixo, conforme a comparação das configurações 6 e 8, podemos perceber um aumento na precisão quando atribuímos valores maiores para essas duas variáveis.

Ainda, quando temos listas estruturadas com 5 ou menos registros, elas não são coletadas quando o RL for igual a 6. Isso fez com que os valores de revocação ficassem mais baixos em todas as comparações pareadas.

5.4.4. Considerações sobre os Experimentos

Os dois primeiros *settings*, nos quais foram definidos valores mais liberais para suas variáveis, apresentam uma revocação bem superior às demais, apesar de uma leve queda no valor de precisão nos dois experimentos realizados. Mesmo assim, pode-se afirmar que foram os settings com melhores resultados gerais, pois apresentaram um melhor equilíbrio entre precisão e revocação, com valores em torno de 0,8 para ambas as métricas.

Deixando o percentual de identidade predominante alto, temos os *settings* 5 e 6, que apresentaram uma precisão quase tão alta quanto as configurações mais conservadoras, mas com uma revocação de valor médio, comparando com os demais *settings*.

Já os *settings* 3, 4, 7 e 8, apesar de apresentarem um valor de precisão levemente superior aos demais, tiveram índices de revocação muito inferiores, deixando de coletar quase a metade das listas relevantes das páginas selecionadas no segundo experimento.

6. Conclusão

Este trabalho apresenta o SWLE, um algoritmo para a extração de dados estruturados a partir de listas encontradas em páginas da web. Esse tipo de extração envolve algumas problemáticas conhecidas na literatura como a falta de delimitadores explícitos e ausência de rótulos para os atributos dos registros.

Diferentemente de trabalhos relacionados, o algoritmo proposto é considerado mais simples, pois não utiliza bases de conhecimento como suporte, nem vincula a extração a um mecanismo de busca baseado em uma consulta de entrada. As heurísticas utilizadas neste trabalho dependem unicamente da presença e repetição de caracteres especiais entre os registros da lista candidata. Além disso, as listas são extraídas uma única vez, no momento da visita a cada página.

Mesmo com uma abordagem menos complexa em relação aos trabalhos relacionados, o algoritmo SWLE apresentou bons resultados em termos de precisão e revocação na extração de dados estruturados de listas na Web, chegando a alcançar valores superiores a 80% para ambas as métricas na melhor configuração.

6.1. Trabalhos Futuros

Duas evoluções foram identificadas no algoritmo que podem melhorar os resultados obtidos. A primeira delas diz respeito à unificação de listas adjacentes que apresentam a mesma identidade padrão. A segunda é a resolução do problema identificado no segundo experimento em que listas que possuem seus registros separados por tags
 e que possuem tags internas, não são extraídas corretamente.

Referências Bibliográficas

MELLO, Ronaldo: **PLATINUM: Plataforma para Busca Integrada a Fontes de Dados na Web e na Nuvem**. Disponível em http://www.inf.ufsc.br/~Ronaldo/Platinum. Último acesso em 29/10/2015

ELMELEEGY, Hazem; MADHAVAN, Jayant; HALEVY, Alon: **Harvesting relational tables from lists on the web**. VLDB Journal, v. 20, n. 2, 2011

MACHANAVAJJHALA, Ashwin; IYER, Arun; BOHANNON, Philip; MERUGU, Srujana: **Collective Extraction from Heterogeneous Web Lists.** ACM, 2011.

GUPTA, Rahul; SARAWAGI, Sunita: **Answering table augmentation queries from unstructured lists on the web**. Journal Proceedings of the VLDB Endowment, 2009.

CAFARELLA, Michael John: **Extracting and Managing Structured Web Data**. 2009. Disponível em http://turing.cs.washington.edu/papers/cafarella_thesis.pdf

BERNERS-LEE, Tim: **The Semantic Web**. 2009. Disponível em http://www.scientificamerican.com/article.cfm?id=the-semantic-web

LUCENE, Apache. Disponível em https://lucene.apache.org

JSON. Disponível em http://www.json.org/js.html