UNIVERSIDADE FEDERAL DA GRANDE DOURADOS

Jeovano de Oliveira Coutinho

Captura e Extração de Estrutura e Sentenças de Artigos Técnicos Disponíveis na *Web*

DOURADOS – MS

2019

Jeovano de Oliveira Coutinho

Captura e Extração de Estrutura e Sentenças de Artigos Técnicos Disponíveis na *Web*

Trabalho de Conclusão de Curso de graduação apresentado para obtenção do título de Bacharel em Sistemas de Informação pela Faculdade de Ciências Exatas e Tecnologia da Universidade Federal da Grande Dourados.
Orientador: Prof.° Dr. Joinvile Batista Junior

DOURADOS – MS

2019

Jeovano de Oliveira Coutinho

Captura e Extração de Estrutura e Sentenças de Artigos Técnicos

Disponíveis na *Web*

Trabalho de Conclusão de Curso aprovado como requisito para obtenção do título de Bacharel em Sistemas de Informação na Universidade Federal da Grande Dourados, pela comissão formada por:

Orientador Prof. Dr. Joinvile Batista Junior
FACET – UFGD

Profª. Dr. Valguima Victoria Viana Aguiar Odakura
FACET – UFGD

Prof. Me. Carla Adriana Brarvinski
FACET – UFGD

Dourados, 22 de novembro de 2019.

Dados Internacionais de Catalogação na Publicação (CIP).

C871c Coutinho, Jeovano De Oliveira

Captura e extração de estrutura e sentenças de artigos técnicos disponíveis na web [recurso eletrônico] / Jeovano De Oliveira Coutinho. -- 2019.

Arquivo em formato pdf.

Orientador: Joinvile Batista Junior.

TCC (Graduação em Sistemas de Informação)-Universidade Federal da Grande Dourados, 2019.

Disponível no Repositório Institucional da UFGD em:
https://portal.ufgd.edu.br/setor/biblioteca/repositorio

1. Extração de sentenças automatizada. 2. Processamento de Linguagem Natural. 3. Web Scrapping. I. Batista Junior, Joinvile . II. Título.

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

RESUMO

Este trabalho apresenta uma solução automatizada para extração de sentenças e estruturas de artigos técnicos disponíveis na *web* no formato PDF através de uma aplicação com interface gráfica amigável. A aplicação contém um *Web Scrapper* para identificação e realização de *download* dos artigos técnicos da web. No processo de extração de estruturas e sentenças, são utilizadas heurísticas junto a expressões regulares para identificações de conteúdos presentes em artigos técnicos como parágrafos, seções, notas de rodapé, figuras, lista de itens, e referências. O resultado final é um arquivo XML descrevendo a estrutura extraída dos artigos técnicos processados. As sentenças obtidas a partir da estrutura do artigo são utilizadas como entrada para várias atividades de processamento de linguagem natural.

ABSTRACT

This work presents an automated solution for structures and sentence extraction of technical articles available on the web in PDF format through an application with a friendly user interface. The application has a web scrapper for identification and download of the technical articles from the web. In the structures and sentence extraction process, heuristics are used with regular expressions for contents identifying present on technical articles as paragraphs, sections, footers, figures, items list, and references. The final result is an XML file describing the extracted structure from the processed technical articles. The obtained sentences from the structure of technical articles are used as input to many natural language processing activities.

Palavras-Chave: Extração de sentenças automatizada, Processamento de Linguagem Natural, *Web Scrapping.*

SUMÁRIO

LISTA DE FIGURAS

1. Introdução

Nesse capítulo será justificada a relevância do trabalho e descritos o objetivo, a relevância e conteúdo do trabalho.

1.1. Motivação, Oportunidades e Relevância

Este trabalho é uma continuidade de outros projetos executados pelo orientador deste artigo no contexto de iniciação científica, sendo eles:

- Desenvolvimento de Protótipo para Detecção de Sentenças de Corpus para Popular Ontologia Núcleo – ENEPEX 2014.
- Detecção Automática de Sentenças de Artigos Técnicos para Extensão de Ontologia Núcleo – ENEPEX 2015.

As duas propostas são baseadas em processamento espacial, a partir de um txt gerado pelo Foxit Reader. No arquivo gerado não há separação clara entre as colunas, e entre os vários componentes do artigo técnico: texto de parágrafos, figuras, notas de rodapé, títulos de seção, etc.

No primeiro trabalho houve as seguintes considerações:

- “A solução proposta, especialmente para a separação entre as colunas das páginas, automatiza um tratamento para uma grande variedade de situações. A caracterização correta das fronteiras entre as colunas esquerda e direita, nas páginas do corpus, não é simples em função das invasões, que os espaços ocupados pelas sentenças das seções dos capítulos sofrem em função da representação dos componentes. Estas invasões de espaço, que não existem originalmente no formato PDF, mas que são geradas na conversão para o formato TXT, caracterizam contornos de fronteira completamente irregulares, entre as colunas esquerda e direita de cada página” (BATISTA; OLIVEIRA, 2014).
- “A solução proposta, para a implementação do protótipo [...], requer a inserção manual, no arquivo com formato TXT do corpus, de marcadores para viabilizar a detecção: (a) da ocupação de colunas dos componentes *Figure*, *Table* e *HowTo_TryIt*; (b) das linhas de término dos componente *Figure* e *Table*, que não coincidem com o final de página que ocupam; e (c) da separação de palavras de sentenças de colunas distintas que não pode ser realizada de forma

automática. Embora a introdução destes marcadores tenha tornado o processo semi-automático, a quantidade de marcadores inseridos foi muito pequena em relação ao texto de entrada" (BATISTA; OLIVEIRA, 2014).

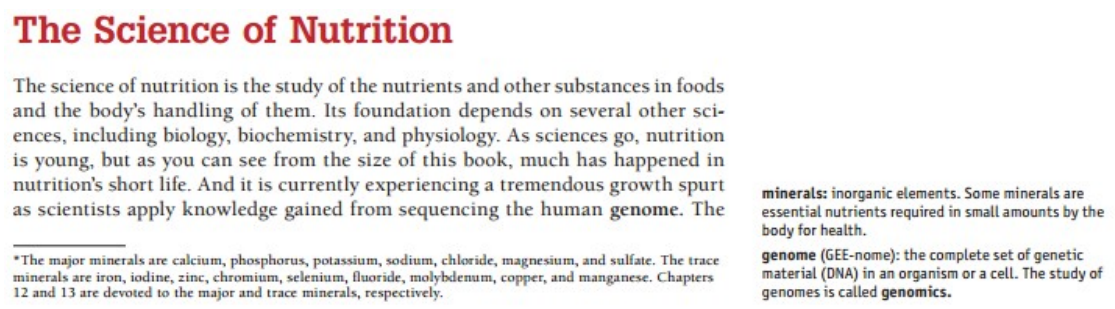
Sobre o segundo trabalho, houve as seguintes pendências:

- "O primeiro desafio, para um trabalho futuro, é o tratamento da situação na qual não exista um intervalo mínimo de dois espaços em branco entre a coluna esquerda e a coluna direita de uma determinada página, o que inviabiliza a separação das colunas esquerda e direita pela solução proposta" (BATISTA; OLIVEIRA, 2015).

- "O segundo desafio é tratar a situação na qual o texto descritivo de uma figura ou tabela que ocupa as duas colunas não bloqueia a caracterização do intervalo separador de colunas, inviabilizando a detecção de que a figura ocupa as duas colunas" (BATISTA; OLIVEIRA, 2015). Esta situação pode ser visualizada nas Figuras 1 e 2.

O problema apresentado no primeiro trabalho é que há necessidade de acrescentar marcadores manualmente, tornando o processo semiautomático. No contexto do segundo trabalho, o problema encontrado é que os textos de duas colunas distintas se entrelaçam de tal forma que é impossível separá-los de forma automática.

O presente trabalho busca solucionar integralmente os problemas relacionados aos trabalhos anteriores, suportando um processo automatizado.



The Science of Nutrition

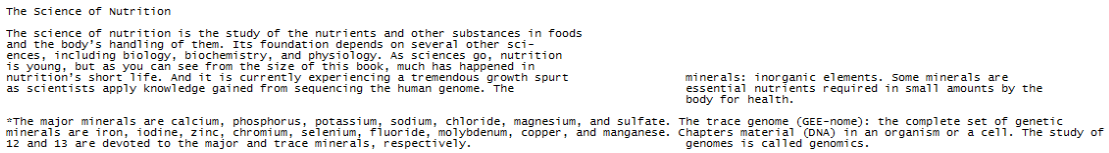
The science of nutrition is the study of the nutrients and other substances in foods and the body's handling of them. Its foundation depends on several other sciences, including biology, biochemistry, and physiology. As sciences go, nutrition is young, but as you can see from the size of this book, much has happened in nutrition's short life. And it is currently experiencing a tremendous growth spurt as scientists apply knowledge gained from sequencing the human **genome**. The

*The major minerals are calcium, phosphorus, potassium, sodium, chloride, magnesium, and sulfate. The trace minerals are iron, iodine, zinc, chromium, selenium, fluoride, molybdenum, copper, and manganese. Chapters 12 and 13 are devoted to the major and trace minerals, respectively.

minerals: inorganic elements. Some minerals are essential nutrients required in small amounts by the body for health.

genome (GEE-nome): the complete set of genetic material (DNA) in an organism or a cell. The study of genomes is called **genomics.**

Figura 1: Trecho no formato PDF (WHITNEY; ROLFES, 2011; p. 33)
com Nota de Rodapé ao Lado de Definições



The Science of Nutrition

The science of nutrition is the study of the nutrients and other substances in foods and the body's handling of them. Its foundation depends on several other sciences, including biology, biochemistry, and physiology. As sciences go, nutrition is young, but as you can see from the size of this book, much has happened in nutrition's short life. And it is currently experiencing a tremendous growth spurt as scientists apply knowledge gained from sequencing the human genome. The

minerals: inorganic elements. Some minerals are essential nutrients required in small amounts by the body for health.

*The major minerals are calcium, phosphorus, potassium, sodium, chloride, magnesium, and sulfate. The trace minerals are iron, iodine, zinc, chromium, selenium, fluoride, molybdenum, copper, and manganese. Chapters 12 and 13 are devoted to the major and trace minerals, respectively.

The trace genome (GEE-nome): the complete set of genetic material (DNA) in an organism or a cell. The study of genomes is called genomics.

Figura 2: Trecho no formato TXT (WHITNEY; ROLFES, 2011; p. 33) com Mistura de Texto entre Nota de Rodapé e Definições

O resultado do presente trabalho é extração de estruturas de artigos disponíveis na *web* no formato PDF. A partir desta estrutura pode-se obter as sentenças associadas às seções de tais artigos. As sentenças correspondem à entrada para várias atividades de processamento de linguagem naturais tais como: Extração de Relações, Entidades e de Eventos utilizando informações internas a uma sentença ou entre sentenças (WADDEN; *et al*, 2019).

Devida a diversidade de estruturas de artigos técnicos disponíveis na *web* foi selecionado um artigo técnico de referência pada o desenvolvimento deste trabalho. A escolha deste artigo se justifica em função dos seguintes itens:

- utilização de seções e subseções;
- utilização de notas de rodapé em sequência a parágrafos concluídos ou não (o texto de alguns parágrafos será concluído na próxima página);
- utilização de figuras e tabelas;
- utilização de expressões matemáticas;
- utilização de referências;
- itens de lista com *bullets* ou numerados.

1.2. Objetivo do Trabalho

Prover um sistema com uma interface amigável para *download* de artigos técnicos disponíveis na *web*, suportando a extração automática destes artigos. A estrutura extraída destes artigos será representada em um arquivo no formato XML.

1.3. Metodologia Adotada

O objetivo geral deste trabalho é fazer o download e a extração automatizada de sentenças de artigos técnicos disponibilizados na *web* através de uma interface amigável. Buscou-se ferramentas de apoio para o desenvolvimento de uma aplicação

que suporte a pesquisa, o download de artigos técnicos, a extração das suas estruturas e a visualização das mesmas em um arquivo XML.

A metodologia adotada pode ser descrita em 4 etapas.

- **Primeira etapa:** Desenvolvimento de uma ferramenta de análise e extração de informações de páginas da *web* para detectar e baixar artigos técnicos disponíveis na *web*.
- **Segunda etapa:** Realizar a conversão do conteúdo PDF para texto com apoio de ferramentas de conversão.
- **Terceira etapa**: Extração automatizada de sentenças a partir da identificação de conteúdos como: seções, parágrafos, figuras, listas, notas de rodapé feitas com apoio de heurísticas e informações de seções providas pela ferramenta de conversão de PDF para texto.
- **Quarta etapa:** Dotar a aplicação de uma representação gráfica adequada (XML).

1.4. Conteúdo do Trabalho

No capítulo 2 são apresentados: comparação com trabalhos existentes; análise e extração de informações de páginas da *web;* inviabilidade de extração da estrutura dos artigos a partir de arquivos PDF. No contexto da impossibilidade de extrair informações diretamente o PDF, foi apresentada uma biblioteca utilizada para conversão de PDF para uma forma intermediária no formato texto, a partir da qual serão aplicadas heurísticas para extração de estrutura do artigo técnico.

No capítulo 3 são relatados os detalhes do projeto como as decisões tomadas com relação ao escopo da aplicação, e as metodologias adotadas para o processo de análise e extração de informações de páginas *web* e de extração de sentenças.

No capítulo 4 o trabalho é concluído apresentando os resultados, dificuldades e trabalhos futuros.

2. Fundamentação Teórica

O presente capítulo inicia apresentando a revisão bibliográfica do tema proposto. Em seguida, apresenta o conceito de análise e extração de informações com a solução de *web scrappers* e, por fim, descreve a dificuldade em extrair informações diretamente de um arquivo PDF.

2.1. Trabalhos relacionados

Em Bui, Fiol e Joannalagadda (2016) desenvolve-se uma ferramenta que categoriza textos de PDF para utilização em sistemas de extração de informações. Utilizando uma biblioteca open-source (PDFBox) para extração de textos dos documentos em PDF e os tipos de fontes no arquivo PDF.

Outra biblioteca (GATE) é utilizada para marcação de conteúdos em alto nível através de esquemas. Os esquemas utilizados com a biblioteca de marcação não são apresentados no artigo. O conteúdo dos textos são classificados em: TITLE (título), ABSTRACT (resumo), BODYTEXT (texto), SEMISTRUCTURE (figuras, tabelas) e METADATA (autores, revistas, jornais publicadores, notas de rodapés, etc). São levadas em consideração as informações de título, resumo, autores e metadados provenientes do MEDLINE® para a classificação dos conteúdos através de comparação direta. O MEDLINE® é uma base de dados específica de artigos técnicos na área de medicina.

Nessa abordagem é feita uma análise que pode utilizar tanto a informação anterior quanto a posterior em relação à posição corrente do texto. Em função disso, em cada contexto é necessário determinar a direção em que a informação será analisada. Para a classificação dos conteúdos, é apresentado um algoritmo estruturado com condições de início, continuação e parada, quantidade de repetições e direção da próxima informação que será analisada. As condições de início, normalmente utilizam dicionários de palavras e/ou posição do texto na página (início, ou fim) para detectar o início de um tipo específico de conteúdo. Exemplificando, para figuras a condição de início é uma comparação com resultado verdadeiro em um padrão comum para figuras, por exemplo: conter na legenda “Figure 1:”. Todas as linhas na sequência serão inicialmente tidas como conteúdo da figura. As condições de continuação são: o conteúdo manter a fonte da linha anterior, não conter a fonte do texto principal, e conter símbolos. A condição de parada é a falha da condição de continuação. As repetições são indefinidas e a direção de análise do conteúdo é dinâmica. Sobre o dinamismo da

direção para realização da análise e classificação do conteúdo o texto só informa que é decidido a partir dos conteúdos ao redor do conteúdo que está sendo analisado, sem maiores detalhes.

Para a distinção entre o conteúdo do texto principal e as notas de rodapés, a solução apresentada utiliza a marcação feita pela biblioteca GATE a partir de um modelo de detecção fornecido que não é especificado no artigo. As notas de rodapés são classificadas como METADATA, sendo, portanto, distinguíveis do texto principal que é classificado como BODYTEXT.

Diferentemente, o sistema desenvolvido no presente trabalho não está limitado ao escopo de artigos técnicos da área da medicina, nem utiliza uma biblioteca para marcação intermediária de conteúdos em alto nível, nem mesmo se vale de informações de tipos de fontes ou início e fim de página.

Em Daudert e Ahmadi (2019) é descrita uma abordagem para extração de sentenças a partir de listas não estruturadas de tokens, obtidas de arquivos em formato PDF, aplicando um classificador de modelos específico para os idiomas inglês e francês. Este classificador utilizou redes neurais recorrentes para determinar o início e fim de sentenças de documentos PDF. Este modelo foi treinado com prospecções financeiras de empresas, portanto o modelo apresentado está limitado a avaliar documentos da área financeira. O modelo ignora todas as segmentações naturais de um texto (pontuações) e avalia o texto extraído do PDF, e através de predição probabilística, obtida pelo treinamento da rede neural, determina o início e o fim de uma sentença. Este trabalho não descreve como trata as estruturas encontradas num artigo como figuras e tabelas, notas de rodapé, itens de lista.

2.2. Análise e extração de informações automatizada com web scrappers

O processo de busca, mapeamento e extração de informações de uma página da web é conhecido como Web Scrapping, que consiste em um processo automatizado que envolve a análise de alguma quantidade de dados para somente obter informações (MITCHELL, 2013, p7). Para a sua realização, é necessário utilizar um HMTL parser, que suporte a análise do conteúdo em HTML de maneira estruturada. A partir dessa análise é gerada a extração de informações da página a partir de padrões identificados em tags HTML e em alguns casos, classes de CSS. CSS (Cascading Styles Sheets) é utilizado para estilização de conteúdos HTML. Para essa finalidade, encontram-se disponíveis algumas alternativas nos repositórios de bibliotecas de código, como por exemplo:

- ***Jsoup**:* É uma biblioteca java para trabalhar com HTML em casos reais. Ela provê uma API muito conveniente para extração e manipulação de dados, utilizando o melhor do DOM, CSS e métodos *jquery-like*. (HEDLEY, 2019).

- ***Jaunt:*** É uma biblioteca java para *web-scrapping*, automação *web* e consultas em JSON (CERVENKA, 2019).

O Processo consiste nos seguintes passos:

- Fornecer a URL da página que será tratada;
- Realizar consultas no conteúdo HTML com as operações provida pelas bibliotecas;
- Extrair o conteúdo;

Neste trabalho, o pacote escolhido foi o *jsoup*, pois oferece o conjunto de recursos necessários para construção do *web scrapper*, sendo os principais: o cliente http, e o *parser* de HTML. Ele contém uma coleção de classes para mapear especificamente uma página em HTML e seus nodos. Os nodos HTML são os blocos da estrutura que compõem um documento HTML.

A instanciação do *jsoup* é muito simples, como se pode ver na Figura 3, definindo a URL destino, um valor em milissegundos de *timeout* para espera máxima da requisição, e a definição de um *user-gent. User-agent* é header http para a identificação da plataforma que está originando a requisição.

```
Document page = Jsoup.connect(url)
        .userAgent(usrAgent)
        .timeout(5000)
        .get();
```

Figura 3: Instanciação *jsoup*

Para a análise do HTML, pode-se utilizar um conjunto de métodos que selecionam determinados nodos do HTML, e filtram o conteúdo indesejado. Para esta tarefa o *jsoup* permite que sejam feitas seleções com métodos similares aos contidos no javascript como document.getElementById() ou document.getElementByClassName(), onde *document* é um nodo HTML. A interface do *jsoup* também fornece um método chamado *select(),* que abstrai as demais seleções, baseando-se na *string* de entrada.

Portanto, *strings* iniciadas com “.” (ponto) serão interpretadas como uma classe CSS, *strings* iniciadas com “#” (cerquilha) serão interpretadas como um ID HTML, retornando então uma coleção com os nodos que satisfaçam a consulta. Tal uso é exemplificado pela Figura 4:

```
String cssClass= ".gs_r.gs_or.gs_scl";
Elements results = page.select(cssClass);
```

Figura 4: Consulta com *jsoup*

2.3. Inviabilidade de extração da estrutura dos artigos em arquivos no formato PDF

Os artigos técnicos encontrados na *web* em sua maioria estão no formato *Portable Document Format* (PDF). Segundo a Adobe®, criadora do formato, PDF é um formato de arquivo confiável para apresentação e troca de arquivos, independente de software, hardware ou sistema operacional (ADOBE, 2019). Atualmente o formato é mantido pela *International Organization for Standardization* (ISO).

A extração de informações de conteúdos de PDF é complexa, o conteúdo interno do PDF encontra-se no formato hexadecimal no qual são utilizas representações específicas para cada detalhe dos vários tipos de conteúdo encontrados em um arquivo PDF como: tipo de fonte, tamanho da fonte, posicionamento das seções de conteúdos, quantidade de colunas, posicionamento de numeração de páginas, notas de rodapé, cabeçalhos, entre outros. Dada a complexidade e a ausência de um padrão na fonte hexadecimal de PDF são necessárias heurísticas para realizar o processo de extração de conteúdo de maneira mais assertiva possível.

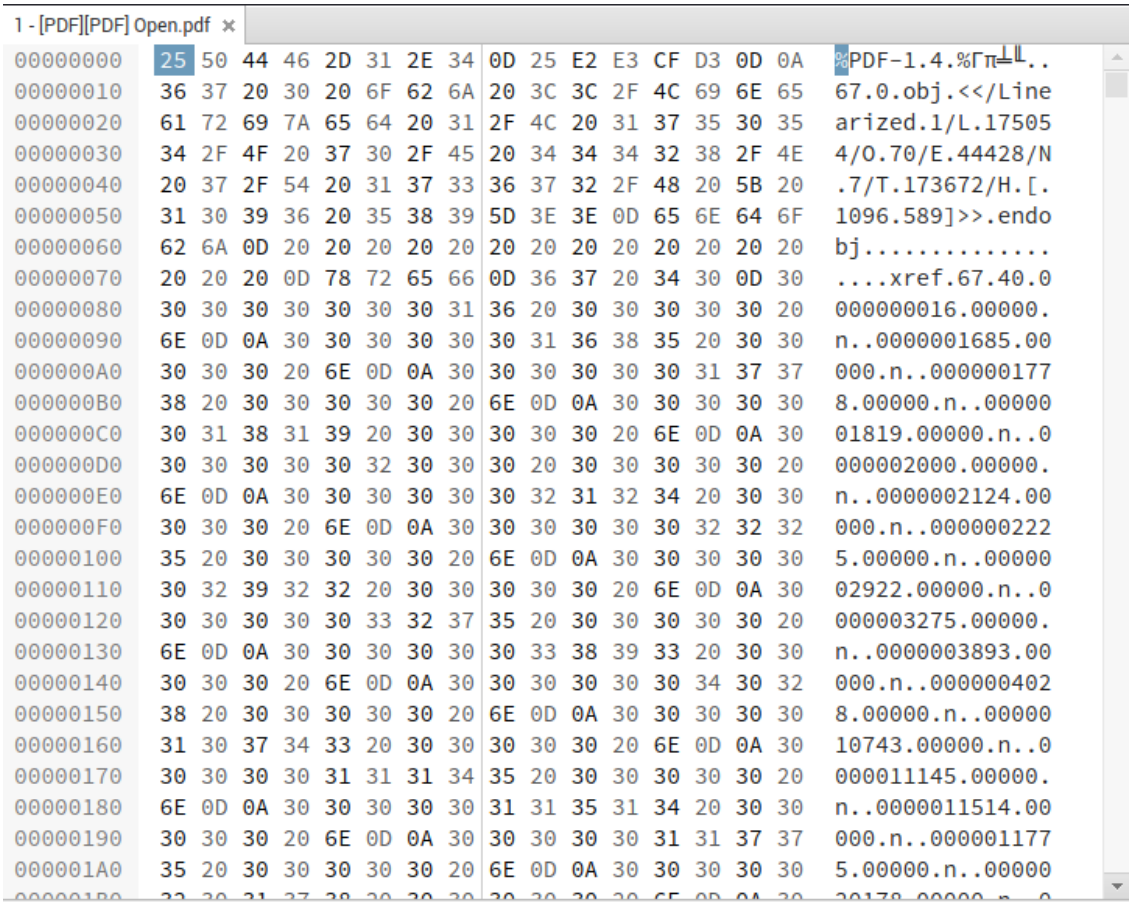


Figura 5: Um artigo técnico PDF representado em formato hexadecimal

PDFBox é uma biblioteca Java *Open Source* para trabalhar com documentos PDF. Esta biblioteca foi utilizada para conversão do formato PDF em um arquivo intermediário no formato texto. Esta biblioteca suporta a criação de novos documentos PDF, manipulação de documentos PDF existentes e a habilidade de extrair conteúdo de documentos. (APACHE, 2019). Além disso é possível acrescentar algumas *Strings* para marcar alguns tipos de seções que são encontradas por meio de heurísticas, para auxiliar a marcação do conteúdo de texto extraído como por exemplo: inicio e fim de parágrafos, início e fim de páginas, separador de palavras (espaço em branco). Salienta-se que as marcações não são perfeitamente assertivas, para suprir tal deficiência é necessário implementar novas heurísticas que trabalhem em conjunto com as marcações de conteúdo customizadas que são plotadas junto ao texto de saída.

Sua distribuição é feita nos formatos de código fonte ou arquivo compilado onde além da disponibilização dos pacotes, disponibiliza uma interface para executar operações por linha de comando.

3. Desenvolvimento do Trabalho Proposto

A ideia inicial deste trabalho era suportar o download de artigos técnicos a partir de links genéricos da *web*. Este objetivo inicial se mostrou muito complexo em função da diversidade das estruturas de HTML nas páginas publicadoras de artigos técnicos. A quantidade de casos a serem tratados seria tão ampla quanto a quantidade de sites publicadores de artigos técnicos. Conduzindo o projeto a limitar o escopo à página do Google *Scholar* para extração do conteúdo. O Google *Scholar* tem a limitação com relação ao tamanho do *abstract*, o que pode limitar o conteúdo, porém possibilita a extração de artigos técnicos de diversas publicadoras.

Além da diversidade de estruturas de HTML, outro problema encontrado ao tentar realizar scrapping em diversas páginas são páginas com conteúdos renderizados dinamicamente com javascript, o que torna necessário o uso de um *headless-browser* para processar o conteúdo dinâmico e então obter o HTML. *Headless-browsers* são ferramentas que realizam todo o processamento de HTML, CSS e javascript como em um navegador comum, porém em formato de biblioteca ou *plugin* para automação de testes *end-to-end*. Comumente eles não possuem interface gráfica. Quando os links de referência ao arquivo PDF estão ou indisponíveis, ou redirecionam para outra página ou devolvem um conteúdo diferente de PDF a aplicação se limita a advertir o usuário que não prosseguirá com o Download. Os detalhes estão descritos nas seções 3.1 e 3.2.

Na etapa de extração do conteúdo do PDF, dada a complexidade já exposta na seção 1.2, optou-se por utilizar o PDFBox para a conversão de PDF para texto, acrescentando as marcações de seções feitas no texto de saída conforme descrito na seção 3.3.

O desenvolvimento, portanto, se resume em: análise e desenvolvimento do *web scrapper*, conversão do arquivo PDF para texto, desenvolvimento do extrator de estruturas e sentenças do conteúdo de artigos técnicos e a representação estruturada em XML.

3.1. A aplicação

A aplicação desenvolvida é composta por um w*eb scrapper*, um conversor de PDF para texto, um extrator de sentenças e uma unidade responsável por gerar a saída em XML. Através de uma interface gráfica, o usuário pode inserir os parâmetros de busca, sendo eles: título do trabalho, autor do trabalho, e um intervalo de anos para os

resultados. As consultas são oriundas do Google *Scholar*, portanto os resultados apresentados são equivalentes a uma pesquisa avançada na plataforma de buscas.

A busca é iniciada ao clicar no botão *search.* Ao realizar uma busca, na interface gráfica é exibida uma lista com o título, autores, ano, e um *checkbox* para seleção de quais artigos técnicos serão processados. Após realizar a busca e seleção dos artigos técnicos, o usuário deve clicar no botão exibido no canto inferior da interface com o título "*Download and extract sentences from PDFs*". *A*pós o clique será realizado o download, a conversão do texto, a extração de sentenças, a geração de um arquivo XML com a estrutura do arquivo e, por fim, a exibição de janela com o XML gerado. Em caso de indisponibilidade no link de download extraído no processo de *scrapping* ou um conteúdo que não seja PDF será exibido um aviso com o detalhe da falha.

A interface gráfica foi construída utilizando os componentes disponíveis no pacote *java swing.* As Figuras 7 e 8 exibem a interface gráfica da aplicação.

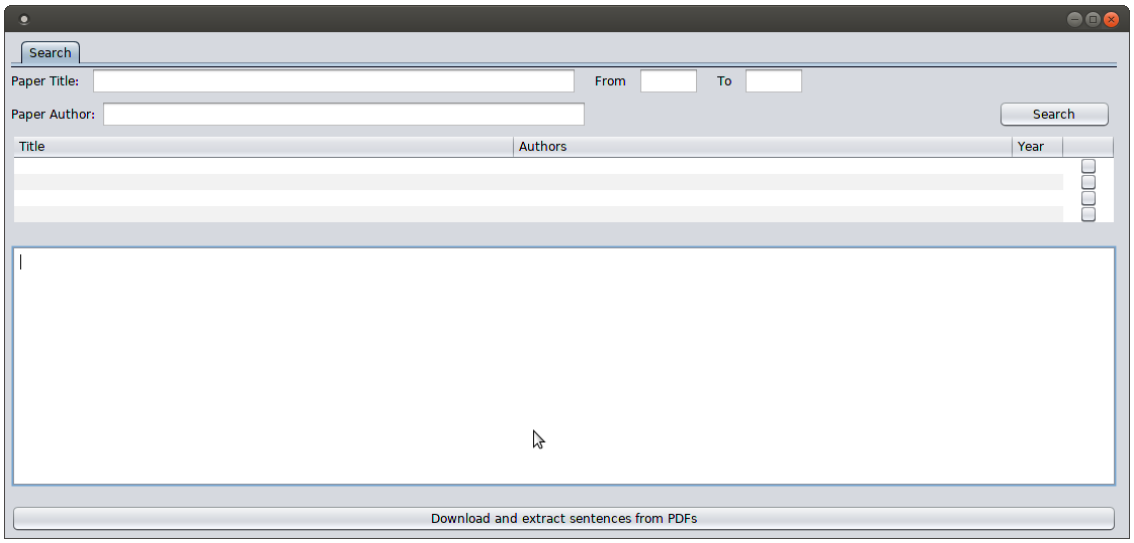


Figura 6: Tela principal da aplicação para pesquisa e *download***.**

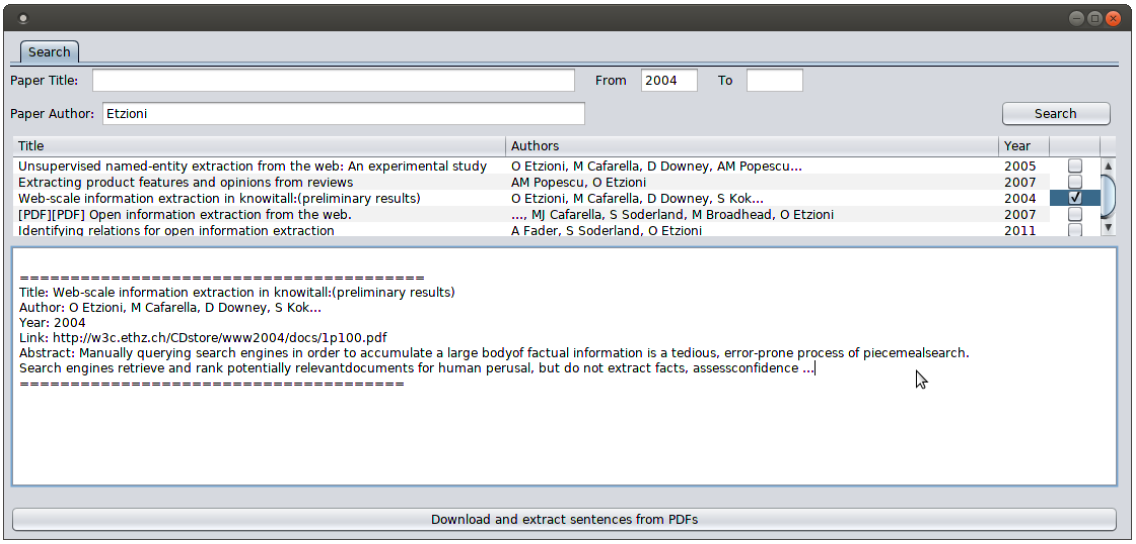


Figura 7: Tela principal da aplicação após pesquisa realizada**.**

3.2. *Web Scrapper*

O processo de análise e extração de informações da página do Google *Scholar* foi feito a partir de identificação de padrões e elementos na estrutura da página para: extração de título, autores, ano de publicação, *abstract* e URL para download do artigo ténico.

O primeiro passo é analisar a composição da URL da página que se deseja extrair as informações. No Google *Scholar* os dados sobre título, autor, ano início e ano fim, e a paginação dos resultados estão na URL no campo de *query string.* A *query-string* é uma área de uma URL onde parâmetros são fornecidos para uma consulta. Adicionalmente, é necessário informar a identificação a plataforma de onde a requisição está sendo feita. Alguns serviços costumam tratar, como uma ameaça, as requisições sem esta informação.

Para que a paginação possa ser automatizada, é necessário implementar um atraso para a próxima requisição. O Google *Scholar* ao detectar que várias requisições são originadas de uma determinada origem num intervalo de tempo muito curto, trata essa atividade como uma automação e então aciona seus tratamentos para este cenário. Apresentando validações em formato de desafios que somente um a pessoa utilizando um navegador, poderia solucionar.

A presente aplicação não implementa o avanço nas páginas de resultados através da interface gráfica.

As páginas da *web* possuem, em geral, classes de CSS padronizadas associadas para cada tipo de elemento (*tag*). Desta forma, a partir do nome da classe CSS é possível identificar o conteúdo descrito no HTML.

Utilizando-se a função de seleção provida pelo *jsoup*, descrita na seção 1.1, pode-se consultar elementos da página HTML e extrair o seu texto ou o valor de seus atributos. No caso do elemento que contém referência para o arquivo PDF onde são extraídos ambos. Inicialmente as seções da página que não contêm as informações objetivas como as barras de navegação são descartadas.

O Google *Scholar* retorna como resultado citações, livros, páginas HTML e artigos técnicos no formato PDF. A Figura 8 exibe um resultado que ilustra a variedade de casos que podem ocorrer em uma busca. Dado que o foco do presente trabalho são os conteúdos em PDF, são descartados os resultados identificáveis como não PDF, sempre que em seus títulos ou no elemento de referência direta para o documento for encontrada uma *tag* informando um tipo de conteúdo distinto do PDF, como por exemplo: [BOOK] e [HTML].

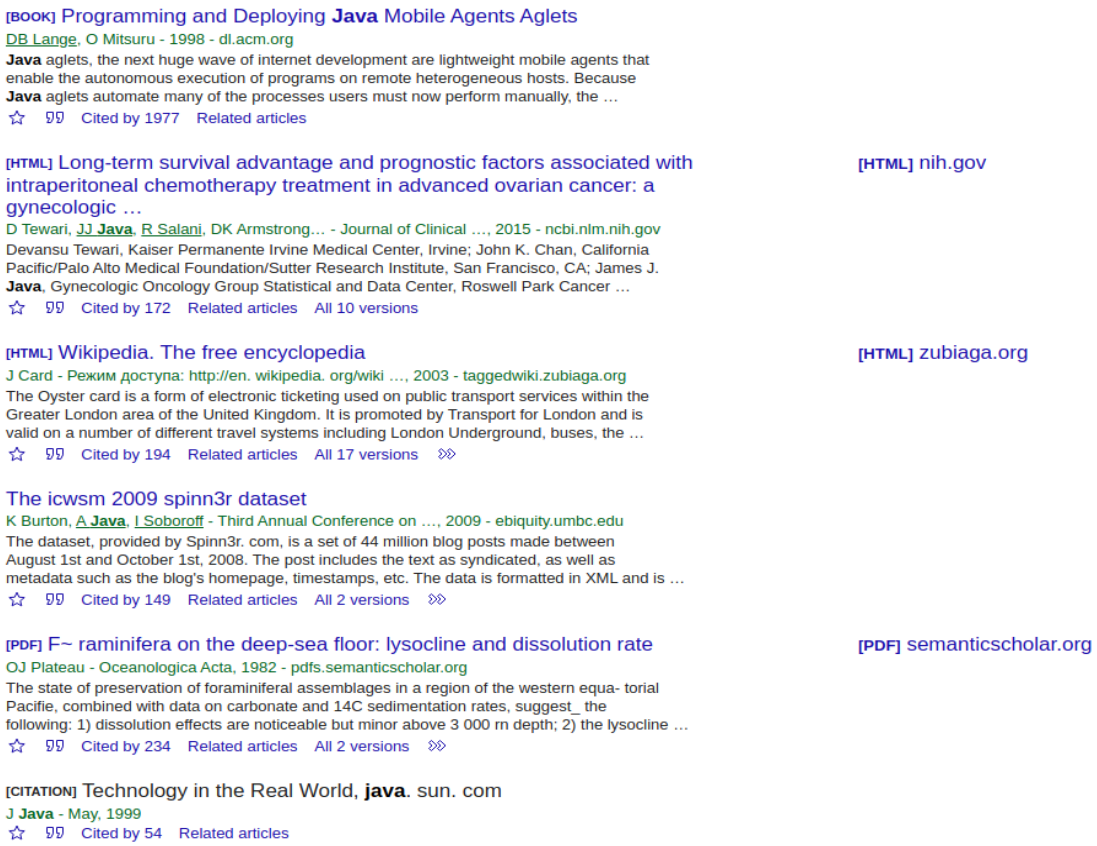


Figura 8: Um resultado de uma pesquisa convencional através do Google *Scholar***.**

Após extrair os resultados, um objeto contendo título, nome do autor, ano de publicação e link de acesso para um artigo técnico é construído e então a coleção de artigos é renderizada na interface gráfica.

Para a realização do *download* de um artigo técnico, é esperado que o retorno da requisição tenha o *header* http *content-type* com valor *application/pdf* e que a requisição não retorne um *status code* diferente de 200. Uma requisição http com *status code* igual a 200 significa uma requisição bem-sucedida. Quando um endereço precisa realizar um redirecionamento, o *status code* retornado é 301. Dado que a aplicação não lida com redirecionamentos, links com *status code* diferente de 200 serão descartados. Os parâmetros retornados em cada requisição, como por exemplo *content-type e status code*, podem ser obtidos através da utilização do *jsoup*. Quando o *download* é bem-sucedido, o arquivo é guardado dentro do diretório interno da aplicação.

3.3. Conversão de PDF para texto

A conversão nativa oferecida pelo PDFBox apresenta algumas limitações, sendo elas identificadas com uma extração simples como exibida na Figura 9. O conteúdo exibido na linha 147 é um fim de parágrafo, seguido por uma seção de notas de rodapés e uma nova seção, contudo não se identifica nenhum tipo de informação explícita de início e término de seções.

```
140 KNOWITALL was inspired, in part, by the WebKB project [6, 7]
141 and its motivation. However, the two projects rely on a different ar-
142 chitecture and very different learning techniques. Most important,
143 WebKB relies on supervised learning methods that take as input la-
144 beled hypertext regions, whereas KNOWITALL employs unsuper-
145 vised learning methods that extract facts by using search engines to
146 home in on easy-to-understand sentences scattered throughout the
147 Web.
148 1froogle.google.com
149 2www.flipdog.com
150 3http://www.eliyon.com
151 2. KNOWITALL
152 KNOWITALL is an autonomous system that extracts facts, con-
153 cepts, and relationships from the web. KNOWITALL is seeded with
154 an extensible ontology and a small number of generic rule tem-
155 plates from which it creates text extraction rules for each class
156 and relation in its ontology The system relies on a domain- and
```

Figura 9: Saída PDFBox padrão de conversão**.**

Utilizando as funcionalidades providas pelo *PDFBox* pode-se fornecer algum tipo de *string* que marque identifique início de término de seções. Na Figura 10 é

possível observar a marcação de início (@STARTPARAGRAPH@) e fim de parágrafo (@ENDPARAGRAPH@). No entanto, a separação realizada pelo PDFBox pode, em alguns casos, misturar elementos de interesse da estrutura. O parágrafo identificado entre as linhas 244 e 248 da Figura 10 ilustra a situação na qual o título da seção e a primeira linha do primeiro parágrafo foram incorporados em uma delimitação associada a um parágrafo. Neste caso, é preciso utilizar uma heurística para separar o título da seção do início do próximo parágrafo.

```
227 @STARTPARAGRAPH@
228 KNOWITALL was inspired, in part, by the WebKB project [6, 7]
229 and its motivation. However, the two projects rely on a different ar-
230 chitecture and very different learning techniques. Most important,
231 WebKB relies on supervised learning methods that take as input la-
232 beled hypertext regions, whereas KNOWITALL employs unsuper-
233 vised learning methods that extract facts by using search engines to
234 home in on easy-to-understand sentences scattered throughout the
235 Web.
236
237 @ENDPARAGRAPH@
238 @STARTPARAGRAPH@
239 1froogle.google.com
240 2www.flipdog.com
241 3http://www.eliyon.com
242
243 @ENDPARAGRAPH@
244 @STARTPARAGRAPH@
245 2. KNOWITALL
246 KNOWITALL is an autonomous system that extracts facts, con-
247
248 @ENDPARAGRAPH@
249 @STARTPARAGRAPH@
250 cepts, and relationships from the web. KNOWITALL is seeded with
251 an extensible ontology and a small number of generic rule tem-
252 plates from which it creates text extraction rules for each class
253 and relation in its ontology The system relies on a domain- and
254 language-independent architecture to populate the ontology with
255 specific facts and relations. KNOWITALL is designed to support
256 scalability and high throughput. Each KNOWITALL module runs
257 as a thread and communication between modules is accomplished
258 by asynchronous message passing.
```

Figura 10: Saída PDFBox com identificação de início e término de seções.

Também é importante ressaltar que os padrões saída de um arquivo PDF, como as marcações de início e fim de parágrafo, não são iguais quando comparadas com a saída de outro arquivo PDF, como é evidenciado na Figura 11.

```
236 @STARTPARAGRAPH@
237 2 Open IE in TEXTRUNNER
238
239 @ENDPARAGRAPH@
240 @STARTPARAGRAPH@
241 This section describes TEXTRUNNER's architecture focus-
242 ing on its novel components, and then considers how
243 TEXTRUNNER addresses each of the challenges outlined in
244 Section 1. TEXTRUNNER's sole input is a corpus and its out-
245 put is a set of extractions that are efficiently indexed to sup-
246 port exploration via user queries.
```

Figura 11: Saída PDF com separação de título de seção e próxima linha.

Os marcadores selecionados para indicar início e fim de seções foram "@STARTPARAGRAPH@" e "@ENDPARAGRAPH@", respectivamente.

O processo se resume então, em a aplicação capturar todos os PDFs selecionados na interface gráfica e transferidos com sucesso da internet e executar o processo de conversão aplicando as marcações de seções especificadas na implementação da aplicação.

3.4. Extração de sentenças

Com o conteúdo em formato texto e com as devidas marcações de início e fim de seções o processo de extração de sentenças pode ser iniciado.

Visando uma implementação abrangente para um número elevado de artigos técnicos, a seleção de um artigo técnico de referência deve considerar a existência de várias situações que possam ocasionar dificuldades para a extração dos diferentes elementos da estrutura do artigo. Os elementos de interesse para a extração da estrutura dos artigos são: figuras e tabelas, notas de rodapé, lista de itens numeradas ou com *bullets*, *layout* de múltiplas colunas, títulos de seção, referências e separação de parágrafos. As figuras podem estar posicionadas em diversos lugares, como início de página, troca de coluna, ou mesmo após outra figura, e possuir formatos distintos podendo ser uma figura totalmente textual (como algoritmos), mescla entre conteúdo gráfico e textual (gráficos numéricos) ou inteiramente gráfica. O artigo técnico foi selecionado por conter praticamente todos os casos supracitados, com exceção das

figuras inteiramente gráficas, as quais são completamente descartadas no processo de conversão de PDF para texto.

O ponto de partida para a elaboração das heurísticas foi a estrutura gerada pela conversão do formato PDF para texto a partir da biblioteca PDFBox. As principais estruturas que serão extraídas serão comentadas a seguir:

O primeiro elemento de interesse, para a geração da estrutura dos artigos técnicos é a nota de rodapé**.** Inicia-se com um número imediatamente seguido pelo início do texto. O fim de uma nota de rodapé é um ponto final seguindo de uma quebra de linha seguido por uma nova nota de rodapé ou um @ENDPARAGRAPH@, ou uma simples quebra de linha para quando a nota de rodapé for uma URL. Salienta-se que não se pode usar sempre o marcador de início de seção como condicionante para o início de uma seção de rodapés por existir casos em que uma seção de rodapés não foi iniciada por um @STARTPARAGRAPH@. Além disso, esclarece-se que esta heurística não soluciona o problema de uma nota de rodapé que está numa coluna ou página, e se encerra em outra coluna ou página.

O segundo elemento de interesse é a representação dos itens de lista. Estes conteúdos se iniciam com *bullets*, ou números seguidos de pontuação. Seu término é indicado por um ponto final, seguido por uma quebra de linha e um @ENDPARAGRAPH@.

O terceiro elemento de interesse é a representação de figuras e tabelas. É o conteúdo mais irregular encontrado na saída da conversão. Tem-se os casos explícitos nas figuras 12, 13 e 14. A heurística aplicada consegue remover as figuras que contêm uma legenda no formato "*figure xx: text content…*", onde xx é o número da figura. O procedimento da heurística é suspeitar de uma linha com tamanho menor do que a média das demais com conteúdo textual, e que não seja o fim de uma seção. Após uma linha ser posta em suspeita, o cursor textual avança uma quantidade de linhas determinadas pela aplicação, caso encontre alguma linha dentro do limite máximo que esteja no padrão da legenda de figura, todo o conteúdo no intervalo da linha de suspeita, até a linha anterior a legenda é removido. Um ponto negativo desta heurística é que caso haja suspeita de uma determinada linha, e por coincidência dentro do intervalo limite haver uma figura que faça com que o cursor alcance uma legenda, todo conteúdo textual será perdido, este cenário foi observado neste artigo técnico, exposto na Figura 13. A Figura 15 ilustra a execução da heurística que suspeitará da linha 508 e alcançará a linha 530 onde encontrará a satisfação da heurística para remover o conteúdo no intervalo.

```
@STARTPARAGRAPH@
KNOWITALL(information focus I, rule templates T)
{
@ENDPARAGRAPH@
@STARTPARAGRAPH@
Set rules R, queries Q, and discriminators D using BootStrap(I,T)
Do until queries in Q are exhausted {
@ENDPARAGRAPH@
@STARTPARAGRAPH@
ExtractionCycle(R, Q, D)
}
@ENDPARAGRAPH@
@STARTPARAGRAPH@
}
@ENDPARAGRAPH@
@STARTPARAGRAPH@
ExtractionCycle(rules R, queries Q, discriminators D)
{
@ENDPARAGRAPH@
@STARTPARAGRAPH@
Set number of downloads for each query in Q
Send queries selected from Q to search engines
For each webpage w returned by search engines {
@ENDPARAGRAPH@
@STARTPARAGRAPH@
Extract fact e from w using the rule associated with the query
Assign probability p to e using Bayesian classifier based on D
Add (e,p) to the Database
@ENDPARAGRAPH@
@STARTPARAGRAPH@
}
}
@ENDPARAGRAPH@
@STARTPARAGRAPH@
BootStrap(information focus I, rule templates T)
{
@ENDPARAGRAPH@
@STARTPARAGRAPH@
R = generate rules from T for each predicate in I
Q = generate queries associated with each rule in R
D = generate discriminators from rules in R, class names in I
Do ExtractionCycle(R, Q, D) without adding facts to Database
S = select extractions with high average PMI score as seeds
Use S to train Bayesian classifier for the discriminators
D = select k best discriminators for each class in I
@ENDPARAGRAPH@
@STARTPARAGRAPH@
}
@ENDPARAGRAPH@
@STARTPARAGRAPH@
Figure 1: High-level pseudocode for KNOWITALL.
```

Figura 12: Figura contendo pseudo-código sem regularidade no conteúdo**.**

```
546 @STARTPARAGRAPH@
547 Extraction Rule:
548
549 @ENDPARAGRAPH@
550 @STARTPARAGRAPH@
551 NP1 "such as" NPList2
552 & head(NP1)="countries"
553 & properNoun(head(each(NPList2)))
554 =>
555 instanceOf(Country,head(each(NPList2)))
556 keywords: "countries such as"
557
558 @ENDPARAGRAPH@
559 @STARTPARAGRAPH@
560 Figure 3: This extraction rule looks for web pages containing
561 the phrase “countries such as”. It extracts any proper nouns
562 immediately after that phrase as instances of Country.
```

Figura 13: Figura textual com padrão regular**.**

```
693 @STARTPARAGRAPH@
694 <rule> ::= <pattern> <constraints> <bindings> <keywords>
695 <pattern> ::= (<context>) (<slot> <context>)* <slot> (<context>)
696 <context> ::= ’"”’ | ’"’ string ’"’
697 <slot> ::= (’NP’<d> | ’NPList’<d> | ’P’<d>)
698 <d> ::= digit
699 <constraints> ::= (’&’ <constr>)*
700 <constr> ::= <phrase> = ’"’ string ’"’ | ’properNoun(’ <phrase> ’)’
701 <phrase> ::= ’NP’<d> | ’P’<d> | ’head(NP’<d> ’)’ |
702
703 @ENDPARAGRAPH@
704 @STARTPARAGRAPH@
705 ’each(NPList’ <d> ’)’ | ’head(each(NPList’ <d> ’))’
706 <bindings> ::= ’=> instanceOf(’ <class> ’,’ <phrase> ’)’ |
707
708 @ENDPARAGRAPH@
709 @STARTPARAGRAPH@
710 ’=> instanceOf(’ <class> ’,’ <phrase> ’)’
711 (’& instanceOf(’ <class> ’,’ <phrase> ’)’)*
712 ’&’ <pred> ’(’ <phrase> (’,’ <phrase>)* ’)’
713
714 @ENDPARAGRAPH@
715 @STARTPARAGRAPH@
716 <class> ::= string
717 <pred> ::= string
718 <keywords> ::= ’Keywords:’ ( ’"’ string ’"’ )*
719
720 @ENDPARAGRAPH@
721 @STARTPARAGRAPH@
722 Figure 5: BNF description of the extraction rule language. An extraction pattern
723 that can be a simple noun phrase (NP), a list of NPs, or an arbitrary phrase (P)
724 match an exact string or to be a proper noun. The “each” operator applies a cons
```

Figura 14: Figura textual com irregularidades.

```
507 @STARTPARAGRAPH@
508 A) “China is a country in Asia.”
509 B) “Garth Brooks is a country singer.”
510
511 @ENDPARAGRAPH@
512 @STARTPARAGRAPH@
513 In sentence A the word “country” is the head of a simple noun
514 phrase, and China is indeed an instance of the class Country. In
515 sentence B, noun phrase analysis can detect that “country” is not
516 the head of a noun phrase, so Garth Brooks won’t be extracted as
517 the name of a country.
518
519 @ENDPARAGRAPH@
520 @STARTPARAGRAPH@
521 Rule Template:
522 NP1 "such as" NPList2
523 & head(NP1)= plural(name(Class1))
524 & properNoun(head(each(NPList2)))
525 =>
526 instanceOf(Class1,head(each(NPList2)))
527
528 @ENDPARAGRAPH@
529 @STARTPARAGRAPH@
530 Figure 2: This generic rule template is instantiated for a partic-
531 ular class in the ontology to create an extraction rule that looks
532 for instances of that class.
```

Figura 15: Cenário de remoção de conteúdo indevida.

O quarto elemento de interesse é o parágrafo**.** O início de um parágrafo é identificado através de uma gestão elementar de contexto. Um contexto de parágrafo se inicia logo após um título, ou uma linha @STARTPARAGRAPH@, sendo este segundo válido desde que a linha atual não seja um título. Todo parágrafo termina com um ponto final, uma quebra de linha seguida por uma linha @ENDPARAGRAPH@.

O quinto elemento de interesse é o título de seção. Esse conteúdo é identificado através da seguinte condição, um texto, precedido por uma linha @STARTPARAGRAPH@, que pode iniciar com numeração, obrigatoriamente tendo o primeiro caractere maiúsculo, com tamanho de 5 a 40 caracteres. A fim de alcançar uma assertividade maior, a aplicação contém uma lógica para identificar se as seções do artigo são numeradas ou não, analisando o formato do título de introdução. Caso haja numeração, todos os títulos deverão ser numerados antes do conteúdo textual, exceto os títulos *Abstract*, *Acknowledgments* e *References*.

A identificação de cada conteúdo é feita através de expressões regulares e de verificações iterativas sobre o conteúdo em texto.

O processamento é feito em dois passos, o primeiro passo é uma identificação e tratamento inicial das sentenças, títulos de seção, notas de rodapés, legenda de figuras, itens de lista. Neste momento, as sentenças identificadas nos parágrafos já são separadas e devidamente posicionadas. Os conteúdos das seções de referências, notas de rodapés, listas de itens e legendas de figuras não são separados em sentenças nesse momento. É necessário garantir a consistência destes elementos que têm um tratamento específico. Os conteúdos de notas de rodapés e de legendas de figuras são depositados em coleções para um processamento posterior a fim de não comprometer a integridade de seu conteúdo com os próximos passos aplicados para as sentenças gerais dos parágrafos.

A segunda etapa consiste em remover as seções vazias oriundas dos primeiros descartes e extrações, e processar os dados que não foram possíveis no primeiro processamento. Os conteúdos como as notas de rodapé, strings das figuras, produzem estas seções vazias. Após a remoção das seções vazias, as seções separadas por elas são unidas e é aplicado o processamento de extração de sentenças.

Após o conteúdo ser extraído, ele é depositado integralmente num novo arquivo de texto, dentro do diretório da aplicação. Cada seção é devidamente separada, e está pronto para a geração do arquivo XML. As notas de rodapés e figuras são processadas e adicionadas após o fim das referências.

3.5. Geração do arquivo XML

Com o novo arquivo de texto contendo o texto processado, o processo de plotagem é feito de maneira iterativa, onde cada linha do texto é testada por expressões regulares. O processo utiliza as marcações de início de parágrafo do processamento de extração de relações, sempre que um novo parágrafo é encontrado no texto, ele se transforma em um novo nó do XML. Linhas em branco são ignoradas pelo processo.

A hierarquia do XML é composta por *section* > *paragraph* > itens, onde os itens podem ser *sentence, footnote, figure, reference, item.* É possível haver subníveis de seção. A presente implementação desce até o terceiro nível a partir de uma numeração inicial, contemplando títulos no formato: “x.x.x.x *title*...”.

A aplicação de contextos de parágrafos é fundamental para que a plotagem seja coerente com o conteúdo no arquivo de texto. Outro contexto utilizado é o de *References,* em que toda linha dentro desse contexto é imediatamente tratada como uma referência, sendo contido dentro de um nó *reference.*

Com o término da plotagem, é exibida a interface conforme a ilustra a Figura 16:

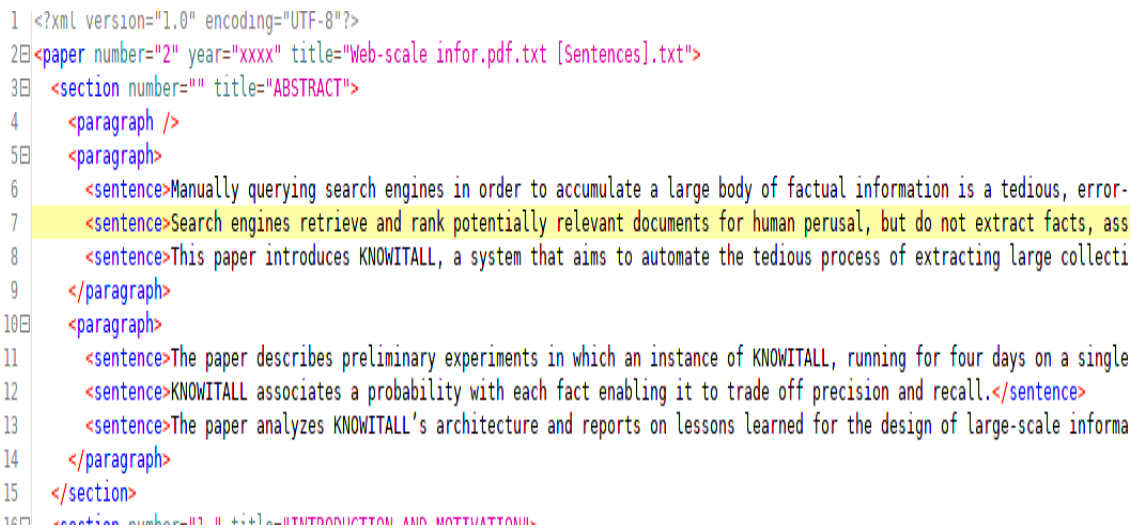


Figura 16: Visualização do XML gerado**.**

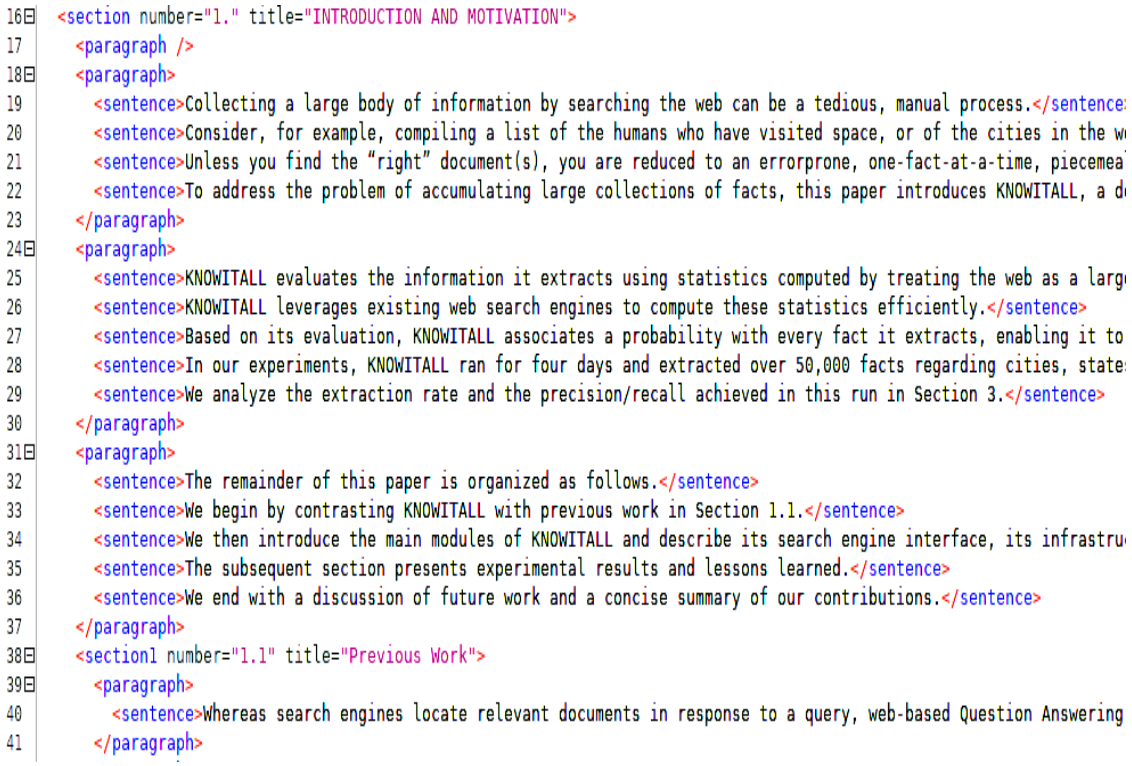


Figura 17: Continuação da visualização do XML gerado **.**

4. Considerações Finais

4.1. Conclusões

No presente trabalho são suportadas diversas funcionalidades. O processo de busca é suportado através de uma interface amigável, a partir qual é possível executar o *download* de artigos disponibilizados pelo *Google Scholar*. O artigo em PDF é convertido para texto, acrescentando-se marcações de início e fim de conteúdos. A partir desta representação textual são aplicadas heurísticas para gerar a estrutura do artigo, que será representada em um arquivo XML.

A automação e a interatividade proporcionada pela ferramenta possibilitam trabalhar com uma escala muito maior de artigos técnicos. Afinal evitam todo pré-processamento manual necessário quando se utiliza outras formas de conversão. Mais ainda, agrega valor no resultado semântico com a estruturação dos dados em XML, garantido que o processamento de linguagem natural abstrativo seja realizado com maior eficiência.

4.2. Dificuldades Encontradas

O processo de extração e classificação de seções de um artigo técnico é complexo devido a falta de regularidade nas marcações que são feitas no processo de conversão e a falta de informações que limitem corretamente os conteúdos. Contudo, vale mencionar que no início do projeto, estas marcações não estavam sendo utilizadas o que deixava o processo de identificação de seções sem assertividade, pois não havia nenhum tipo de informação de seções, logo as trocas de colunas, notas de rodapés, fim de páginas, começo e fim de figuras, listas não eram marcadas e portanto acabavam produzindo um resultado não satisfatório.

4.3. Trabalhos Futuros

Como continuidade do presente estudo, pode-se apontar como possíveis trabalhos futuros:

- Solucionar problema na extração do conteúdo de figuras que não podem ser processados.

- Extração de sentenças de notas de rodapés que se iniciam em uma coluna ou página e se encerram em outra coluna ou página

REFERÊNCIAS

BUI, Duy Duc An; FIOL, Guilherme Del; JONNALAGADDA, Siddhartha. **PDF text classification to leverage information extraction from publication reports.** Journal of Biomedical Informatics, 2016.

DAUDERT, Tobias; AHMADI, Sina. **NUIG at the FinSBD Task: Sentence Boundary Detection for Noisy Financial PDFs in English and French**, Insight Centre for Data Analytics – National University from Ireland, 2019

BATISTA, Joinvile Junior; OLIVEIRA, Paulo Edson Misokami. **Detecção Automática de Sentenças de Artigos Técnicos para Extensão de Ontologia Núcleo**. Relatório Final de Iniciação Científica, 2015.

BATISTA, Joinvile Junior ; OLIVEIRA, Paulo Edson Misokami. **Desenvolvimento de Protótipo para Detecção de Sentenças de Corpus para Popular Ontologia Núcleo**. ENEPEX, 2014.

WHITNEY Ellie; ROLFES Sharon Rady. **Understanding Nutrition**; Twelfth Edition: Wadsworth Cengage Learning, 2011, 1007 p.

WADDEN, David; WENNBERG, Ulme; LUAN, Yi; HAJISHIRZI, Hannaneh; **Entity, Relation, and Event Extraction with Contextualized Span Representations**. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019

MITCHELL, Ryan. **Instant Web Scraping with Java**: Build simple scrapers or vast armies of Java-based bots to untangle and capture the Web. 1st Edition. Birmingham: Packt Publishing, 2013, 63 p.

HEDLEY, Jonathan. **jsoup HTML parser**. Versão 1.12.1. 2019. Disponível em: https://jsoup.org. Acesso em: 1 de nov. 2019.

CERVENKA, Tom. **Jaunt Java Web Scrapping & JSON Querying**. Versão 1.6.0. 2019. Disponível em: https://jaunt-api.com/. Acesso em: 1 de nov. 2019.

ADOBE; WARNOCK, John. **Portable Document Format**. Disponível em: https://acrobat.adobe.com/br/pt/acrobat/about-adobe-pdf.html. Acesso em: 1 de nov. 2019.

APACHE. **PDFBox**. Versão 2.0.17. 2019. Disponível em: https://pdfbox.apache.org/. Acesso em: 1 de nov. 2019.

W3C. **URL:** URI RECOMENDATIONS. 2001. Disponível em: https://www.w3.org/Addressing/URL/4_URI_Recommentations.html. Acesso em: 1 de nov. 2019.

ETZIONI, Oren; CAFARELLA, Michael; DOWNEY, Doug; KOK, Stanley, POPESCU, Ana-Maria; SHAKED, Tal; SODERLAND, Stephen; WELD, Daniel S; YATES, Alexander; **Web-scale information extraction in knowitall: (preliminary results)**, Proceedings of the 13th international conference on World Wide Web, 2004.