Extração Automática de Documentos Médicos da Web para Análise Textual

Inês Fraga Gomes

UMinho | 2019

Inês Fraga Gomes

Extração Automática de Documentos Médicos da Web para Análise Textual

Outubro de 2019

Universidade do Minho
Escola de Engenharia

Inês Fraga Gomes

Extração Automática de Documentos Médicos da Web para Análise Textual

Dissertação de Mestrado
Mestrado Integrado em Engenharia Biomédica
Ramo de Informática Médica

Trabalho efetuado sob a orientação de
Victor Manuel Rodrigues Alves

e supervisionado por
Martinho Gonçalves Antunes Braga

Outubro de 2019

DECLARAÇÃO

Nome: Inês Fraga Gomes

Título dissertação: Extração Automática de Documentos Médicos da Web para Análise Textual

Orientador: Victor Manuel Rodrigues Alves

Ano de conclusão: 2019

Designação do Mestrado: Mestrado Integrado em Engenharia Biomédica

Área de Especialização: Informática Médica

Universidade do Minho, _____/_____/_____

Assinatura: _____

AGRADECIMENTOS

Concluída uma das etapas mais importantes da minha vida, não posso deixar de recordar as pessoas que me acompanharam nesta jornada e partilharam comigo, não só os momentos bons, mas também os mais difíceis.

Em primeiro lugar, quero agradecer ao meu orientador, o Professor Victor Alves, por todo o apoio, dedicação e disponibilidade que mostrou ao longo de a dissertação. Ao meu supervisor de estágio, Martinho Braga, agradeço, por todas as vezes que me orientou e incentivou à conclusão desta etapa, mesmo nos momentos em que motivação estava em falta. À Isabel, que apesar de não estar envolvida diretamente no projeto de dissertação, marcou o meu ano de estágio e em tudo contribuiu para o meu crescimento enquanto profissional.

Às amigas de longa data, Ana, Cris e Diana, um obrigada muito especial, por passados tantos anos se manterem fiéis à nossa amizade e nunca desistirem de mim, mesmo quando não lhes dou toda a atenção que merecem.

À Meli e à Taipas, agradeço por me terem acompanhado neste percurso e por fazerem parte de todas as boas recordações que levo da minha passagem pela universidade. À minha Pica, agradeço por toda a paciência, companheirismo e por estar sempre lá, seja para festejar ou reclamar. Ao Américo, mil obrigadas por me ter acolhido da melhor forma possível, por ser uma inspiração e por continuar a tentar manter-se presente na minha vida, mesmo quando eu não colaboro. À Téni, por ter confiado em mim, por ser um motivo de orgulho e por me permitir seguir o seu crescimento. A todas as amizades que a UM me deu e que sei que vão permanecer, um obrigada.

A toda a minha família, agradeço pelo carinho e apreço, em especial à minha madrinha, que esteve ao meu lado e me apoiou neste ano de algumas mudanças na minha vida.

Por último, aos responsáveis por tudo isto, àqueles que me tornaram na pessoa que sou hoje e pelo amor incondicional que me dão todos os dias, dedico o meu maior agradecimento à minha mãe, pai e irmão. É a eles que tudo isto deve.

Declaração de Integridade

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

Universidade do Minho, _____/_____/_____

Assinatura: _____

Resumo

A literatura científica na biomedicina é um elemento fundamental no processo de obtenção de conhecimento, uma vez que é a maior e mais confiável fonte de informação. Com os avanços tecnológicos e o aumento da competição profissional, o volume e diversidade de documentos médicos científicos tem vindo a aumentar consideravelmente, impedindo que os investigadores acompanhem o crescimento da bibliografia. Para contornar esta situação e reduzir o tempo gasto pelos profissionais na extração dos dados e na revisão da literatura, surgiram os conceitos de *Web Crawling*, *Web Scraping* e Processamento de Linguagem Natural, que permitem, respetivamente, a procura, extração e processamento automático de grandes quantidades de texto, abrangendo uma maior gama de documentos científicos do que os normalmente analisados de forma manual.

O trabalho desenvolvido para a presente dissertação teve como foco principal o rastreamento e recolha de documentos científicos completos, do campo da biomedicina. Como a maioria dos repositórios da web não disponibiliza, gratuitamente, a totalidade de um documento, mas sim apenas o resumo da publicação, foi importante a seleção de uma base de dados adequada. Por este motivo, as páginas web alvo de rastreamento foram restringidas ao domínio dos repositórios da editora BioMed Central, que disponibilizam por completo, milhares de documentos científicos na área da biomedicina.

A arquitetura do sistema desenvolvido divide-se em duas partes principais: fase *online* e a fase *offline*. A primeira inclui a procura e extração dos URLs das páginas candidatas a serem extraídas, a recolha dos campos de texto pretendidos e o seu armazenamento numa base de dados. A segunda fase consiste no tratamento e limpeza dos documentos recolhidos, deixando-os num formato estruturado e válido para ser utilizado como entrada de qualquer sistema de análise de texto. Para a concretização da primeira parte, foram utilizadas a *framework* Scrapy, como base para a construção do *scraper,* e a base de dados de documentos MongoDB, para o armazenamento das publicações científicas recolhidas. Na segunda etapa do processo, ou seja, na aplicação de técnicas de limpeza e padronização dos dados, foram aproveitadas algumas das inúmeras bibliotecas e funcionalidades que a linguagem Python oferece.

Para demonstrar o funcionamento do sistema de extração e tratamento de documentos da área médica, foi estudado o caso prático de recolha de publicações científicas relacionadas com Transtornos Obsessivo Compulsivos. Como resultado de todo o procedimento, foi obtida uma base de dados com quatro coleções de documentos com diferentes níveis de processamento.

Abstract

The scientific literature in biomedicine is a fundamental element in the process of obtaining knowledge, since it is the largest and most reliable source of information. With technological advances and increasing professional competition, the volume and diversity of scientific medical documents increased considerably, preventing researchers from keeping up with the growth of bibliography. To circumvent this situation and reduce the time spent by professionals in data extraction and literature review, the concepts of web crawling, web scraping and natural language processing have emerged, which allow, respectively, the search, extraction and automatic processing of large text, covering a wider range of scientific documents than those normally handled.

The work developed for the present dissertation focused on the crawling and collection of complete scientific documents from the field of biomedicine. As most web repositories do not provide the entire document for free, but only the abstract of the publication, it was important to select an appropriate database. For this reason, the crawled web pages have been restricted to the domain of BioMed Central repositories, which provide thousands of scientific papers in the field of biomedicine.

The system architecture in question is divided into two main parts: the online phase and the offline phase. The first one includes searching and extracting the URLs of the candidate pages to be extracted, collecting the desired text fields and storing them in a database. The second phase is the handling and cleaning of the collected documents, leaving them in a structured and valid format to be used as input to any text analysis system. For the realization of the first part, it was used the Scrapy framework as the basis for the construction of the scraper and the MongoDB document database for storing the collected scientific publications. In the second step of the process, that is, for the application of data cleaning and standardization techniques, some of the numerous libraries and functionalities that the Python language offers are taken advantage of.

In order to demonstrate the operation of the document extraction system, the practical case of collecting scientific publications related to Obsessive Compulsive Disorders was studied. As a result of the entire procedure, a database with four document collections with different processing levels was obtained.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

Notação e Acrónimos

Notação Geral

Ao longo de todo o documento, a notação utilizada seguiu o seguinte padrão:

Texto em *itálico*	Para palavras em língua estrangeira (e.g., Inglês), equações e fórmulas matemáticas. Para dar ênfase a um determinado termo ou expressão e para destacar nomes próprios;
Texto em **negrito**	Para realçar um conceito ou palavra no meio de um parágrafo.

Siglas e Acrónimos

BD Base de Dados

BSON JSON binário *(Binary JSON)*

BMC *BioMed Central*

CQL Linguagem de Consulta Cassandra *(Cassandra Query Language*)

EI Extração de Informação

GUI Interface Gráfica do Utilizador *(Graphical User Interface)*

HTML Linguagem de Marcação de Hipertexto (*HyperText Markup Language*)

IDE Ambiente de Desenvolvimento Integrado (*Integrated Development Environment*)

JSON Notação de Objetos do Javascript *(JavaScript Object Notation)*

JVM Máquina Virtual Java *(Java Vector Machine)*

KDD Descoberta de Conhecimento em Bases de Dados *(Knowledge Discover in Databases)*

MD Mineração de Dados

MT Mineração de Texto

NLTK Conjunto de ferramenta de Linguagem Natural *(Natural Language ToolKit)*

NoSQL Não apenas SQL *(Not Only SQL)*

REN Reconhecimento de Entidades Nomeadas

RI Recuperação de Informação

PLN Processamento de Linguagem Natural

POS Parte do discurso *(Part of Speech)*

SQL Linguagem de Consulta Estruturada (*Structured Query Language*)

TF-IDF Frequência do Termo – Inverso da Frequência nos Documentos *(Term Frequency – Inverse Document Frequency)*

TOC Transtornos Obsessivo Compulsivos

URL Localizador Uniforme de Recursos (*Uniform Resource Locator*)

WWW Rede Mundial de Computadores *(World Wide Web)*

XML Linguagem de Marcação Extensível (*Extensible Markup Language*)

XPath Linguagem XML Path

GLOSSÁRIO

Análise Semântica Estudo do significado de uma linguagem. Inclui a semântica lexical (significado das palavras utilizando morfologias e sintaxe) e a semântica composicional (relações entre palavras ou frases) [1].

Análise Sintática Estudo das regras que controlam a forma como as palavras se dispõem numa frase. Relaciona-se com o modo de combinar diferente palavras (categorizadas como substantivos, adjetivos, verbos etc.) em cláusulas, que, por sua vez, são unidas para formar frases [2].

Array Termo da computação, referente a uma estrutura de dados que armazena um conjunto de elementos que podem ser identificados por um índice ou chave [3].

Base de Dados Coleção de dados organizada, que pode ser armazenada e acedida eletronicamente, através de sistemas computacionais [4].

Big Data Elevado volume de dados não estruturados, presentes em todas as organizações, independentemente do domínio a que pertencem [1].

Bot Abreviatura de Robot. Programa de computador que funciona automaticamente, executando tarefas na Internet para um utilizador ou para outro sistema [5].

Corpus Corpo de um texto. Por norma inclui um elevado número de frases [6].

Corpora Plural de *corpus*. Coleção com grande número de documentos ou dados textuais [1].

Debugging Processo de procurar e resolver problemas que possam existir num programa de computador [7].

Framework Conjunto de técnicas, ferramentas ou conceitos pré-definidos utilizados para resolver um determinado problema. É uma estrutura de trabalho com funções pré estabelecidas que podem ser adaptadas à situação pretendida [8].

Inteligência Artificial Área da ciência da computação dedicada ao estudo e construção de máquinas inteligentes capazes de desempenhar funções que normalmente requerem a inteligência humana [9].

Lemma Forma canónica ou base de uma palavra. Parte principal de um termo, que pode ser obtida através do processo de lematização [1].

Linguagem Natural Linguagem desenvolvida por humanos através da comunicação natural, distinta de linguagens geradas artificialmente, como linguagens de programação de computadores [1].

Machine Learning Subcampo Inteligência Artificial que se foca em atribuir a máquinas a capacidade de aprender automaticamente padrões e relacionamentos, a partir dos dados subjacentes, aperfeiçoando-se com o tempo e sem uma programação explicita ou a codificação de regras específicas [1].

Stem Parte de uma palavra onde normalmente são adicionados afixos. Designação dada ao vocábulo resultante do processo de *stemming* [1].

Token Bloco primitivo de um texto estruturado. Unidades linguísticas como frases, palavras, pontuação ou números [2].

Tuplo Na linguagem Python, um tuplo é uma sequência de valores separados por vírgulas e delimitados por parêntesis curvos [10].

1. Introdução

Pontos Chave:

- Grande parte dos documentos científicos disponibilizados online abordam temas relacionados com a saúde;
- O aumento exponencial de publicações académicas nesta área impossibilita o acompanhamento de toda a literatura por parte dos investigadores;
- O aparecimento de tecnologias de web crawling/scraping veio permitir o rastreamento web e extração do conteúdo das páginas.
- A recolha de documentos científicos das páginas web, em conjunto com a aplicação de métodos de processamento de linguagem natural ao texto recolhido, facilita e automatiza o processo manual dos investigadores.

Nos pontos que se seguem, é apresentada a motivação e os principais objetivos associados ao trabalho desenvolvido. É feita uma exposição do problema e clarificada a estrutura dos capítulos que se seguem.

1.1 CONTEXTO E MOTIVAÇÃO

Um dos tópicos que mais convida à utilização da Internet é a saúde, dado que 87% dos usuários da web pesquisam sobre informações relacionadas com este tema [11]. Como é de esperar, com o aumento da procura por parte dos utilizadores, a quantidade de informações médicas disponibilizadas também cresce, resultando numa sobrecarga de informação. Apesar de nos últimos anos, se verificar um crescimento significativo no número de documentos académicos publicados em qualquer domínio científico, a saúde, e todos os tópicos que esta inclui, destaca-se de entre os vários setores de pesquisa. Temas como a biomedicina, que une os conceitos da biologia com a medicina, são alvo de grande foco na investigação científica e, consequentemente, com um elevado número de pesquisas realizadas e documentos publicados [12, 13].

O aparecimento da Internet veio facilitar o acesso a estes documentos e ao mesmo tempo, acelerar o seu crescimento. Este trata-se de um meio essencial de disponibilização de informação útil, permitindo o armazenamento de grandes quantidades de texto em bibliotecas digitais e repositórios. Este sistema global de redes de computadores inclui todo o conteúdo de *e-mails*, *blogs* e redes sociais, no entanto,

de entre a grande diversidade de texto que a web disponibiliza, a literatura científica consiste na principal fonte de informação de interesse utilizada pelos investigadores [11, 14]. Pode ser empregue no estudo do estado de arte de um determinado tópico, pode ajudar na construção de hipóteses, que posteriormente possam ser validadas de forma experimental, ou pode auxiliar na interpretação dos resultados experimentais [2].

O rápido crescimento da literatura biomédica, a uma taxa de milhares de trabalhos por semana, torna o acompanhamento de todos os desenvolvimentos uma tarefa praticamente impossível. Para gerir de forma automática esta sobrecarga de dados, surgiram ferramentas e técnicas que automatizam e facilitam tanto processo de recolha dos dados, como o seu tratamento e análise. Através da implementação de *Web Scrapers*, entidades capazes de rastrear a web e extrair a informação desejada, é possível a recolha e armazenamento automáticos de documentação científica. No entanto, devido à falta de organização e estrutura da literatura biomédica disponibilizada pelos repositórios online, existe ainda a necessidade de completar este mecanismo de recolha com a utilização de métodos de Processamento de Linguagem Natural. Após todo o processo de extração de dados é fundamental uma limpeza e processamento do texto obtido, para que este se apresente num formato válido para a aplicação de técnicas de análise e extração de conhecimento [15].

1.2 DESCRIÇÃO DO PROBLEMA

Os documentos médicos científicos dizem respeito a todas as publicações, devidamente validadas, que incluem informação relacionada com a área da saúde. Estes documentos, disponibilizados em repositórios online, abordam temas de interesse médico e podem-se apresentar, por exemplo, na forma de artigos, dissertações, relatórios ou monografias. Por sua vez, o conteúdo destas publicações pode incluir mais uma variedade de temas como a análise de doenças, sintomas, tratamentos, constituição química de medicamentos ou estudos genéticos.

Com o elevado número de publicações deste tipo, disponíveis online, é praticamente impossível que os investigadores do setor médico conheçam tudo o que passa no meio. Uma investigação manual pelos documentos num motor de busca e a leitura e análise completa de cada documento podem ser tarefas difíceis e bastante morosas. Para acelerar e facilitar a fase de investigação científica e descoberta de conhecimento na biomedicina, existem diversos métodos e algoritmos de análise textual, em áreas como a mineração de texto ou *machine learning*. No entanto, para o bom funcionamento destes sistemas é

fundamental uma recolha rápida e um tratamento eficaz dos textos que lhes são passados como entrada. Sem uma recolha automatizada dos documentos corretos e uma padronização dos dados, qualquer sistema de análise textual é incapaz de alcançar bons resultados.

1.3 OBJETIVOS

Com o presente trabalho, pretende-se o cumprimento de dois objetivos principais. O primeiro consiste no desenvolvimento de um sistema capaz de rastrear automaticamente repositórios online, onde constam documentos científicos do setor biomédico, pertencentes a um determinado domínio, e extrair o conteúdo textual da documentação encontrada. O segundo objetivo resume-se à criação de uma base de dados apropriada que armazene todos os dados textuais, não estruturados, recolhidos e a aplicação de métodos de tratamento e padronização dos mesmos. Para o cumprimento destes pontos, é importante o estabelecimento de objetivos secundários, que consistem no estudo das várias técnicas e tecnologias envolvidas nestes processos. A implementação da primeira parte do procedimento, ou seja, o desenvolvimento do sistema de procura e recolha automática das publicações, exige o estudo de várias ferramentas que auxiliam nesta ação. Para atingir o último ponto, isto é, para processamento e formatação da informação recolhida, é necessário o estudo e aplicação das diversas metodologias e áreas envolvidas nos métodos de Processamento de Linguagem Natural.

1.4 ESTRUTURA DA DISSERTAÇÃO

O presente documento encontra-se dividido em 4 capítulos, incluindo a secção atual – onde é feita uma contextualização do problema e onde constam as motivações e objetivos da dissertação.

No capítulo que se segue, é feita uma análise introdutória aos conceitos e tecnologias subjacentes ao trabalho. Esta secção fornece ao leitor uma descrição do estado atual da literatura científica, dando ainda um enquadramento do tema escolhido para o estudo de caso – Transtornos Obsessivo Compulsivos. É feita uma demostração do funcionamento da ferramenta de recolha de documentos *(web scraper),* expostas possíveis formas de armazenamento de dados e estudados alguns conceitos envolvidos no Processamento de Linguagem Natural.

No terceiro capítulo, são descritos o planeamento e arquitetura do sistema de recolha, armazenamento e tratamento dos documentos médicos e exposto o modo de implementação e funcionamento.

O capítulo quatro faz a análise de um estudo de caso, onde é demonstrada uma aplicação prática do trabalho desenvolvido e são observados os resultados obtidos.

Por fim, no capítulo cinco, são expostas as principais conclusões do trabalho desenvolvido e sugeridas possíveis áreas de trabalho futuro

2. Conceitos e Tecnologias

Pontos Chave:

- A literatura científica desempenha um papel fundamental no estudo das ciências da vida, servindo como uma potencial fonte para a descoberta de conhecimento na biomedicina.
- No âmbito da saúde, o estudo das perturbações mentais, como é o caso dos Transtornos Obsessivo Compulsivos, tem vindo a ser alvo da atenção dos investigadores, resultando num aumento dos documentos publicados sobre o tema.
- Para acompanhar o crescimento do número de publicações científicas, surgiram tecnologias de web *crawling* e *scraping*, que permitem o rastreamento e recolha automática de dados da web, para fins como a análise textual.
- O armazenamento de dados pode seguir diversos modelos, no entanto, para a reserva de grandes quantidades de textos é vantajosa a utilização de uma base de dados não relacional.
- O processamento de textos em linguagem natural exige que lhes sejam aplicadas técnicas de tratamento, para transformação dos dados num formato estruturado e válido para ser utilizado em sistemas de análise textual.

2.1 LITERATURA CIENTÍFICA

Desde sempre, verifica-se uma constante necessidade dos seres humanos se manterem atualizados acerca dos mais variados temas que a ciência oferece, no entanto, o tempo e esforço que dantes eram necessários para tornar esta tarefa possível excediam o limite aceitável. Para contornar esta situação, surgiu a possibilidade de transmitir conhecimento fidedigno através da publicação de documentos científicos. Este tipo de literatura passou a desempenhar um papel fundamental no estudo das ciências da vida, servindo como uma potencial fonte para a descoberta de conhecimento subjacente. Os mais variados conteúdos de diversos temas passaram a ser publicados em periódicos, ou seja, em revistas científicas (do inglês *Journals*), que eram atualizadas com novos documentos a cada intervalo de tempo [16, 17].

A literatura científica na biomedicina é um elemento fundamental no processo de obtenção de conhecimento, uma vez que é a maior e mais confiável fonte de informação. Com os avanços tecnológicos e o aumento da competição profissional, o volume e diversidade de documentos médicos

científicos tem vindo a aumentar consideravelmente, impedindo que os investigadores acompanhem o crescimento da bibliografia.

Com o aumento da quantidade e diversidade destas publicações, principalmente no âmbito da medicina, o que dantes era escassez de informação, posteriormente passou a sobrecarga. Em resposta a esta crescente taxa de publicações de literatura científica por ano, acabaram por surgir as bibliotecas digitais, que permitiram o armazenamento de grandes volumes de documentos [16].

Atualmente, qualquer investigador na área médica recorre às mais diversas ferramentas disponíveis na web, para obter a documentação científica que vai de encontro às suas necessidades. Existem milhares de repositórios online que armazenam e disponibilizam aos seus utilizadores documentos científicos nas mais variadas áreas. A forma mais comum de comunicar conhecimento em qualquer setor das ciências é através de artigos científicos devidamente certificados, que permitem aos investigadores a troca de resultados das suas pesquisas. Para além dos artigos, a bibliografia científica pode ainda se apresentar na forma de monografias, dissertações, relatórios ou até mesmo livros científicos em formato digital. As bases de dados online de armazenamento destes documentos estão muitas vezes organizadas em temas, como saúde, química, matemática, economia ou educação [18, 19].

Apesar da literatura biomédica estar reservada em repositórios específicos para tal, uma vez que todas as publicações se encontram relacionadas com a medicina ou a biologia, os documentos disponibilizados, quase sempre, abordam tópicos divergentes. Este tipo de publicações literárias tanto pode incluir documentos que se focam na análise de doenças, a suas causas, sintomas e tratamentos, como pode abranger estudos de métodos de gestão de uma instituição hospitalar. Pode ser estudada a evolução tecnológica na saúde, abordados estudos de relações entre genes, proteínas e doenças, ou ser dado destaque à descoberta e desenvolvimento de novos fármacos [18, 19] .

Restringindo a gama de temas que o setor da biomedicina inclui, ao estudo das perturbações mentais, observa-se que os Transtornos Obsessivo Compulsivos têm vindo a ser alvo da atenção dos investigadores e que os documentos sobre esta síndrome têm sofrido um aumento no número de publicações [20].

2.1.1 TRANSTORNOS OBSESSIVO COMPULSIVOS

Um Transtorno Obsessivo-Compulsivo (TOC) é uma síndrome que se carateriza pelos efeitos de obsessão e compulsão que provoca. As obsessões consistem em pensamentos ou impulsos de concretizar uma

determinada tarefa, o que provoca perturbação e ansiedade no indivíduo. As compulsões são a repetição frequente de determinadas ações motoras, a fim de melhorar a ansiedade provocada pela obsessão. Este transtorno neuropsiquiátrico, que afeta cerca de 2-3% da população mundial, surge de forma espontânea na mente das pessoas, levando-as a comportamentos muito meticulosos e precisos que lhes ocupam uma grande percentagem do tempo [21, 22].

O tratamento deste distúrbio assenta principalmente em três metodologias, que podem ser usadas em separado, mas que demonstram maior eficácia quando combinadas, são elas: psicoterapia, farmacoterapia e modulação cerebral. A terapêutica cognitivo-comportamental, que consiste no tratamento da psicoterapia com melhores resultados, visa identificar e alterar padrões de pensamento destrutivos e perturbadores que influenciam negativamente os comportamentos. As intervenções farmacológicas, nomeadamente a utilização de inibidores seletivos da recaptação de serotonina, ajudam no controlo deste distúrbio, no entanto, cerca de 40-60% dos pacientes mostram resistência ao tratamento. A modulação cerebral, que inclui uma estimulação profunda do cérebro, é um método cirúrgico eficaz no tratamento de pacientes resistentes às modalidades anteriores. No entanto, por se tratar de um procedimento invasivo e por não serem conhecidos os seus efeitos a longo prazo, este método causa alguma controvérsia [21].

2.2 RASTREAMENTO E EXTRAÇÃO DE DADOS DA WEB

Desde a descoberta da Internet, nos anos 90, o uso desta tecnologia foi-se expandindo, até se tornar numa necessidade primária para os utilizadores. Os seus benefícios, em áreas de atividade cada vez mais diversas, contribuem para o crescimento do número de utilizadores da Internet [23]. A população atual do mundo é de cerca de 7700 milhões de pessoas, das quais 4500 milhões (58,8%) utilizam a Internet. Segundo o estudo de um instituto de pesquisa de mercado (*e-Marketer*), pelo menos até ao ano de 2017 a China era o país com mais pessoas a usufruir desta tecnologia, tal como se pode observar na Tabela 2.1, que mostra os 5 países do mundo com mais utilizadores e a evolução deste número ao longo de 5 anos [23].

Tabela 2.1 - Número de utilizadores da Internet ao longo de cinco anos, nos países onde este valor é mais elevado [23]

Ordem	**País**	**Número de utilizadores da Internet (em milhões)**				
		2013	*2014*	*2015*	*2016*	*2017*
1	China	620,7	643,6	669,8	700,1	736,2
2	Estados Unidos	246,0	252,9	259,3	264,9	269,7
3	Índia	167,2	215,6	252,3	283,3	313,8
4	Brasil	99,2	107,7	113,7	119,8	123,3
5	Japão	100,0	102,1	103,6	104,5	105,0

A rede mundial de computadores (WWW[1]) consiste num sistema que se baseia na completa autonomia do servidor para disponibilizar as informações presentes na Internet. Os *browsers* utilizam motores de busca que exploram os servidores para encontrar as páginas pretendidas e que são, posteriormente, processadas do lado do cliente. Estes dados armazenados encontram-se organizados num sistema de texto de grandes dimensões, distribuído e não linear, denominado *Hypertext Document*. Nesta estrutura, parte de um documento é definida como hipertexto, ou seja, porções de texto ou imagens que se ligam a outros documentos por *hyperlinks* [24].

2.2.1 WEB CRAWLING

A dificuldade do trabalho de pesquisa tem vindo a diminuir com o aparecimento da era de informação. Desde relatórios técnicos, artigos ou tutoriais, os vários tipos de literatura passaram a ser apresentados sob a forma digital, verificando-se assim, nos últimos anos, uma explosão da informação científica publicada [25]. Este crescimento significativo da informação disponível, e consequentemente, da utilização da web faz com que os motores de busca recorram a *Web Crawlers*, a fim de recolher apenas as páginas pretendidas pelo utilizador e evitar o carregamento de todas as páginas da web [13].

De uma forma geral, o termo *crawler*, ou rastreador, indica a capacidade de um programa navegar por páginas da web de forma autónoma, muitas vezes sem qualquer objetivo final bem definido, explorando continuamente o que um site ou a web tem para oferecer. Os rastreadores da Web são muito usados por motores de busca como o *Google*, para recuperar o conteúdo de um URL, examinar a página em busca de outros links e recuperar os URLs desses links [26].

[1] Do inglês: *World Wide Web*

Um *Web Crawler* consiste num programa ou software que navega na WWW de forma automática e sistemática. Estes *robots* são capazes de, a partir de uma única página Web, percorrer várias outras páginas que se encontrem vinculadas a ela. O objetivo do desenvolvimento destes programas está na recuperação de páginas da Web para que estas sejam armazenadas num repositório local [24].

O processo de rastreamento, ou crawling, representado na Figura 2.1, inicia-se com uma lista de URLs iniciais, fornecidos por um utilizador ou programa. Estes endereços são acedidos pelo rastreador e as páginas correspondentes são analisadas para que todos os *hiperlinks* sejam extraídos e armazenados numa lista de páginas por visitar. Este processo de busca e extração de URLs é repetido, seguindo um conjunto de regras e políticas, até que a lista de *links* esteja vazia ou alguma condição o faça parar [27].

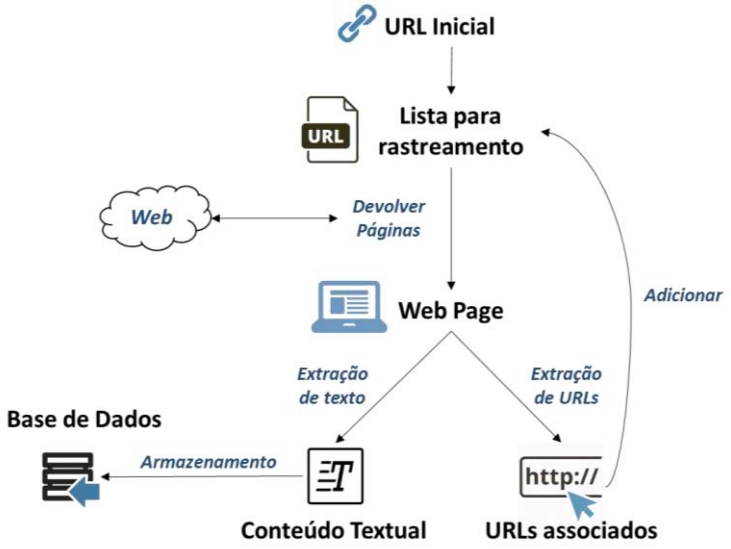


Figura 2.1 - Representação esquemática do processo de rastreamento da web.

A fim de garantir o bom funcionamento deste processo, um *Web Crawler* deve estar preparado para resistir à interceção de alguns servidores web que provocam a paralisação dos rastreadores e respeitar as regras e políticas dos servidores web evitando sobrecargas nos sites [28].

2.2.2 WEB SCRAPING

A diferença entre *Web Crawling* e *Web Scraping* é relativamente vaga e as duas designações são muitas vezes utilizadas para se referir ao mesmo processo, no entanto, alguns autores conseguem apontar divergências entre ambos os processos. Enquanto que um *Web Crawler* se foca na consulta de páginas web, de onde extrai todos os links que pretende visitar, um *Web Scraper* visa a extração do conteúdo dessas páginas [26].

A maior parte dos dados necessários para qualquer processo de análise é disponibilizada através da Internet. No entanto, não é possível a um analista a recolha manual destas grandes quantidades de informação da web, tornando útil a implementação destes mecanismos automáticos [29].

As páginas web são criadas com recurso a linguagens de marcação baseadas em texto, como HTML e XML, e armazenam dados úteis, preparados para ser acedidos por utilizadores humanos e não por programas computacionais. Por este motivo, apareceram os *Web Scrapers*, que se responsabilizam pela extração automática de dados apresentados nos websites [29]. Estas entidades são consideradas uma variante recente dos rastreadores, capazes de realizar uma extração de informações úteis de páginas HTML, a principal ferramenta de formatação de dados na WWW. Este procedimento é implementado através da utilização de qualquer linguagem de programação e anotação semântica [30]. Todo o processo de *Web Scraping*, também conhecido por Extração de dados da Web, pode ser equiparado a um agente que extrai, analisa e organiza dados da web de forma automatizada. Ou seja, a tarefa que um utilizador tem de navegar pela internet e copiar as partes que lhe interessam passa a ser desempenhada por um programa de computador, que executa este procedimento de modo automático e inteligente.

Para além dos autores que utilizam os conceitos de *Crawling* e *Scraping* de forma indistinta e dos que os definem como processos independentes, existem ainda investigadores que assumem o mecanismo de *Web Crawling* como a primeira etapa do processo de *Web Scraping*. Para estes, um *Web Scraper* divide-se em dois componentes principais: um *Web Crawler* e um Extrator de Dados [29, 31]. No presente documento, é adotada esta conceção, ou seja, sempre que se utiliza o conceito de *scraper*, está subjacente que a etapa de *crawling* se encontra incluída.

2.2.3 FERRAMENTAS DE RASTREAMENTO E RECOLHA DE DADOS

Várias são as ferramentas que podem ser utilizadas para automatizar o processo manual de "copiar e colar" grandes quantidades de informação disponibilizada nos websites [32]. Para o desenvolvimento de sistemas de rastreamento da web e recolha de dados podem ser aplicados diferentes rastreadores de código aberto que diferem entre si em caraterísticas como escalabilidade, flexibilidade e desempenho. Nos tópicos que se seguem são descritas algumas das *frameworks* de rastreamento mais utilizadas, sendo dado especial destaque à estrutura *Scrapy*, uma vez que foi a escolhida para o desenvolvimento do trabalho [28].

2.2.3.1 APACHE NUTCH

O *Apache Nutch* é um software de código aberto para rastreamento web. O código que utiliza é integralmente na linguagem Java e apresenta uma arquitetura modular, permitindo que os utilizadores criem plug-ins para análise de informação dos *media*, para recuperação de informação, para consulta ou agrupamento de dados. Trata-se de uma ferramenta bastante vantajosa na criação de rastreadores distribuídos, mas a sua implementação apresenta um elevado grau de dificuldade. Além disso, a documentação disponibilizada encontra-se desatualizada o que dificulta o processo de instalação e utilização [28].

2.2.3.2 HERITRIX

O *Heritrix* consiste num projeto de rastreamento web extensível, de código aberto e com qualidade de armazenamento. Tal como o *Apache Nutch*, também este rastreador é escrito em Java. Para a sua implementação, não é necessária instalação, basta que seja executado numa máquina virtual Java (JVM[2]). No entanto, para além da sua arquitetura bastante complexa, esta ferramenta não permite selecionar os dados para recolha e obriga a que toda a página HTML seja extraída de modo integral sem qualquer filtro de texto [28].

2.2.3.3 SCRAPY

A *framework* Scrapy é uma poderosa estrutura de rastreamento e extração de dados da Web, escrita em Python, na qual os *robots* são definidos como classes herdadas da classe base *Spider*. Esta classe deve incluir a definição de um conjunto de URLs iniciais e o desenvolvimento de pelo menos uma função *parse* chamada em cada iteração. As páginas da Web são analisadas automaticamente e o conteúdo é extraído usando expressões *XPath*. Esta ferramenta, desenvolvida por uma empresa Londrina, permite, para além do rastreamento, a extração do conteúdo de páginas de diferentes fontes da Internet, sendo bastante utilizada por diversas empresas de tecnologia de informação [33].

A primeira versão desta ferramenta foi lançada em junho de 2008 e a última, até à data em outubro de 2019. O Scrapy consiste num sistema integrado que inclui um motor de controlo de fluxo de dados entre componentes (*Engine*), um escalonador para receber os pedidos (*Scheduler*), um mecanismo de

[2] Do inglês: *Java Virtual Machine*.

carregamento das páginas web (*Downloader*) e classes personalizadas desenvolvidas pelos utilizadores para analisar as respostas e extrair os itens pretendidos (*Spiders*) [34].

A Figura 2.2 mostra uma visão geral da arquitetura *Scrapy* com os seus componentes e ilustra o fluxo de dados que ocorre dentro do sistema. Para uma melhor compreensão deste processo, importa descrever cada um dos constituintes da arquitetura:

Engine: Principal componente da arquitetura Scrapy, uma vez que controla todo o processo de rastreamento e extração de páginas web. É o elemento responsável pelo funcionamento de todo o fluxo de dados (pedidos e respostas) entre os restantes componentes do sistema e por acionar eventos quando determinadas ações ocorrem. De um modo geral, este motor mantém o fluxo de transmissão de dados entre os blocos que constituem a arquitetura [35].

Scheduler: Elemento encarregue de controlar o tempo de pedido e resposta do Scrapy. Recebe solicitações do *Engine*, organiza-as por ordem e volta a solicitar o *Engine* para que este faça o pedido ao *Downloader* [35].

Downloader: Responsável por carregar as páginas da web e alimentar o *Engine*, que por sua vez, alimenta o *Spider*. Componente que interage com a Internet e extrai as páginas solicitadas para o Engine [35].

Downloader Middleware: Componente responsável por estabelecer conexão entre o *Engine* e o *Downloader*. Passa pedidos do *Engine* para o *Downloader* e respostas deste para o *Engine* [35].

Spider: Classes personalizadas escritas em Python pelos utilizadores do Scrapy para analisar respostas do *Downloader* e extrair os itens pretendidos ou para enviar solicitações adicionais. Estas classes definem que páginas ou domínios devem ser rastreados e o modo como a extração dos itens deve ser feita. Um *Spider* envia pedidos ao *Engine* para que estes cheguem ao *Downloader*, que por sua vez devolve a resposta para ser analisada [35].

Spider Middleware: Elemento que permite a comunicação entre o *Spider* e o *Engine*. Passa os pedidos do *Spider* para o *Engine* e as respostas deste para o *Spider*, possibilitando a extração de itens [35].

Item Pipeline: Responsável pelo processamento dos itens depois de extraídos pelo *Spider*. Funciona como um filtro onde é selecionada a informação que se pretende armazenar. As tarefas deste elemento podem incluir limpeza dos itens, validação e armazenamento dos itens numa base de dados [35].

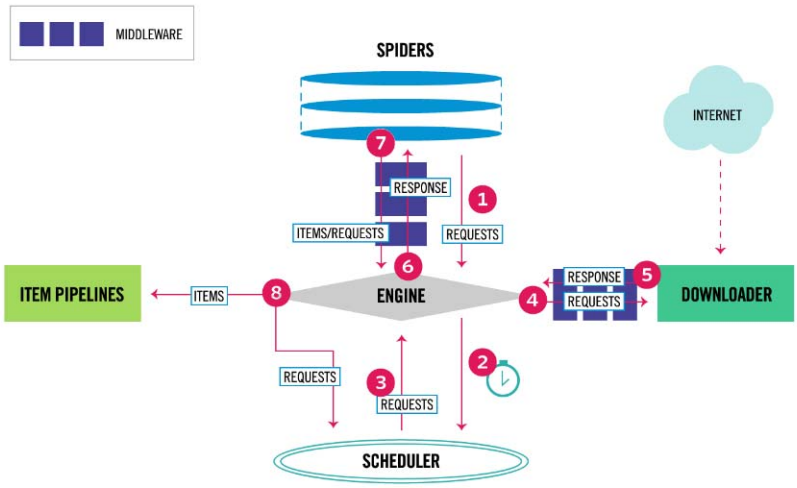


Figura 2.2 - Arquitetura de um sistema *Scrapy* [34].

Todo o processo de rastreamento inicia quando o *Engine* é solicitado pelo *Spider*, que lhe passa os primeiros URLs a serem visitados (1). O *Engine,* por sua vez, agenda o pedido no *Scheduler* e espera a próxima solicitação do *Spider* (2). O *Scheduler* solicita o Engine com o primeiro URL a ser visitado (3), para que este o passe ao *Downloader,* através *Downloader Middleware* (4). O *Downloader* carrega a página web associada ao URL e gera a resposta a devolver ao *Engine*, novamente via *Downloader Middleware* (5). O *Engine* alimenta o *Spider* com a resposta (via *Spider Middleware*) para que esta seja processada e os itens pretendidos extraídos (6). Por sua vez, o *Spider* vai retornar ao *Engine* os dados extraídos e o novo URL a rastrear, novamente através do *Spider Middleware* (7). O *Engine* passa os itens para o *Item Pipeline*, onde são filtrados e armazenados, e o URL para o *Scheduler* que lhe responde com o próximo pedido (8). A partir deste ponto, o processo repete-se até que todos os URLs tenham sido rastreados ou até que alguma condição faça parar o rastreador [34]. Como já referido, todos os elementos da arquitetura *Scrapy* interagem entre si num fluxo organizado e que se encontra esquematizado na Figura 2.2.

Esta ferramenta destaca-se das anteriores por diversos motivos. Para além de ser uma ferramenta capaz de lidar facilmente com a linguagem HTML, que constitui as páginas web, o Scrapy é totalmente escrito na linguagem de programação Python, permitindo que os utilizadores desenvolvam os seus programas com uma linha de código relativamente simples. Esta *framework* apresenta forte comunidade de suporte, que pode ser bastante útil no caso de surgir qualquer problema de execução do *scraper* [33].

Para que qualquer estrutura de rastreamento, desenvolvida sob a *framework* Scrapy, possa tirar vantagem das funcionalidades desta arquitetura, é importante que seja conhecida a organização do

documento HTML. A página web que um utilizador visualiza, após a pesquisa por um endereço, resulta da resposta do servidor à solicitação desse URL. Este servidor devolve um documento HTML ao navegador, que o interpreta e transforma na página observada no ecrã [36].

Para facilitar a extração de texto das páginas HTML, a *framework* Scrapy utiliza *Selectors* que permitem selecionar a parte da estrutura HTML que se pretende recolher através da utilização de expressões XPath ou CSS. Para além de navegar pela hierarquia de *tags* HTML, os seletores também podem observar o conteúdo destes marcadores, o que os torna extremamente úteis no rastreamento de um conteúdo pesado [37]. A linguagem XPath é utilizada para navegar pelos nós de documentos XML, mas também é pode ser bastante útil no acesso às *tags* HTML [34].

Independentemente do tipo de *Scraper* utilizado no processo de recolha e extração de informação da web, todos se focam na transformação de dados não estruturados, normalmente apresentados no formato HTML, em estruturas organizadas e que podem ser armazenadas em diretorias ou bases de dados locais [38].

2.2.4 BASES DE DADOS NOSQL

Mais uma vez, devido ao desenvolvimento da Internet e ao aumento exponencial de dados, surgiu a necessidade da criação de bases de dados para armazenar e processar *Big Data* de forma eficiente. Esta grande quantidade de informação exige que a base de dados apresente um elevado desempenho de leitura e escrita, o que põe em causa a eficácia das tradicionais bases de dados relacionais [39].

Dos vários modelos de dados existentes, o modelo relacional domina desde os anos 80, com implementações como bases de dados Oracle, MySQL e Microsoft SQL. Nestes sistemas, é utilizada a linguagem de consulta SQL[3] para obter e armazenar dados. No entanto, a utilização deste tipo de bases de dados em grandes volumes de informação tem mostrado alguma ineficiência na modelação de dados e várias restrições de escalabilidade nos servidores. O acesso aos dados mostrou-se bastante lento o que degrada a performance de qualquer aplicação que lide com grandes quantidades de dados. Para ultrapassar estas questões de eficácia, acabaram por surgir novas bases de dados que cumprem os requisitos desejados: bases de dados NoSQL, a abreviatura de *Not Only SQL* [39–41].

[3] Do inglês: *Structured Query Language*

Estes sistemas de gestão de dados não relacionais distribuídos não têm tabelas como constituintes primários, nem utilizam a linguagem SQL para manipulação de dados. Consistem em sistemas adequados para armazenar informação em larga escala e que não necessitam de um modelo relacional.

Para além dos modelos relacionais das bases de dados tradicionais, as bases de dados NoSQL, apesar de se basearem nos mesmos princípios, podem apresentar os seguintes modelos possíveis: *Key-Value*, orientada a colunas ou orientada a documentos.

Base de Dados *Key-Value*

Este tipo de base de dados permite armazenar identificadores alfanuméricos (*Key*) e os valores que lhes correspondem (*Value*). Todas as chaves são únicas e dos dados podem ser acedidos através da sua associação à chave. Os valores podem assumir o formato de texto simples ou estruturas mais complexas, como listas ou conjuntos de dados. Nestes modelos, embora a estrutura seja mais simples do que no modelo relacional, a velocidade de consulta é superior. Suportam grandes quantidades de dados armazenados e estão preparados para lidar com elevada concorrência. A consulta e modificação dos dados é feita através da chave primária e não pelos valores [39, 40].

Base de Dados Orientada a Colunas

Este sistema, apesar de utilizar tabelas como modelo de dados, não suporta a sua associação. Numa base de dados deste tipo, os dados são armazenados separadamente por colunas e cada uma destas corresponde a um índice da base de dados. A estrutura desta base de dados define-se por Super Colunas e Famílias de Colunas e os dados podem ser acedidos através da indicação da família, chave e coluna. Cada coluna tem o mesmo tipo de dados e caraterísticas semelhantes [39, 40].

Base de Dados de Documentos

A estrutura das Bases de Dados de Documentos é muito semelhante à das *Key-Value*, no entanto, nestes casos, o *Value* assume a estrutura de um documento que pode ser armazenado sob a forma de *XML* ou *JSON*. Cada documento é identificado por uma chave única, no entanto, ao contrário das bases de dados *Key-Value*, estes sistemas permitem que a consulta seja feita tanto pela chave como pelo valor. O principal foco deste tipo de sistemas não está em controlar a concorrência entre leitura e escrita, mas sim em garantir o armazenamento de *Big Data* e um bom desempenho na consulta de dados [39, 40].

Vários são os softwares que permitem o armazenamento destes tipos de modelos de dados. Nas secções que se seguem, é feita uma exposição de alguns exemplos destes programas, sendo dado especial

destaque ao MongoDB, a estrutura onde foi construída a base de dados de documentos utilizada no presente trabalho.

2.2.4.1 REDIS

O projeto *Redis* segue a estrutura de base de dados *Key-Value*. Quando este programa é executado, os dados são inteiramente guardados em memória, onde todas as operações são realizadas. De forma assíncrona, esta informação armazenada é, periodicamente, copiada para o disco. O facto de as operações serem realizadas diretamente em memória confere ao sistema uma excelente performance, atribuindo-lhe a capacidade de lidar com cerca de 100 000 operações de leitura/escrita por segundo. O limite máximo do valor associado a uma chave é de cerca de 1GB. No entanto, uma vez que a capacidade desta base de dados é limitada pela memória física da máquina, esta não pode ser usada para armazenamento de *Big Data* [39].

2.2.4.2 CASSANDRA

A base de dados *Cassandra*, escrita em JAVA, faz parte das bases de dados orientadas a Colunas e é muito semelhante aos sistemas relacionais por ser constituída por linhas e colunas. A principal diferença está no facto de suportar o armazenamento de dados no formato estruturado, semiestruturado e não-estruturado. Esta base de dados de código aberto é capaz de lidar com bilhões de colunas e milhões de operações por dia. Os dados guardados podem ser acedidos através da linguagem CQL – *Cassandra Query Language*, que se baseia na SQL [41].

2.2.4.3 COUCHDB

A base de dados CouchDB, orientada a documentos, apresenta a vantagem de ser uma base de dados flexível e tolerante a falhas [39]. Permite o armazenamento de documentos JSON sem limite máximo definido e a consulta e manipulação de dados pode ser feita com JavaScript. Os documentos são diretamente armazenados na base de dados (Figura 2.3) e identificados por um ID único, atribuído manualmente aquando a inserção, ou automaticamente pelo CouchDB. Este sistema disponibiliza uma interface baseada em HTTP REST, o que permite o acesso à base de dados, de qualquer ambiente capaz de realizar pedidos via HTTP [42]. No entanto, este método de interação com os dados tem algumas limitações, como a falta de performance na leitura e escrita de dados em simultâneo.

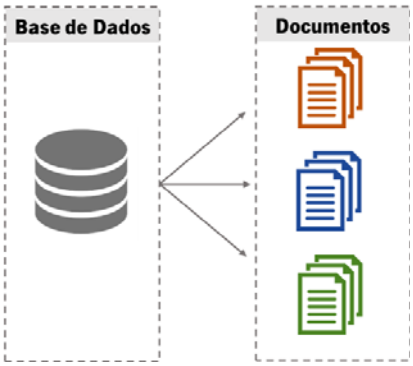


Figura 2.3 - Esquematização do modelo de dados CouchDB.

2.2.4.4 MONGODB

A MongoDB é uma base de dados de código aberto, desenvolvida em C++, bastante semelhante às bases de dados relacionais, mas com algumas caraterísticas que lhe confere vantagens. Foi desenvolvida em 2007 e lançada pela primeira vez em 2009, consiste numa base de dados de armazenamento de documentos, com uma capacidade máxima de 16MB, que são agrupados em coleções de acordo com a sua estrutura.

Tal como ilustrado na Figura 2.4, cada base de dados deste tipo pode ser constituída por várias coleções, que se distinguem pela estrutura chave-valor dos documentos que armazenam. Documentos com a mesma estrutura JSON são armazenados na mesma coleção. Ao contrário do que acontece nas bases de dados relacionais, nesta abordagem não é possível que diferentes documentos compartilhem a mesma chave. Esta funciona como um identificador único do documento e não pode ser utilizada por diferentes documentos na mesma coleção. O acesso aos dados pode ser feito através dos vários atributos definidos e não apenas pela chave identificadora [41].

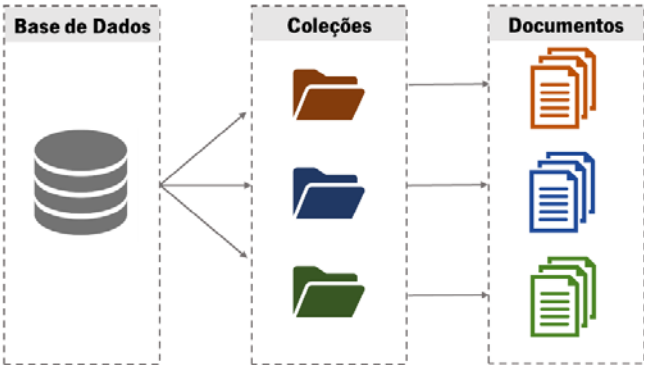


Figura 2.4 – Esquematização do modelo de dados MongoDB.

Este sistema de armazenamento orientado a documentos é capaz de suportar tipos de dados complexos e utiliza uma linguagem de consulta que permite a maioria das funções das bases de dados relacionais.

Tal como nestas, podem ser executadas quatro operações principais: criar, consultar, atualizar e apagar [39, 41]. Por defeito, o meio utilizado para gestão da base de dados é através de *Shell JavaScript*, no entanto, também é possível a sua manipulação através de outras linguagens. Algumas delas permitem a criação de documentos de entrada na MongoDB, para isso basta que possuam tipos de dados compostos por pares chave-valor, ou até mesmo estruturas JSON.

Esta base de dados utiliza o formato BSON - a forma binária do JSON, como estrutura de armazenamento de dados. A informação é inserida no formato JSON e só depois convertida para BSON. Sempre que for devolvida, é novamente convertida para JSON. Ou seja, esta estrutura binária nunca é visível para o utilizador, servindo apenas para fins internos [42].

Esta base de dados é reconhecida pela sua durabilidade e concorrência. A capacidade de criação de réplicas, implementada por um mecanismo *Master-Slave*, confere durabilidade ao sistema. Enquanto o *Master* desempenha funções de escrita ou leitura de ficheiros, o *Slave* é usado como cópia de segurança assíncrona, permitindo apenas operações de leitura. Sempre que o *Master* deixar de funcionar, o *Slave* com dados mais recentes passa a desempenhar as funções de *Master*. A MongoDB utiliza *locks* para assegurar a consistência dos dados e evitar a leitura e atualização dos dados seja feita por vários clientes em simultâneo. Para melhorar a performance, os documentos são mapeados em memória e enviados para o disco a cada 60 segundos [41].

2.3 PROCESSAMENTO DE LINGUAGEM NATURAL

Um dos campos mais antigos e que mais desafios oferece na área da Inteligência Artificial é o Processamento de Linguagem Natural (PLN). Esta área dedica-se ao estudo da linguagem humana, tendo como principal objetivo proporcionar ao computador a competência exclusiva dos humanos de entender línguas naturais, ou seja, de perceber o significado de uma frase ou de um documento [43]. Nos primeiros anos de estudo desta área, os cientistas tentaram aprovisionar as máquinas com os vocabulários e regras das linguagens humanas, no entanto, devido à variabilidade, ambiguidade e interpretação subjetiva destas línguas, o procedimento mostrou-se demasiado difícil de concretizar.

Por definição, o PLN é um subcampo da ciência da computação que se dedica à utilização de técnicas computacionais para gerar e compreender conteúdo na linguagem humana. A maioria dos sistemas de

geração de Linguagem Natural garante que as regras gramaticais, como concordância entre sujeito e verbo, sejam obedecidas, ajudando também na organização coerente de frases ou parágrafos. Estes sistemas são muito utilizados nos mecanismos de tradução automática: os textos na língua mãe são analisados do ponto de vista gramatical e concetual e é gerado o texto correspondente na língua pretendida. Os sistemas de compreensão de Linguagem Natural englobam diversas componentes como processamento de textos, análise lexical ou morfológica e análise sintática e semântica [1].

A análise semântica permite traduzir a estrutura sintática de uma frase na sua forma semântica, representando o seu significado de maneira precisa e inequívoca. Este tipo de interpretação pode ser orientado de dois modos: interpretação independente do contexto e interpretação tendo em conta o contexto. No primeiro, é feito um estudo do significado das palavras e do modo como estes significados se combinam em frases. No segundo, é tida em consideração a situação em que a frase é apresentada, é analisado o modo como o contexto afeta a interpretação de uma frase [1].

Por norma, o texto de qualquer documento na sua forma original, não se encontra bem formatado ou padronizado. Para que este tipo de dados não estruturados seja convertido em sequências de componentes linguísticos bem definidos, com uma notação e estrutura padrão, é fundamental um processamento do texto, que pode ser cumprido através de uma diversidade de métodos de PLN [1]. Na Tabela 2.2 é descrita a lista de algumas das técnicas de processamento e compreensão de texto utilizadas pelos sistemas de PLN.

Tabela 2.2 - Lista de algumas das técnicas utilizadas no PLN e respetivas descrições [1, 44, 45].

Técnica	Descrição
Remover *tags* HTML	Quando a informação a tratar é um documento HTML, é essencial a eliminação de todo o ruído que vem aglutinado ao texto, como *tags* (</>) ou JavaScript. Elementos deste tipo não acrescentam qualquer relevância aos dados e por isso podem ser removidos.
Tokenização	Qualquer porção de texto pode ser constituída por diversos componentes, como parágrafos, frases ou palavras. A tokenização consiste no processo de dividir dados textuais em elementos mais pequenos e com maior significado. Os *tokens* resultantes deste processo podem ser palavras ou frases, consoante o tipo de tokenização aplicado.
Remover acentuação	Para que o texto analisado siga a mesma codificação, é importante a remoção de acentos através da conversão dos termos para o *encoding* padrão (ASCII).

Remover *stopwords*	Nem todas as palavras que surgem num texto têm relevância suficiente para serem utilizadas na sua análise, onde importam apenas os termos com maior significado e contexto. Palavras como determinantes ou preposições aparecem com elevada frequência em documentos e são importantes para manter a sua coerência, no entanto, devido à sua baixa significância devem ser removidos.
Tratar contrações	Ao longo de um texto, principalmente quando escrito em inglês, são muitas vezes utilizadas formas contraídas das expressões, que resultam da eliminação de determinadas partes dos termos e de concatenações entre eles, por exemplo: *have not* e *haven't.* Estas versões mais curtas devem ser revertidas para o texto que lhes deu origem, evitando-se assim que os mesmos termos sejam reconhecidos como palavras divergentes.
Stemming	Uma grande parte dos termos que surgem nos textos resulta da contração entre afixos e a estrutura base da palavra *(stem)*. O processo que reverte esta contração denomina-se *stemming* e é muitas vezes utilizado para padronizar palavras, reduzindo termos que dantes eram distintos no fragmento comum que lhes deu origem. Por exemplo, o *stem* resultante das palavras *studies* e *studying* é *stud*.
Lematização	Semelhante ao *stemming*, também a lematização permite transformar um termo na sua forma base. No entanto, neste caso, o *lemma* resultante é a palavra válida que lhe deu origem e não a raiz obtida com a eliminação dos afixos. Por exemplo, o *lemma* das palavras *studies* e *studying é study*.

A acrescentar a estas técnicas, são muitas vezes realizadas operações como a eliminação de ruído dos dados, através da exclusão de carateres especiais, ou conversões de todo o texto para minúsculas/maiúsculas, uniformizando a totalidade das palavras [1].

Esta fase de processamento dos dados é uma parte fundamental de um sistema de PLN, dado que a eficácia de qualquer método de análise de texto pode ser afetada por problemas que surjam nesta fase de pré-processamento. Nesta etapa são definidas regras de padronização do texto, ou seja, são convertidos os dados não estruturados num formato intermédio, tornando o processo de descoberta de conhecimento mais eficiente [14].

Independentemente da utilização que possa ser dada ao texto, o seu tratamento e estruturação contribuirão sempre para melhores resultados em futuras aplicações. Para além de permitirem um aumento da exatidão de classificadores, uma boa formatação dos dados facilita a obtenção de informações adicionas acerca do texto [1].

O PLN inclui ainda métodos relevantes para a compreensão e análise dos dados, como é o caso da marcação lexical de partes do texto, por norma designada de POS[4] *tagging*. A POS tagging consiste no processo de classificar e atribuir classes a partes de um texto. As várias palavras que este contém são rotuladas de acordo com o contexto sintático em que se encontram, ou seja, com base na categoria lexical a que pertencem. A Figura 2.5 mostra um exemplo de marcação POS da frase *"The brown fox is quick and he is jumping over the lazy dog"*. Nesta imagem os vários termos encontram-se marcados como: determinantes (DET), adjetivos (ADJ), nomes (N), verbos (V), conjunções (CONJ), pronomes (PRON) ou advérbios (ADV) [1, 46].

DET	ADJ	N	V	ADJ	CONJ	PRON	V	V	ADV	DET	ADJ	N
The	*brown*	*fox*	*is*	*quick*	*and*	*he*	*is*	*jumping*	*over*	*the*	*lazy*	*dog*

Figura 2.5 – Exemplo do processo de POS *tagging* de uma frase [1].

Esta técnica de classificação de palavras é bastante útil em aplicações de PNL, uma vez que permite filtrar por partes específicas do discurso e utilizar estas informações para realizar análises específicas [1].

A linguagem Python inclui uma grande parte das funcionalidades necessárias para executar tarefas simples de PLN, no entanto, para o desempenho de algumas funções de processamento mais complexas surge a necessidade de a complementar com a utilização de outras ferramentas. A NLTK[5] é uma coleção de módulos, lançados sob uma licença de código-fonte aberto, que permite o processamento simbólico e estatístico de linguagem natural, em inglês. Para além de disponibilizar funções úteis que podem ser usadas como base em tarefas de PLN, também fornece versões em bruto e pré-processadas de *corpora* padrão usados na literatura. Alguns exemplos destes textos são: *Brown Corpus*, que inclui cerca de um milhão de palavras em inglês americano de diversas áreas como ficção, notícias ou religião; *Gutenberg Corpus*, uma seleção de 14 textos (1,7 milhões de palavras) recolhidos da maior coleção gratuita de e-books; e *Stopwords Corpus*, que para além de palavras comuns, inclui uma classe dedicada a *StopWords*, que consiste numa lista de cerca de 2400 palavras, em 11 idiomas, com pouco significado lexical, como proposições ou determinantes. Para além deste módulo *corpus* que permite aceder e explorar os *corpora* com facilidade, um outro programa da biblioteca NLTK bastante utilizado é o módulo probabilístico, que inclui várias classes e funções úteis no cálculo de distribuições de probabilidade [1, 6].

[4] Do inglês: *Part-of-Speech*

[5] Do inglês: *Natural Language Toolkit*

2.3.1 ÁREAS RELACIONADAS COM PLN

A utilização destes métodos de Processamento de Linguagem Natural tem um papel fundamental no sucesso de qualquer sistema de Mineração de Texto, Recuperação de Informação ou Extração de Informação. Apesar de todos estes campos se encontrarem, de certa forma, relacionados, nos pontos que se seguem é exposta uma descrição individual de cada um deles [47].

2.3.1.1 MINERAÇÃO DE TEXTO

A mineração de textos é área que se dedica à descoberta de informação disfarçada em documentos textuais. Padrões muitas vezes invisíveis e dados que não seriam acedidos através de técnicas de estatística simples, podem ser encontrados com o auxílio desta metodologia. O principal propósito das técnicas de Mineração de Texto, um domínio da mineração de dados (MD), é a extração de conhecimento a partir de informação não estruturada ou semiestruturada [48]. Consiste no processo de extração de padrões com interesse, de grandes quantidades de texto armazenado em bases de dados, objetivando a descoberta de conhecimento [49]. Um dos primeiros projetos para análise de texto na área biomédica foi desenvolvido na Universidade de Nova Iorque e consistia na análise dos sinais e sintomas de pacientes para identificar possíveis efeitos colaterais de medicamentos [50].

O termo *mineração,* tradicionalmente, refere-se ao processo de separação de um minério valioso da matéria sem valor económico. Este procedimento de divisão é idêntico ao que ocorre na mineração de textos: documentos importantes são separados de outros com pouca relevância, ou termos relevantes extraídos de uma imensidão de palavras sem significado. Neste último caso, as palavras-chave selecionadas podem ser, posteriormente, usadas para identificar padrões significativos ou fazer previsões [51].

A complexidade e ambiguidade da linguagem natural nos documentos constituem os principais obstáculos de um sistema de análise de dados. Esta caraterística atribui ao texto a capacidade de ser entendido em mais do que um sentido. Uma palavra pode ter múltiplos significados e uma frase interpretada de diferentes formas, o que dificulta bastante o processo de extração de conhecimento através de métodos computacionais [47]. Outro tema que levanta questões nesta área é a possibilidade dos mesmos documentos apresentarem, no seu conteúdo, texto em mais do que uma língua. A maioria dos sistemas de mineração não está preparado para atuar em documentos multilíngues e acaba por pô-los de parte do processo. Esta exclusão de documentos, muitas vezes relevantes, dificulta o processo de descoberta de conhecimento e tomada de decisão [14].

Para evitar este tipo de constrangimentos, são aplicadas as metodologias de PLN. Como a informação que constitui o corpo da maioria dos textos apresenta-se escrito numa linguagem humana e de forma não estruturada, é necessária a aplicação de mecanismos computacionais de Processamento de Linguagem Natural, tornando possível a extração de conhecimento de texto em bruto [52, 53].

2.3.1.2 RECUPERAÇÃO DE INFORMAÇÃO

Um sistema de Recuperação de Informação ajuda a restringir o conjunto de documentos que são relevantes para um problema específico. Uma vez que a mineração de texto envolve a aplicação de algoritmos muito complexos para grandes coleções de documentos, esta área pode acelerar significativamente a análise, reduzindo o número de documentos a avaliar [43]. Esta área está responsável por, dada uma coleção de documentos, devolver apenas os relevantes para um dado conjunto de palavras-chave [54].

Os exemplos mais conhecidos de sistemas de RI são os motores de busca (como o *Google*), que dado um conjunto de palavras fornecido pelo utilizador, identificam os documentos com maior interesse para esse tópico em toda a web. Por exemplo, caso se pretenda a análise de textos relacionados com determinada patologia, a coleção de documentos analisada deverá estar relacionada com essa doença. Para cumprir este objetivo, são utilizadas medidas e métodos estatísticos para processamento de texto e comparação com a interrogação realizada [55, 56].

Quando se pretende encontrar as palavras mais relevantes num documento, não basta obter a frequência absoluta de termos nos textos. Se assim fosse, caso se verificasse a ocorrência de alguns termos em todos os documentos, estes acabariam por ofuscar as restantes palavras da coleção, principalmente, as que não aparecem com tanta frequência, mas que podem ter relevância e eficácia superiores na representação do documento [1, 55, 56]. Para calcular esta importância de um termo num documento surgiu o índice *Tf-Idf* [6], que corresponde à combinação entre a medida da frequência de termos *(Tf)* e o inverso da frequência do termo nos vários documentos da coleção *(Idf)*. A primeira medida calcula-se através da divisão entre o número de vezes que um termo aparece num documento e o total de termos do documento. A segunda métrica é obtida através da escala logarítmica da divisão entre o total de documentos da coleção e o número de documentos onde o termo em análise aparece. Assim que obtidas ambas as variáveis, já se encontram reunidas as condições para o cálculo do índice *Tf-idf*, que resulta

[6] Do inglês: *Term-Frequency - Inverse Document Frequency*

da multiplicação da *Tf* com a *Idf*. Assumindo que um documento se encontra representado por d e um termo por x, o cálculo desta variável pode ser traduzido pela expressão:

$$Tf - Idf = \frac{nº\ de\ vezes\ que\ x\ aparece\ em\ d}{nº\ total\ de\ termos\ de\ d} \times \log\left(\frac{nº\ total\ de\ docs}{nº\ de\ docs\ que\ incluem\ x}\right)$$

Pela observação da fórmula matemática, é possível afirmar que quanto mais elevado for o valor *Tf-Idf* de um termo, maior é a relevância desta palavra na representação do documento.

2.3.1.3 EXTRAÇÃO DE INFORMAÇÃO

A Extração de Informação (EI) trata-se do campo responsável por encontrar correlações entre entidades ou atributos específicos, extraídos de documentos [14]. É uma área que participa na identificação e extração de dados textuais de variadas fontes de informação, agregando-os e criando um formato único. Para além de aplicar mecanismos de Recuperação de Informação, um sistema de EI inclui, na maioria das vezes, técnicas de PLN para que seja possível a extração de conhecimento útil a partir de dados em linguagem natural [43, 53].

Uma das metodologias usadas por estes sistemas é o reconhecimento de entidades, que permite categorizar, normalizar e mapear termos num documento, classificando os elementos em categorias predefinidas. Por exemplo, num texto o termo "*rofecoxib*" pode identificado como um medicamento, enquanto que a expressão "*enfarte do miocárdio*" como uma condição médica. Para além desta metodologia, também faz parte das tarefas da EI a descoberta de relações entre elas. [54, 57].

Na biomedicina, uma grande parte dos trabalhos de identificação de relações concentra-se na extração automática de interações entre substâncias químicas e outras entidades, como genes, proteínas, mutações pontuais, doenças, contexto fenotípico ou efeitos secundários [58]. Num contexto clínico, as relações de maior interesse são, por exemplo, os problemas de saúde apresentados pelos pacientes e os testes ou tratamentos a que estes estão sujeitos. Utilizando a extração de relações na literatura científica, é possível a descoberta de relações ocultas entre o conhecimento disponível. Este procedimento visa identificar associações que merecem uma investigação científica adicional ou encontrar evidências que corroborem uma suspeita de relação [59].

Com recurso a publicações científicas, nos anos 80, *Don R. Swanson* conseguiu relacionar uma substância conhecida pelos seus benefícios cardiovasculares – o óleo de peixe, com a síndrome de *Raynaud*, um distúrbio nos vasos sanguíneos que causa a sua constrição. A relação encontrada sugeria que suplementos de óleo de peixe podiam auxiliar no controlo dos sintomas da síndrome [59, 60]. Para

Swanson, existem sempre duas comunidades científicas que não comunicam entre si. Por exemplo, enquanto uma comunidade estuda os efeitos de uma determinada substância numa caraterística, outra pesquisa sobre a relação entre essa caraterística e uma doença. Com a aplicação da descoberta de relações na literatura científica, poderá ser possível estabelecer uma ligação entre os dois estudos, associando a substância à doença em causa (Figura 2.6) [59, 60].

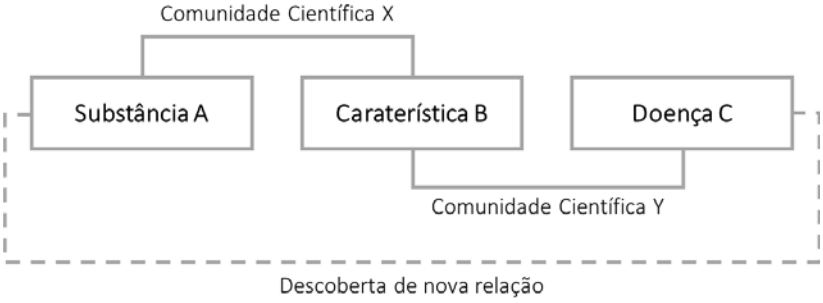


Figura 2.6 – Exemplo ilustrativo de descoberta de relações entre entidades presentes na literatura científica.

2.3.2 APLICABILIDADE DE PLN

Devido à sua aplicabilidade nas diversas técnicas de processamento e extração de conhecimento anteriormente descritas, o PLN é usado num vasto conjunto de campos da ciência. A utilização desta técnica permite obter dados devidamente processados e num formato válido para que lhes possa ser retirado o máximo proveito possível. Este procedimento contribui para uma tomada de decisão mais rápida e permite que várias organizações explorem padrões com interesse, modelos, direções, tendências e regras a partir de dados de texto em bruto.

BIBLIOTECAS DIGITAIS

As bibliotecas digitais oferecem um novo método de organização de informação que permite disponibilizar vários milhões de documentos *online*. Estes repositórios digitais reúnem esforços para a compreensão do significado dos documentos na sua coleção. Técnicas, como a RI e a MT, estão envolvidas em operações como a seleção de documentos, enriquecimento, extração de informações e entidades de controlo dos documentos. GATE, Net Owl e Aylien são exemplos de ferramentas frequentemente usadas para mineração de texto em bibliotecas digitais [14].

SETOR ACADÉMICO

Na educação, as técnicas de PLN podem ser utilizadas para variados fins, como o processamento de dados para análise de tendências educacionais numa dada região, o interesse dos alunos num setor específico ou até mesmo para encontrar a taxa de empregabilidade nesse campo. Na investigação, técnicas deste tipo ajudam a encontrar e classificar documentos de pesquisa e material de diferentes áreas num só local [14].

CIÊNCIA DA VIDA

Uma área onde a quantidade de dados textuais é bastante elevada é a que se dedica ao estudo saúde e dos organismos vivos. Neste campo, os registos médicos, como dados de pacientes, doenças, medicamentos ou sintomas, apresentam uma natureza variável que utiliza vocabulários complexos e técnicos, aumentado a dificuldade do processo de descoberta de conhecimento. Na biomédica, as ferramentas de PLN, MT ou EI permitem a extração de informações relevantes, associações e relações entre doenças, espécies e genes. Ajudam, por exemplo, na avaliação da eficácia de tratamentos médicos, através da comparação das doenças, sintomas e metodologia de tratamento. Estas técnicas são ainda aplicadas na descoberta de biomarcadores, na análise de comércio clínico e no mapeamento e exploração de doenças genéticas [14].

Quanto à bibliografia, o estilo de escrita, mais formal e complexo da literatura biomédica torna o processo de mineração de texto particularmente desafiador. Este domínio da biomedicina inclui uma grande variedade de termos e expressões que podem ser usados para referir diferentes entidades, como genes, espécies, procedimentos e técnicas. Para além disto, cada termo pode ainda apresentar diferentes grafias, abreviações ou identificadores nas bases de dados [45]. A mineração de textos biomédicos objetiva uma redução no custo e uma aceleração da descoberta, através do acesso aos factos e às suas relações explícitas e implícitas [59].

REDES SOCIAIS

A atuar no domínio das redes socias, estão alguns *softwares* de MT que se responsabilizam pela análise e monitorização de texto sem formatação em notícias *online*, *blogs* ou *e-mails*, por exemplo. Recorrendo a estas ferramentas é possível a identificação do número de publicações, gostos e seguidores nas redes sociais, o que evidencia comportamentos do público, de diferentes ou semelhantes comunidades e faixas etárias, perante as publicações a que reagiram [14].

BUSINESS INTELLIGENCE

Para auxiliar empresas e organizações a tomar as melhores decisões, a análise de textos contribui significativamente para o estudo dos seus clientes e concorrentes, proporcionando uma visão aprofundada do negócio e encontrando a melhor forma de satisfazer o comprador e obter vantagens competitivas. Ferramentas como o *IBM Text Analytics*, *Rapid Miner* e *GATE* auxiliam organizações na tomada de decisão, gerando alertas de desempenho e de mudanças de mercado e permitindo a execução de ações corretivas [14].

3. Sistema de Recolha e Tratamento de Dados

Pontos Chave:

- O sistema desenvolvido tem como principal objetivo a procura, recolha e tratamento de documentos científicos, disponibilizados por repositórios online.
- A base de dados selecionada para a extração de documentos foi a *BioMed Central*, por se tratar de uma das editoras dedicadas apenas à biomedicina e que disponibiliza a integralidade dos seus documentos.
- Todo o processo foi dividido em duas etapas principais: fase *online*, onde foi desenvolvido o *scraper* para rastreamento, recolha e armazenamento das páginas dos documentos, e a fase *offline*, onde os dados guardados foram tratados através de técnicas de PLN.
- A etapa de procura e extração de textos teve por base a arquitetura Scrapy, o armazenamento dos dados foi efetuado na BD de documentos MongoDB e o tratamento foi aplicado com a linguagem Python, utilizando a aplicação web *Jupyter Notebook*.

Após a apresentação das principais tecnologias nesta área de aplicação, segue-se a descrição do sistema de procura e recolha automática de documentos científicos ligados ao setor médico desenvolvido. Para a primeira parte do trabalho, ou seja, para implementação do *scraper*, foi utilizada a estrutura *Scrapy,* exposta na secção 2.2.3. Na etapa posterior, para processamento e análise dos textos extraídos, foram aplicados os conceitos de Processamento de Linguagem Natural, estudados na secção 2.3.

O sistema de extração proposto e esquematizado na Figura 3.1 divide-se em duas fases principais: o momento de procura, recolha e armazenamento dos dados de texto para análise – fase *online*, e a etapa de processamento dos dados – fase *offline*.

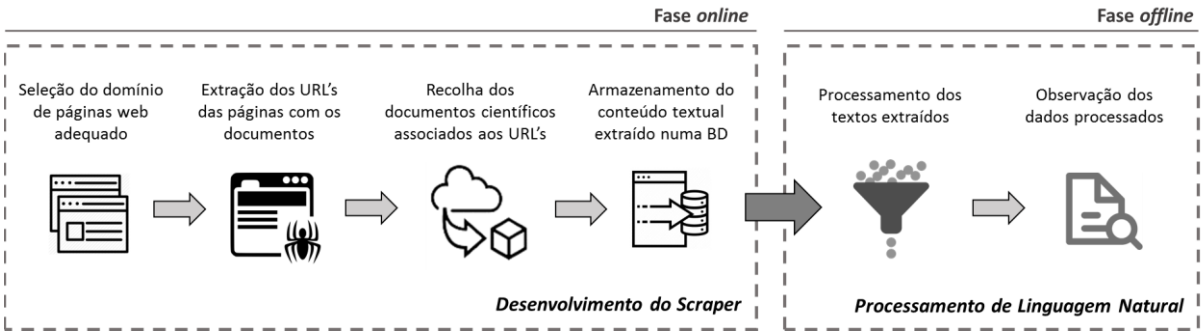


Figura 3.1 – Fluxo do processo de recolha e processamento de documentos científicos.

Na primeira etapa, tal como o próprio nome indica, é imprescindível a conexão à Internet para obtenção dos documentos académicos. A fonte de informação usada para rastreamento e extração de dados é a web e como tal, este procedimento só é possível através da utilização de Internet. Para concretização deste passo, mais à frente descrito, foi desenvolvido um *scraper*, que combina o rastreamento das páginas pretendidas (*crawling*) com a extração do seu conteúdo e termina com o armazenamento do mesmo numa Base de Dados local. Na segunda fase, já sem necessidade de acesso a uma rede, foi necessária uma limpeza e tratamento do conteúdo textual, para análise dos dados extraídos.

Nas próximas secções são descritas de forma detalhada as várias etapas envolvidas na implementação do sistema proposto.

3.1 FASE ONLINE

A Internet, principal fonte de propagação de informação em diversas áreas, constitui um meio fundamental de partilha de toda a documentação científica publicada, incluindo o abrangente setor da biomedicina. Embora a pesquisa e recolha deste tipo de dados disponibilizados online seja, por norma, efetuada manualmente, por um utilizador, este procedimento pode ser automatizado através da implementação de um *bot*. Estes programas, que objetivam a redução do tempo e esforço humano dispensado, baseiam-se num mecanismo de cópia, no qual dados específicos são recolhidos e copiados da web para um arquivo local ou base de dados, onde podem ser, posteriormente, recuperados ou analisados. Visando o cumprimento destas funcionalidades, surgiram os mecanismos de *Web Crawling* e *Web Scraping,* que permitem respetivamente, a navegação por links de websites e a recolha das informações textuais desses links. Tal como exposto nas secções anteriores, um *Web Crawler* define-se como um programa que visita todas as páginas web de um dado conjunto de URLs iniciais, a fim de extrair os links que se encontram a estas associados. Para completar esta funcionalidade, a implementação de um *Web Scraper* permite a extração de informação contida nos *websites.* [61].

A web disponibiliza uma enorme diversidade de recursos, como bases de dados e motores de busca, que permitem o acesso a este tipo de dados. No entanto, nem tudo o que é publicado online pode ser considerada informação confiável. Para evitar a assimilação de conteúdos duvidosos, é importante conhecer a origem e fiabilidade dos dados.

3.1.1 ORIGEM DOS DADOS

O primeiro passo para o planeamento e desenvolvimento do sistema proposto consistiu em explorar vários repositórios disponíveis online, a fim de encontrar uma fonte confiável de documentos científicos, que pudesse ser utilizada para extração de texto. De entre as várias identificadas, apenas interessaria uma que incluísse documentos que abrangessem áreas como a saúde, biomedicina ou biologia e cujo conteúdo completo dos documentos estivesse acessível aos utilizadores.

Há cerca de 20 anos, o setor da publicação científica divergia bastante do atual, uma vez que, a maioria dos artigos científicos era disponibilizada apenas através de modelos com pagamento de assinatura. Nesta altura, surge a *BioMed Central* (BMC), uma das primeiras entidades a introduzir o modelo de livre acesso a publicações científicas, atendendo assim às necessidades de um diverso público: autores, financiadores, bibliotecas e ou a comunidade académica [62].

Em relação às restantes fontes de literatura biomédica encontradas, a BMC destacou-se pela sua organização e facilidade de acesso a documentos académicos. Esta editora científica, criada em 1999, foi a escolhida como fonte de recolha de dados para o desenvolvimento do sistema proposto, já que disponibiliza milhares de publicações científicas sem restrições de acesso. Este repositório de documentos, em vez de disponibilizar apenas o resumo das suas publicações, como a maioria das editoras ou bases de dados científicas fazem, partilha com os seus utilizadores todo o conteúdo dos documentos que inclui [63].

A BMC faz parte da *Springer Nature* e oferece aos utilizadores a possibilidade de conhecer o conteúdo de centenas de revistas científicas de várias áreas, como a Biomedicina, Ciências da Computação, Educação, Química, Ambiente, entre outros. A Figura 3.2 permite obter uma visão real de todos os campos de estudo abrangidos pelos documentos disponíveis na página BMC.

Figura 3.2 - Página web da editora BMC com organização das revistas publicadas por área de estudo [64].

Cada uma destas áreas incluídas pela editora, possui dezenas de revistas científicas com centenas, ou milhares, de publicações cada. A Tabela 3.1 enumera alguns exemplos dos repositórios disponibilizados pela BMC e que também foram utilizados para extração dos documentos.

Tabela 3.1 – Exemplos de revistas utilizadas na extração de documentos científicos.

***Journals* da série BMC**	**Total de documentos**	**Áreas abrangidas**
Medicine	2 331	Práticas clínicas, avanços médicos, saúde pública e saúde global.
Psychiatry	3 790	Prevenção, diagnóstico e tratamento de distúrbios psiquiátricos.
Public Health	13 892	Epidemiologia das doenças e a compreensão de todos os aspetos da saúde pública.
Health Services Research	7 684	Prestação de cuidados, gestão de serviços de saúde, avaliação das necessidades de saúde.
Neurology	2 395	Prevenção, diagnóstico e tratamento de distúrbios neurológicos.
Psychology	357	Psicologia, comportamento humano, psicologia clínica, cognitiva, experimental, de saúde e social.
Medical Genetics	2 047	Efeitos da variação genética em indivíduos e a sua relação com doenças humanas.

Selecionando a revista pretendida, para além da diversa informação adicional relacionada com a mesma, é dada a possibilidade de fazer uma pesquisa orientada ao tema pretendido. Indicando no campo de pesquisa o assunto que se tenciona ver abordado nos artigos, apenas serão devolvidas as publicações relacionadas com o tema. A Figura 3.3 mostra parte do resultado de uma pesquisa na revista *BMC Psychiatry*, filtrando as publicações pelo tema *Obsessive Complusive Disorder*. Cada um dos elementos obtidos na lista de artigos encontra-se associado a outra página onde consta, para além de outros dados, o conteúdo integral do documento selecionado.

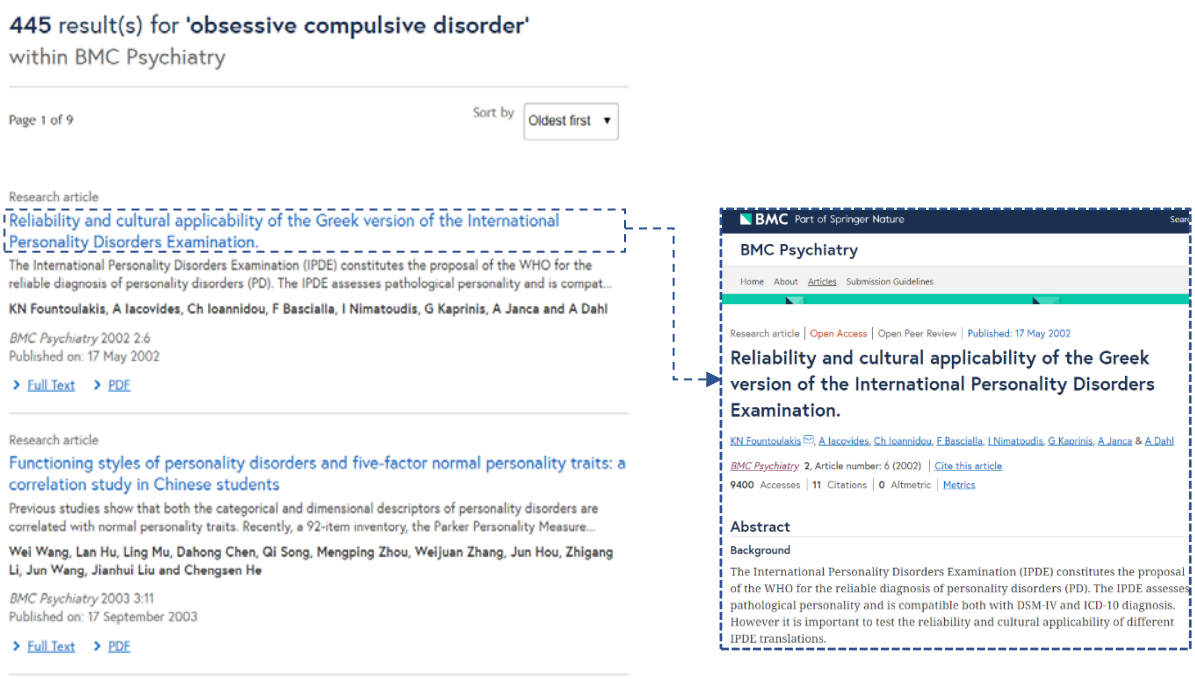


Figura 3.3 - Página devolvida no *Journal BMC Psychiatry* para a consulta *Obsessive Compulsive Disorder* (à esquerda) e parte inicial da página associada ao artigo e que engloba o corpo do documento (à direita) [62, 65].

3.1.2 PROCURA E RECOLHA DE DOCUMENTOS DA WEB

Para a concretização da etapa principal do processo, a fase de recolha de documentos académicos disponíveis na Internet, foi implementado um *web scraper*, capaz de extrair centenas de documentos científicos de repositórios *online* e armazená-los numa base de dados para posterior processamento.

Toda a lógica de rastreamento e extração foi desenvolvida sob o programa Scrapy, a *framework* de código aberto, escrita na linguagem Python, que facilita o acesso e recolha de dados de páginas web, tornando este processo mais rápido e simples. Várias foram as motivações que levaram à escolha desta ferramenta, no meio de outras, para o desenvolvimento do trabalho. Em primeiro lugar, trata-se de uma estrutura que fornece as ferramentas necessárias para ambos os processos: *crawling* e *scraping*, ou seja, permite um rastreamento de URLs e ao mesmo tempo uma extração do conteúdo textual das páginas. O facto de utilizar, como linguagem base, o Python, também constitui uma vantagem, por se tratar de uma linguagem de programação de alto nível, com uma linha de código relativamente simples. Esta *framework* proporciona aos utilizadores uma curva de aprendizagem apropriada, permitindo-lhes adquirir o conhecimento básico necessário num período aceitável. Para além disto, o Scrapy é constituído

por uma abrangente comunidade de utilizadores capazes de dar suporte no surgimento de qualquer adversidade.

Todos os desenvolvimentos realizados para a implementação do *scraper* foram realizados num ambiente de desenvolvimento Python, denominado *Spyder.* Este IDE[7] proporciona aos utilizadores um editor de texto multilingue avançado, que utiliza recursos como coloração de sintaxe, análise de código e sugestões de invocações. O *Spyder* permite fazer testes interativos, correr código ou colocá-lo em *debug* [7].

Para iniciar a utilização da *framework* Scrapy, e após a sua instalação, foi necessária a criação de um projeto para desenvolvimento do *scraper* proposto. Esta ação desencadeou a geração de um conjunto de ficheiros que correspondem aos vários constituintes da *framework*. Algumas das estruturas geradas encontram-se enumeradas na Tabela 3.2.

Tabela 3.2 - Ficheiros base gerados aquando a criação de um projeto Scrapy

Ficheiro	**Descrição**
settings.py	Ficheiro com as configurações do sistema. Permite, por exemplo, alterar o número de pedidos por segundo, adicionar extensões ou lidar com erros.
items.py	Ficheiro de parametrização dos itens a extrair das páginas web. Permite a definição dos dados na sua forma estruturada que se pretende recolher.
pipelines.py	Ficheiro para configuração da base de dados de armazenamento dos itens. Apesar do Scrapy já possuir comandos internos que permitem guardar dados em formatos como JSON ou CSV, neste ficheiro é possível definir o formato pretendido dos dados de saída.
middlewares.py	Ficheiro para definição do modo como é processada e enviada a resposta HTTP para o *Spider*. Por norma, esta componente não é modificada e mantém o padrão do Scrapy.
/spiders	Diretoria principal onde são guardadas as scripts para desenvolvimento do *crawler/scraper* responsável pelo rastreamento e recolha dos dados.

Como se pode constatar pela organização da arquitetura Scrapy, descrita no capítulo 2.2.3, os seus constituintes, na prática, podem ser associados a alguns destes ficheiros criados. Nos pontos que se seguem, serão descritos com mais pormenor os desenvolvimentos realizados em cada um dos elementos gerados.

[7] Do inglês: *Integrated Development Environment*

SETTINGS

O ficheiro *settings.py* é gerado para permitir a configuração de diversos parâmetros envolvidos no funcionamento do *Web Scraper*. É composto por uma infraestrutura que disponibiliza um espaço de varáveis globais, mapeadas com um determinado valor, que o programa vai usar para obter as suas configurações. A Tabela 3.3 resume as definições efetuadas, ou seja, os valores que foram atribuídos a cada variável configurada e uma breve descrição da mesma.

Tabela 3.3 - Lista de variáveis configuradas na componente settings.py.

Variável	Valor	Descrição
BOT_NAME	'BMCdocuments'	Designação do *Web Scraper* desenvolvido. Nome do projeto.
SPIDER_MODULES	['BMCdocuments.spiders',]	Lista de *Spiders* que o *Scrapy* vai utilizar.
ITEM_PIPEPLINES	{'BMCdocuments'.pipelines. MongoPipeline':500,}	Especificação do *Pipeline* para transmissão de dados.
DOWNLOAD_TIMEOUT	200	Máximo tempo de espera de resposta do *Downloader* (em segundos).
CONCURRENT_REQUESTS	20	Número máximo de pedidos em simultâneo realizados pelo *Downloader*.
USER_AGENT	'BMCdocuments (http://www.mydomain.com)'	Nome do utilizador usado no rastreamento das páginas.
ROBOTSTXT_OBEY	TRUE	*Bot* obedece às políticas de rastreamento dos *sites* (ficheiro *robots.txt*).
LOG_STDOUT	TRUE	Ativar armazenamento do *log* de execução.
LOG_FILE	bmc_scrapy_log.txt	Criação do ficheiro que armazena o *log* de execução.
MONGODB_SERVER	'localhost'	Configurações da base de dados criada para armazenamento dos documentos extraídos.
MONGODB_PORT	27017	
MONGODB_DB	'dbBMCdocs2019'	
MONGODB_COLLECTION	'original_publications'	

Enquanto que algumas das variáveis foram automaticamente definidas com valores por defeito aquando a criação do projeto, outras acabaram por ser, posteriormente, adicionadas ou adaptadas, conforme as necessidades do rastreador desenvolvido. O dicionário atribuído à variável ITEM_PIPELINE permite definir os *pipelines* que são utilizados para processamento dos itens extraídos. Neste caso, apenas foi definido um *pipeline* que será responsável por direcionar os itens recolhidos das páginas para a base de dados de documentos MongoDB. Sem esta configuração, o *Item Pipeline* não é ativado e os dados não são armazenados.

O valor atribuído ao elemento CONCURRENT_REQUESTS foi escolhido de modo a aumentar a performance de rastreamento e, ao mesmo tempo, evitar que as páginas alvo fossem sobrecarregadas com pedidos. Para evitar que o rastreador espere demasiado tempo pela resposta de uma página, foi atribuído à variável DOWNLOAD_TIMEOUT o valor 200, ou seja, se ao fim de 200 segundos ainda não tiver sido obtido o download da página, o pedido será descartado.

É fundamental manter o campo ROBOTSTXT_OBEY ativo, para que sejam respeitadas as regras de rastreamento definidas no ficheiro *robots.txt* das páginas percorridas pelo sistema. Desta forma, é garantido que nada é acedido ou rastreado sem o consentimento do site e o rastreador pode operar em conformidade com as normas por ele estabelecidas. A configuração do USER_AGENT também foi necessária, uma vez que a maioria das páginas web não permite rastreamento sem uma identificação do utilizador.

Para que o histórico de execução do processo de rastreamento e extração de documentos seja guardado, foi definida a *True* a variável LOG_STOUD e configurado o nome do ficheiro que se pretende gerar com a informação de log. Este ficheiro permite observar todas as páginas por onde o *web scraper* passou, os tempos de execução e, se for o caso, o registo de erros que possam ter ocorrido durante o processo.

Por fim, para que os itens extraídos das páginas pudessem ser armazenados num formato adequado, foi necessária a configuração de uma base de dados. Este procedimento exige que sejam indicados o servidor e a porta da conexão com a BD (MONGODB_SERVER e MONGODB_PORT), a designação da BD (MONGODB_DB) e a coleção onde os dados vão ser guardados (MONGODB_COLLECTION).

Estas configurações são essenciais para o bom funcionamento do *scraper*, já que permitem personalizar o comportamento do motor principal que controla o rastreador e de todos os componentes que integram a estrutura Scrapy.

SPIDER

Todos os componentes que fazem parte da constituição de uma estrutura Scrapy são essenciais para o funcionamento do *scraper*, no entanto, o elemento que está responsável pelo modo como são feitos o rastreamento e a extração das páginas é o *Spider*. Toda a lógica desenvolvida para a construção deste componente foi guardada num novo ficheiro Python, criado na diretoria */spider*.

Para o desenvolvimento do *Spider*, em primeiro lugar, importa definir o domínio válido para rastreamento, ou seja, o âmbito de URLs que o rastreador está autorizado a percorrer. Sempre que este encontre um *link* que não pertença ao domínio escolhido, a página não é acedida e o seu conteúdo não é recolhido. Por exemplo, para que a página https://www.exemplo.com/teste1.html seja alvo de rastreamento e recolha de conteúdo, um dos domínios definidos na componente *Spider* será exemplo.com [34]. Desta forma, a fim de garantir que apenas são acedidas páginas pertencentes à editora BMC, o domínio configurado neste sistema para rastreamento e recolha de documentos é **biomedcentral.com**. A escolha deste campo, para além de salvaguardar que nenhuma página sem interesse académico é rastreada, permite limitar o número de páginas acedidas pelo *Spider,* ao rejeitar as que não pertencem ao domínio.

Uma vez escolhido o domínio das páginas permitidas para análise, importa ainda definir por onde o rastreador vai começar a sua pesquisa. Nesta fase, é escolhido um URL inicial, ou um conjunto deles, para que a sua página correspondente seja acedida primeiramente e lhe seja feita uma extração de novos URLs por rastrear. Para facilitar o acesso a todas as revistas disponibilizadas pela BMC, o URL inicial escolhido para análise foi: **https://www.biomedcentral.com/journals**. Este URL devolve uma página, representada em parte pela Figura 3.4, com as centenas de revistas, organizados pelos diversos abrangidos (Figura 3.2).

Figura 3.4 - Parte da página devolvida pelo URL inicial [62].

A primeira ação do *Spider* passa por solicitar a página associada ao URL inicial, onde se encontram listadas todas as revistas disponibilizados pela BMC (Figura 3.4). Cada um destes elementos da lista encontra-se vinculado a um *hiperlink* que remete para a página principal da revista. Para que sejam encontrados documentos de todas as revistas, assim que a primeira página for devolvida ao *Spider,* é efetuada uma análise da mesma para que sejam extraídos todos os URLs das revistas BMC. Cada um dos endereços extraídos corresponderá a uma nova solicitação do *Spider.* A Figura 3.5 ilustra um exemplo de uma página devolvida numa resposta aos novos pedidos realizados (revista *BMC Psychiatry*).

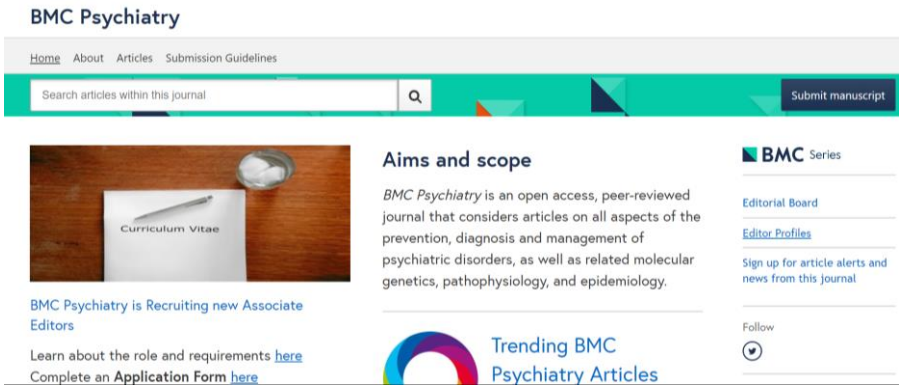


Figura 3.5 - Página principal da revista *BMC Psychiatry* [62].

Para que os documentos extraídos sejam de interesse para o utilizador, é essencial que seja aplicado um filtro na sua pesquisa. Ou seja, em vez de serem recolhidos todos os documentos sobre os mais variados temas que o setor da medicina inclui, serão apenas extraídas as publicações relacionadas com um tópico selecionado pelo utilizador. Para isto, é necessário que o *Spider* explore a página principal de cada revista, e para cada uma delas, faça um novo pedido para obter a lista total de artigos resultantes da consulta efetuada. Para que seja utilizada esta pré-seleção dos dados, basta que o utilizador passe como parâmetro de entrada do sistema a consulta que pretende realizar. Este valor é transferido ao *Spider*, que assim que começar o rastreamento terá em consideração que apenas vai extrair os artigos devolvidos na página resultante da pesquisa. Por exemplo, se um dado utilizador pretender uma análise dos documentos científicos que abordam a síndrome *Transtornos Obsessivo Compulsivos*, o *Spider* será alimentado com os parâmetros de entrada "*obsessive compulsive disorder*". Esta ação decide que páginas serão rastreadas e recolhidas dos vários repositórios, tornando o resultado de maior interesse para o utilizador. A Figura 3.6 mostra uma parte da página web devolvida pela pesquisa "*obsessive complusive disorder"* no *Journal BMC Psychiatry.*

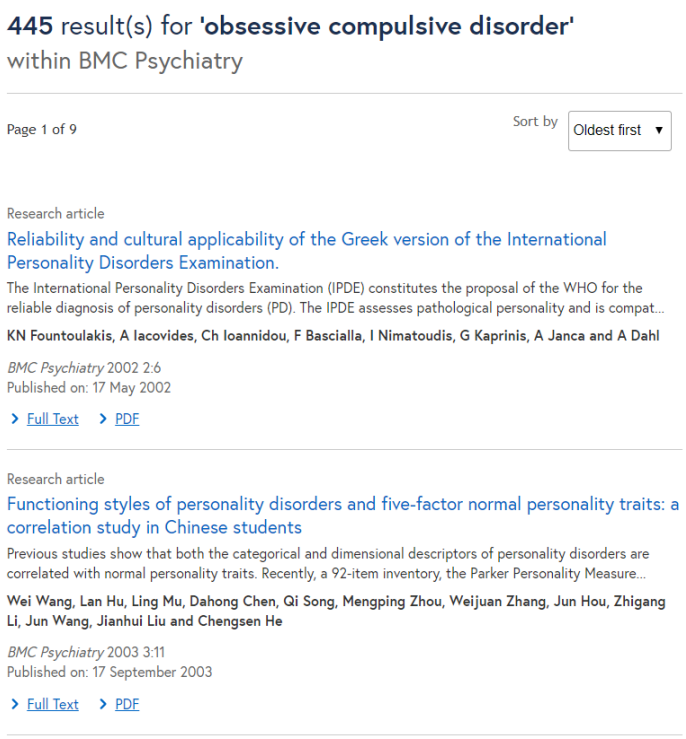


Figura 3.6 - Página devolvida pela pesquisa "*obsessive compulsive disorder*" [62].

A estrutura HTML associada a esta página será devolvida como resposta à solicitação do *Spider*, que por sua vez fará uma nova extração de endereços. Todos os artigos incluídos na lista encontram-se vinculados a um link que remete para a página com o conteúdo do documento (Figura 3.7).

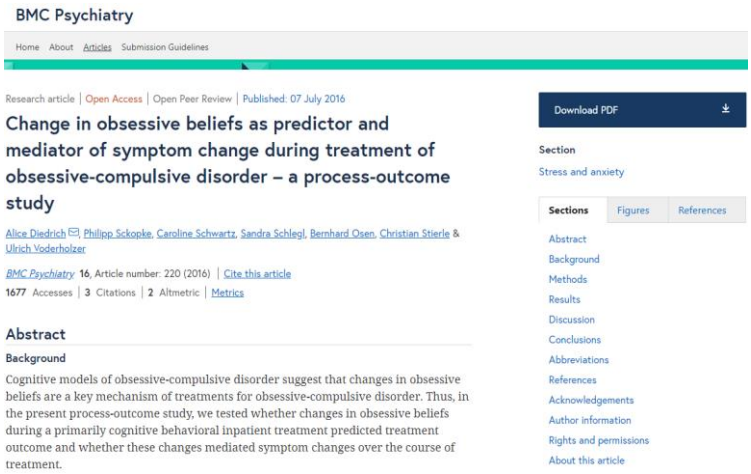


Figura 3.7 - Página com o conteúdo integral de um documento da revista *BMC Psychiatry* [66].

Estes URLs são visitados pelo rastreador e a página correspondente é passada ao *Spider*. Desta vez, este elemento não vai fazer uma extração de endereços, mas sim uma recolha do conteúdo textual da página. Em vez de realizar um novo pedido, o *Spider* retorna um *Item* preenchido com os dados extraídos. Ou seja, a página em causa é examinada para que seja atribuído a cada campo do elemento *Item* o respetivo

valor. O título do documento é extraído e atribuído ao campo *Title*, o corpo do documento ao *Content* e o endereço que permite aceder à página é passado ao campo *URL*. Esta resposta do *Spider* com o *Item* é enviada para o *Pipeline*, para que este trate de encaminhar os documentos para a BD onde vão ser armazenados. Em suma, a componente *Spider* desenvolvida está responsável pela análise das páginas HTML, recolha de URLs para novos pedidos e extração do conteúdo dos documentos para armazenamento (Figura 3.8).

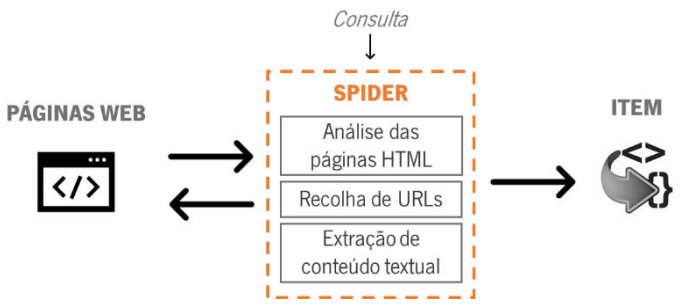


Figura 3.8 – Esquematização das funcionalidades do *Spider* e da sua interação com outras entidades.

Para além deste fluxo de rastreamento e recolha de páginas apresentado, é importante saber o modo como *Spider* as analisa. A informação que chega a este elemento, em resposta a cada solicitação realizada, é a totalidade do código HTML que compõe a página. Para que sejam recolhidas apenas as informações desejada, é essencial conhecer como é que esta estrutura se encontra organizada.

Tal como se pode observar pelo exemplo da Figura 3.9, um documento HTML é constituído por marcadores que delimitam o texto exibido nas páginas web. Para que o *Spider* seja capaz de recolher uma parte específica da página, terá de saber procurar pela posição correspondente na estrutura HTML.

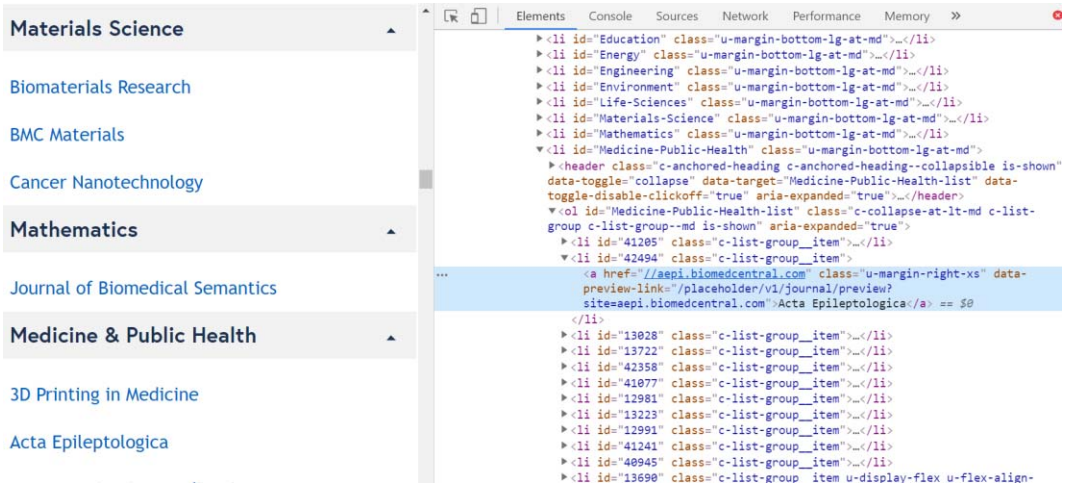


Figura 3.9 - Página devolvida pelo URL inicial (à esquerda) e o seu respetivo código HTML (à direita) [62].

Aqui surgiu a necessidade de aplicar ao código do *Spider* as funcionalidades da linguagem Xpath. Esta linguagem de consulta permite que o rastreador selecione apenas os nós pretendidos do documento HTML e de lá recolha toda a informação que precisa. Tomando como exemplo o primeiro documento HTML que o *Spider* extraiu (Figura 3.9), pode-se verificar que este apresentava na sua constituição mais informação do que a necessária. Para que apenas fossem recolhidos os URLs das revistas BMC, foi necessário especificar a localização destes endereços, no método do *Spider* responsável por examinar a página de resposta ao URL inicial. A expressão XPath utilizada para a extração destes elementos foi:

'//ol[@class="c-collapse-at-lt-md c-list-group c-list-group–md"]/li/a/@href'

Com a aplicação desta metodologia, todos os URLs listados, na localização acima descrita, são recolhidos e o *Spider* pode solicitar as respetivas páginas e analisar o seu código HTML devolvido.

A Figura 3.10 permite visualizar, de uma forma geral, as diferentes páginas por onde o *Spider* passou. Apesar de todas elas pertencerem ao mesmo domínio, o código HTML fonte vai apresentar algumas variações em cada um dos grupos de páginas, representados de A a D. Estas divergências exigem que o *Spider* se comporte de forma distinta quando está, por exemplo, a analisar a página devolvida pelo URL inicial (A) ou as páginas com o conteúdo do documento (D).

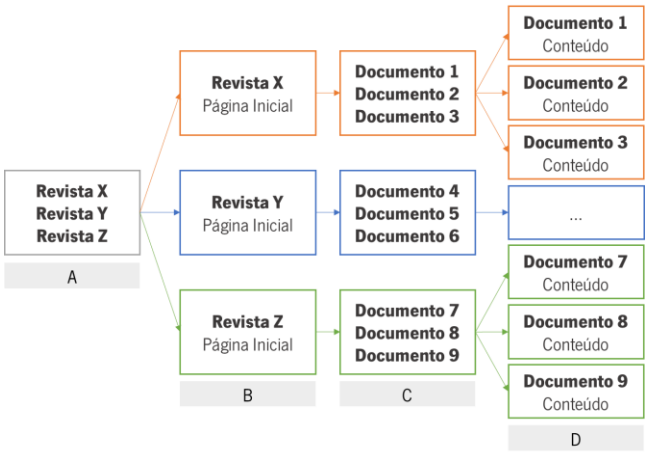


Figura 3.10 - Esquematização das páginas rastreadas pelo *Spider*.

Na última página recolhida, ou seja, na extração do corpo do documento científico, o texto que se pretende extrair está incluído em diversos elementos HTML, com diferentes níveis. Esta organização complexa dificulta a seleção dos elementos que abrangem o conteúdo do documento. Para contornar esta situação, foram selecionados, novamente através de expressões Xpath, os elementos principais, que abrangem o documento pretendido, e a partir desses foi recolhida toda a informação de texto contida nos elementos de nível mais baixo ("<p>").

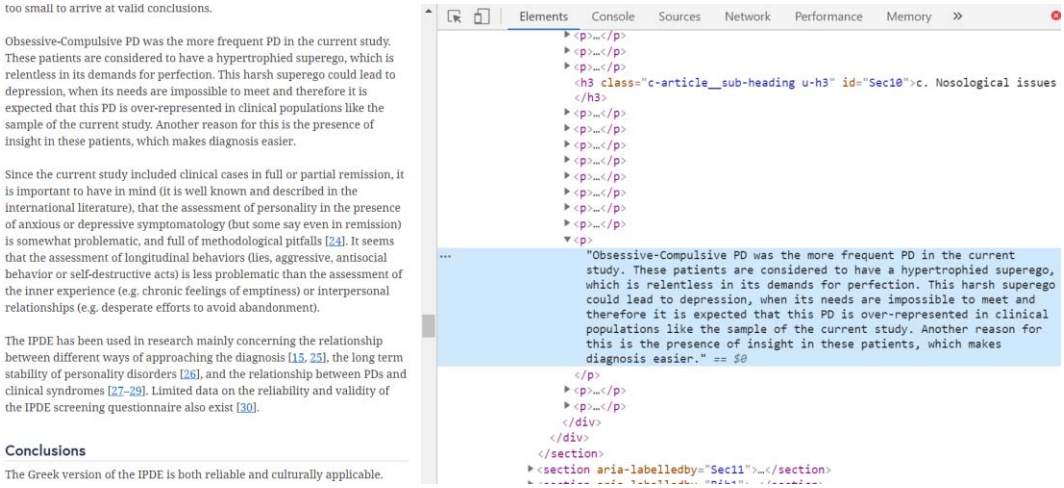


Figura 3.11 - Código fonte HTML da página com o conteúdo do documento [65].

ITEMS

Um dos principais propósitos dos mecanismos de *scraping* é a conversão de informação não estruturada, que constitui as páginas web, em dados estruturados. A classe *Item*, que integra a estrutura Scrapy, é o componente responsável por definir o formato de saída dos dados, comum a todas páginas. Nesta classe, que se encontra declarada no ficheiro *items.py*, são definidos os diferentes campos que representam os elementos a extrair das páginas web [34].

Para o sistema proposto, foi criado um único *Item*, ou seja, uma classe denominada *BmcdocumentItem,* onde foram configurados três campos com as seguintes designações: ***URL***, ***Title*** e ***Content***. Cada um destes campos pode ser visto como uma chave do dicionário que vai armazenar diferente informação da mesma página web. Enquanto o primeiro guarda o URL da página extraída, o segundo reserva o título do documento e o terceiro todo o seu conteúdo textual.

A configuração destes elementos é fundamental, uma vez que toda a arquitetura Scrapy se encontra orientada para a utilização destes componentes. O *Spider* analisa os dados recolhidos das páginas e devolve como estrutura de saída o *Item* com a informação desejada armazenada. Assim que sai do *Spider*, este elemento passa através de um *Item Pipeline* para ser guardado no formato pretendido, ou seja, numa base de dados de documentos (Figura 3.12).

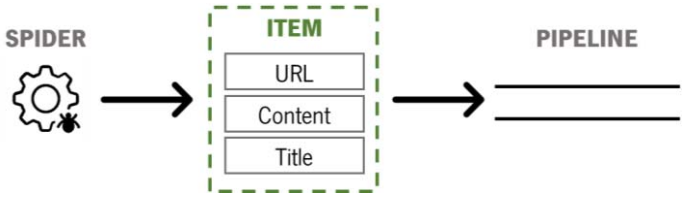


Figura 3.12 - Esquematização da constituição do *Item* e interação com outros componentes.

PIPELINES

Um *Item Pipeline* permite definir, se assim for pretendido, o meio de armazenamento dos dados recolhidos e reservados num *Item* pelo *Spider*. Cada *Pipeline* define-se por uma classe, onde é implementado um método que se responsabiliza por receber o *Item* e realizar um conjunto ações, que decidem se este continua o seu percurso pelo *Pipeline* ou se é eliminado. Para além de aplicar este filtro nos dados, a estrutura em questão permite o seu armazenamento no formato adequado.

Como se objetiva a extração de documentos científicos completos, o que significa grandes quantidades de texto, é essencial a sua reserva numa base de dados. Em vez dos comuns ficheiros de texto ou JSON, para guardar os dados extraídos, nesta componente, foi configurada a base de dados onde se pretende inserir os documentos. Deste modo, a componente *Pipeline* desenvolvida inclui todos os procedimentos necessários para: estabelecer a conexão entre a MongoDB e o Scrapy, para eliminar os dados duplicados e para os armazenar na BD (Figura 3.13).

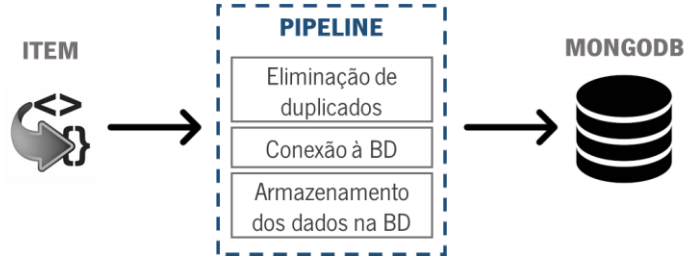


Figura 3.13 - Esquematização das funcionalidades do *Pipeline* e da sua interação com outras entidades.

Resumindo todo o fluxo, em primeiro lugar, o *Spider* é acedido para começar a enviar pedidos para o *Engine*, com base no URL inicial que lhe foi configurado. Por sua vez, o *Engine* transfere o URL para o *Scheduler*, que vai decidir quando é feita a sua recolha. Assim que for o momento, este componente faz um novo pedido ao *Engine*, para que este envie a solicitação da página ao *Downloader*. Como resposta ao pedido, o *Downloader* devolve a página HTML ao *Engine*. Por sua vez, o motor principal passa a resposta ao *Spider*, para que este possa processar a resposta. Neste momento, o *Spider* vai extrair todos os URLs pretendidos do código HTML recebido e vai fazer tantas solicitações ao *Engine* quanto endereços encontrar. O processo vai-se repetindo até que chegue ao *Spider* a resposta com ao do conteúdo documento. Nesta altura, o *Spider* vai extrair o URL, título e conteúdo do documento e retornar esta informação no *Item*. O *Engine* recebe este pedido e transfere-o para o *Pipeline*, que vai tratar de converter os dados num documento JSON e vai armazená-lo numa BD.

3.1.3 ARMAZENAMENTO DE TEXTO

Para armazenamento do texto extraído, foi utilizada a base de dados MongoDB, uma base de dados não relacional, que representa instâncias na forma de documentos JSON, permitindo o armazenamento de dados com estrutura flexível [42].

Tal como referido no capítulo anterior, tanto a criação da base de dados como a sua conexão ao Scrapy são efetuadas através da componente *Item Pipeline* e da realização e algumas configurações. No entanto, para que o Scrapy, que usa o Python como linguagem base, estabeleça ligação com o MongoDB, é necessária a instalação e importação de uma nova biblioteca: *PyMongo*. A utilização desta distribuição do Python permite tanto o acesso aos dados armazenados nas BD, como a sua uma manipulação, através desta linguagem. Para o caso específico da sua aplicação no Scrapy, este recurso foi utilizado para permitir a criação da base de dados e o seu preenchimento com tantos registos, quantos os documentos recolhidos pelo *scraper*.

Através do *Pipeline*, foi gerada uma única base de dados, com o nome '*dbBMCdocs2019*', e com apenas uma coleção, denominada '*original_publications',* onde vão ser guardados todos os documentos extraídos. Uma vez que o *Spider* retorna ao *Pipeline* o *Item* com todos os dados divididos em três campos, é importante que este também siga a mesma distribuição quando os armazena na BD. Assim, para cada *Item* devolvido pelo *Spider*, o *Pipeline* transforma a informação recebida num documento JSON organizado e insere-o na base de dados. A Figura 3.14 representa um exemplo da estrutura JSON gerada.

```
{
  _id: ObjectId("1a2b3c4d5e")
  URL: "endereco.com",
  Title: "Título do documento",
  Content: "Conteúdo integral do documento científico
            extraído pelo scraper."
}
```

Figura 3.14 - Ilustração da estrutura de documento JSON gerado pelo *Pipeline* e inserido na BD.

Cada uma das publicações extraídas pelo *Spider* foi convertida num destes documentos JSON. O primeiro atributo, denominado *_id*, é um identificador interno do MongoDB, gerado automaticamente, e que permite especificar a instância na coleção. Os restantes três atributos são os campos responsáveis por armazenar o URL, título e conteúdo da publicação enviados no *Item*.

A base de dados MongoDB foi a escolhida para armazenamento dos dados, porque a sua capacidade de lidar com grandes quantidades de informação destaca-se de entre as restantes. O facto de se tratar de

uma BD não relacional impede, por um lado, a criação de relacionamentos entre os registos, mas permite guardar elevados volumes de dados e manter um alto desempenho. Uma vez que, para o sistema desenvolvido, cada publicação científica é única e não há necessidade de estabelecer qualquer relacionamento entre elas, a utilização deste tipo de BD é a melhor opção. Comparativamente a outras base de dados do género, o MongoDB mostra uma performance superior, no que diz respeito à velocidade de inserção, atualização e consulta de dados.

3.2 FASE OFFLINE

Após a conclusão da fase de procura, recolha e armazenamento dos documentos, segue-se a etapa de tratamento do texto recolhido. Este processamento de dados não estruturados é fundamental para qualquer aplicação de PLN ou análise de dados. Todos os elementos resultantes deste tipo de procedimento, sejam eles frases ou *tokens*, poderiam ser posteriormente aproveitados para, por exemplo, alimentar sistemas de análise complexos. Nestes casos, um incorreto tratamento da informação contribuiria fortemente para a obtenção de resultados irrelevantes. Para além aumentar a eficácia deste tipo de sistemas, uma completa e organizada estruturação dos dados é essencial para adquirir mais informação acerca de um texto. Independentemente da utilização que possa ser, posteriormente, dada à informação recolhida, a sua limpeza e padronização é indispensável.

Para a concretização desta fase de processamento do texto, foi desenvolvido um programa em Python capaz de realizar o devido tratamento e limpeza dos dados. Em alternativa aos ambientes de desenvolvimento tradicionais (IDEs), como o *Spyder* utilizado para construir o *scraper,* nesta fase do sistema, foi utilizado o aplicativo web: *Jupyter Notebook*. Esta ferramenta de código aberto oferece aos utilizadores a possibilidade de combinar código com texto e visualizar o *output* obtido, tudo na mesma *script*. Para além da linguagem Julia e R*,* este aplicativo suporta código escrito em Python, a linguagem eleita, mais uma vez, para o desenvolvimento da segunda etapa do sistema.

3.2.1 TRATAMENTO E LIMPEZA DE TEXTO

Apesar de já recolhidos e devidamente guardados, os documentos científicos não se apresentam num formato que permita aos investigadores fazer algum tipo de análise ou retirar de lá qualquer informação. O conteúdo destes documentos, no seu estado bruto, é apenas um conjunto de dados sem qualquer

formatação ou padronização. Para o seu processamento, pode ser aplicada uma diversidade de técnicas capazes de converter texto em bruto, numa estrutura de palavras com significado e bem organizada.

Como todos os documentos se encontram escritos numa linguagem natural, o inglês, é fundamental a utilização dos recursos adequados para o processamento deste tipo de dados. A linguagem Python disponibiliza diversas bibliotecas que permitem um processamento deste tipo de informação. As selecionadas para aplicar no tratamento do texto foram: NLTK e Spacy. A primeira disponibiliza as ferramentas e métodos necessários para analisar os dados de texto. Inclui um conjunto de módulos eficientes na remoção de *stopwords*, tokenização, *stemming*, lematização ou análise semântica. O Spacy é uma das bibliotecas mais recentes para PLN e pode ser considerada uma das mais bem preparadas, por ser capaz de tornar as tarefas de processamento de texto bastante eficientes em termos de performance e implementação [1].

No entanto, antes de qualquer processamento, é necessário que o programa seja capaz de aceder à base de dados com os documentos. Tal como no Scrapy, também aqui é essencial uma conexão entre o MongoDB e o Python. Mais uma vez, para concretização desta tarefa, é necessária utilização da biblioteca *PyMongo*, já que esta permite ligar o Python à BD e possibilita o acesso e manipulação dos documentos.

Uma vez conectados à MongoDB, e assim que selecionada a base de dados e coleção pretendidas, todos os documentos podem ser percorridos e tratados. Como já mencionado anteriormente, estes documentos, inseridos pelo *scraper* na BD, são constituídos por quatro campos: chave identificadora, URL, título e conteúdo. Apesar de todos terem importância na caraterização do documento, o único sujeito a um tratamento do texto foi o conteúdo. Este elemento é o que apresenta a maior quantidade de informação, uma vez que inclui a totalidade do texto que compõe o documento.

Aquando o desenvolvimento do *scraper,* exposto na secção 3.1.2, foi tida em consideração que a extração de dados realizada seria apenas do texto incluído em determinados elementos da estrutura HTML, evitando assim, que toda a nomenclatura que compõe o código da página fosse também recolhida e armazenada. No entanto, apesar de apenas ser extraído o conteúdo textual dos documentos, para que seja possível a realização de qualquer tipo de análise dos dados é necessário um tratamento mais profundo da informação retirada. As técnicas utilizadas para este tipo de processamento dos documentos foram a eliminação de sinais de pontuação e caracteres especiais, a tokenização de texto, eliminação de *stopwords*, conversão dos termos para minúsculas, expansão de contrações, *stemming* e lematização.

Para cada um dos documentos contidos na coleção, foram aplicadas, de modo sequencial, todas estas etapas de limpeza e tratamento de dados, esquematizas pela Figura 3.15.

Figura 3.15 - Fluxo com todas as fases de processamento, numeradas por ordem de aplicação ao texto recolhido.

Os pontos que se seguem permitem perceber a necessidade da utilização de cada um destes passos e o modo como foram implementados.

EXPANSÃO DE CONTRAÇÕES

Ao contrário do que acontece na língua portuguesa, o inglês recorre muitas vezes a encurtamentos das palavras, a que denomina contrações. Estas versões mais curtas dos termos resultam, normalmente, da junção de duas palavravas separadas por apóstrofe, e na eliminação de determinadas letras.

Apesar deste tipo de linguagem não ser muito utilizado em documentos formais, como é o caso de documentos científicos, é importante prevenir e preparar o sistema para tratar os casos que possam existir. Para reverter esta redução de duas palavras numa separada por apóstrofes, foi feita uma expansão das contrações para a sua forma original, recorrendo a um módulo dedicado a este tipo de vocábulos: *contractions*. Este módulo disponibiliza o dicionário *contractions_dict*, ilustrado na Figura 3.16, com algumas das contrações, em inglês, mais utilizadas.

```
{"ain't": 'are not', "aren't": 'are not', "can't": 'can not', "can't've": 'can not have', "'cause":
'because', "could've": 'could have', "couldn't": 'could not', "couldn't've": 'could not have', "did
n't": 'did not', "doesn't": 'does not', "don't": 'do not', "hadn't": 'had not', "hadn't've": 'had not
have', "hasn't": 'has not', "haven't": 'have not', "he'd": 'he would', "he'd've": 'he would have', "h

(...)

e', 'would’ve': 'would have', 'wouldn’t': 'would not', 'wouldn’t’ve': 'would not have', 'y’all': 'you
all', 'y’all’d': 'you all would', 'y’all’d’ve': 'you all would have', 'y’all’re': 'you all are', 'y’a
ll’ve': 'you all have', 'you’d': 'you would', 'you’d’ve': 'you would have', 'you’ll': 'you will', 'yo
u’ll’ve': 'you shall have', 'you’re': 'you are', 'you’ve': 'you have'}
```

Figura 3.16 – Parte do dicionário *contractions_dict*, com a expansão de algumas contrações de palavras em inglês.

Com o auxílio do módulo de expressões regulares (*re*), foram substituídas todas as contrações presentes nos documentos armazenados, pela expressão que lhe corresponde.

CONVERSÃO PARA MINÚSCULAS

Outro passo importante para a homogeneização dos dados é a conversão de todas as letras de um texto para o mesmo formato: maiúsculas ou minúsculas.

Para que o texto apresentasse uma composição uniforme e para que fosse possível estabelecer correspondência entre os mesmos *tokens,* foi importante a conversão de todo o documento para carateres em minúsculo. A função utilizada para esta transformação foi:

```
texto.lower()
```

REMOÇÃO DE CARATERES ESPECIAIS E PONTUAÇÃO

Ocasionalmente, ao longo de um documento é possível surgirem alguns carateres ou símbolos não alfanuméricos, como por exemplo: @, • ou >. Uma vez que este tipo de dados não acrescenta nenhuma informação relevante ao texto, a sua eliminação contribui para a diminuição do ruído. Para além destes símbolos, a remoção dos sinais de pontuação, como vírgulas ou pontos finais, também é um passo importante na limpeza do texto.

Para a eliminação destes elementos dos dados recorreu-se à utilização de expressões regulares que podem ser aplicadas através do módulo Python *re*:

```
re.sub('[^a-zA-Z0-9\s]', '', texto)
```

A implementação deste procedimento permite encontrar todos os campos diferentes de carateres alfanuméricos e removê-los do conteúdo do documento.

TOKENIZAÇÃO DO TEXTO

O conteúdo dos documentos científicos é constituído por diversos parágrafos, que podem ser divididos em frases ou palavras. No caso da tokenização de frases, é feita uma partição do corpo do documento em várias frases com significado. Para esta divisão são utilizadas técnicas que se baseiam na procura de delimitadores específicos entre frases, como pontos finais, vírgulas ou marcadores de nova linha (/n). Quando as frases são segmentadas nos termos que as constituem, é realizada uma tokenização de palavras. Nestes casos, uma coleção de palavras é separada numa lista de termos que podem ser usados para reconstruir a frase [1].

Para a aplicação desta fase de normalização dos dados foi utilizada a biblioteca NLTK. Apesar desta estrutura disponibilizar várias interfaces para tokenização, as que esta recomenda e que foram

escolhidas para aplicar no conteúdo dos documentos foram: word_tokenize e sent_tokenize, que permitem, respetivamente, a separação do texto em frases e das frases em palavras.

O tratamento começou com a aplicação destas duas funcionalidades no texto. Em primeiro lugar, todo o conteúdo dos documentos foi dividido numa lista de frases, através da aplicação do módulo:

```
nltk.sent_tokenize(texto)
```

Ao resultado obtido com este passo, foi aplicado o módulo de repartição das frases, para que cada uma delas fosse convertida numa lista de palavras:

```
nltk.word_tokenize(frase)
```

Terminado este passo, e após uma transformação do tipo de dados, o resultado obtido para cada documento tratado foi uma lista com todas as palavras do documento, separadas por vírgulas.

REMOÇÃO DE STOPWORDS

Uma das fases mais importantes para um correto tratamento do texto é a eliminação das *stopwords*, termos com pouco, ou nenhum, significado. Palavras como proposições, determinantes ou conectores de frase, são fundamentais para estabelecer coerência num texto, no entanto, quando isoladas, através da tokenização, não apresentam qualquer relevância semântica [1].

Para a extração destes termos do resto do documento, foi utilizada, mais uma vez a biblioteca NLTK, que tem ao dispor uma lista padrão com as várias *stopwords,* em inglês. Eliminando do texto em tratamento todos os elementos que pertencem a esta lista, obteve-se uma redução significativa do número de *tokens*, restando apenas as palavras com significado. Para facilitar a compreensão deste conceito, a Figura 3.17 expõe a lista de *stopwords* obtida com a utilização da NLTK:

```
nltk.corpus.stopwords.words('english')
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'y
ourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself',
'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those',
'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'a
n', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'b
etween', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'of
f', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',
'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very',
's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'ar
en', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "have
n't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "should
n't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

Figura 3.17 - Lista de *stopwords* contida no módulo NLTK.

STEMMING E LEMATIZAÇÃO

Depois de usadas todas as técnicas de processamento anteriores, pode-se ainda implementar um dos dois processos de transformação de palavras: *stemming* ou lematização. O primeiro está responsável por converter uma palavra na sua forma base (*stem*), ajudando a padronizar as palavras no radical que lhes deu origem, independentemente das derivações a que foram sujeitas. Por outro lado, a lematização, apesar de também efetuar uma redução dos termos na sua forma primária, a palavra que resulta da transformação (*lemma*) é sempre um termo que existe no dicionário. Ao contrário do *stemming* que retira os afixos das palavras, a lematização limita-se a representar os adjetivos e substantivos no masculino do singular e as formas verbais no infinitivo. No entanto, por terem funções semelhantes, estas técnicas não devem ser aplicadas em simultâneo no mesmo conjunto de dados, sendo aconselhável a seleção de apenas uma, consoante a finalidade que se pretende dar ao resultado.

Mais uma vez, a NLTK disponibiliza as ferramentas necessárias para aplicar este tipo tratamento no corpo do texto. Um dos *stemmers* mais utilizados, e aplicado nos *tokens* gerados, é o PorterStemmer que pode ser importado do módulo *stem*.

```
PorterStemmer().stem(token)
```

Nos mesmos dados de entrada do *stemmer*, também foi aplicado um algoritmo de lematização, recorrendo à classe WordNetLemmatizer da biblioteca NLTK.

```
WordNetLemmatizer().lemmatize(token)
```

Por fim, após a aplicação destas técnicas de processamento a cada um dos documentos, surgiu a necessidade de armazenar os dados resultantes. Para reservar esta informação tratada e para que, mais tarde, possa ser utilizada, por exemplo, como parâmetro entrada de sistemas de análise de texto, foram geradas novas coleções na mesma base de dados dos documentos originais.

A primeira coleção, denominada *sentences_publications,* vai guardar os mesmos documentos que a *original_publications*, mas desta vez com o seu conteúdo num formato estruturado em listas de frases. Nesta nova coleção, são armazenados os dados resultantes do processo de tokenização do texto em frases. A acrescentar a estas, foram ainda geradas mais duas coleções: *lemma_word_publications* e *stem_word_publications*. Para além do identificador do objeto, do URL e do título, estas novas coleções guardam o conteúdo do documento organizado numa lista de *tokens,* na forma de termos isolados, devidamente selecionados e tratados. No caso da primeira, os dados armazenados resultam do processo

de lematização e na segunda os *tokens* resultantes foram obtidos pelo processo de *stemming*. No capítulo 4, será visto com maior detalhe o conteúdo destas coleções, através da observação de um exemplo prático.

3.2.2 OBSERVAÇÃO DE DADOS

Uma vez tratados e novamente armazenados os dados dos documentos extraídos da editora BMC, é importante uma observação dos resultados obtidos. Para a concretização deste passo, para além de serem aplicadas técnicas de Processamento de Linguagem Natural, foram ainda estudados métodos envolvidos na área da Recuperação de Informação.

Ainda no âmbito do processamento de texto, a classificação gramatical automática dos termos, ou seja, a POS *tagging* das palavras de um documento é uma tarefa fundamental de qualquer sistema de PLN. Estas categorias de termos servem como conjuntos de caraterísticas que podem ser usados em tarefas de análise complexas. No caso de textos biomédicos, para que sejam, por exemplo, extraídas relações entre entidades é essencial que o sistema em causa seja alimentado com uma correta marcação gramatical do texto [46].

Na área da Recuperação de Informação, é relevante o estudo da frequência dos termos e da relevância que estes apresentam num determinado documento. Uma das tarefas importantes neste campo é o cálculo do índice Tf-Idf, um indicador do nível de importância de um termo num documento.

POS TAGGING

A POS *tagging*, ou seja, a marcação de partes do texto segundo sua categoria lexical, como nomes, verbos, adjetivos ou advérbios, é muito útil quando se pretende filtrar apenas partes específicas do texto. Este tipo de marcação permite a extração e estudo dos termos que pertencem apenas a um determinado grupo da divisão, possibilitando, por exemplo, uma observação dos vocábulos mais frequentes em cada classe.

Para a aplicação desta marcação do conteúdo dos documentos científicos, foi mais uma vez utilizada a biblioteca NLTK, de onde foi obtido o módulo *pos_tag*.

```
nltk.pos_tag(token)
```

Na secção anterior, para além da descrição de todo o procedimento de limpeza e tratamento de texto, foi mencionado que após este processamento dos documentos, a informação resultante seria

armazenada numa nova coleção da BD. Esta reserva dos dados tratados evita que os documentos sejam processados de cada vez que surge a necessidade de os utilizar. Neste caso, para a POS *tagging* dos termos de cada documento, foi utilizada, como fonte de dados, a coleção *sentences_publications*, onde consta o texto organizado numa lista de frases resultantes do processo de tokenização. A cada um dos termos da lista, foi aplicado o módulo *pos_tag*, do qual resultou um tuplo contendo a palavra e a respetiva categoria lexical a que pertence. Por exemplo, se um dos *tokens* da lista for '*obsessive'*, o resultado da aplicação deste método seria *('obsessive', 'JJ').* Isto indica que a palavra '*obsessive'* pertence à classe gramatical dos adjetivos.

FREQUÊNCIA DE TERMOS

Outro método interessante para explorar o texto recolhido é a identificação das palavras que mais informação fornecem acerca do documento. Este cálculo do número de vezes que uma palavra aparece num documento é uma tarefa morosa se for feita manualmente, no entanto, recorrendo às funcionalidades da linguagem Python este processo pode ser facilmente automatizado.

Nesta fase, depois do processamento dos dados e uma vez que estes se apresentam na forma de *tokens,* é possível encontrar a frequência de todas as palavras no documento. A NLTK disponibiliza o método *FreqDist()* que, para um dado texto fornecido como parâmetro de entrada, calcula a frequência de cada termo que o constitui:

```
FreqDist(tokens)
```

Uma vez obtido o número de vezes que cada termo aparece num documento é possível a construção de gráficos ilustrativos que mostrem as palavras com maior frequência no texto e por isso, as palavras que melhor resumem o conteúdo do documento. Um método interessante para uma observação simples e apelativa dos termos com maior frequência nos documentos é a Word Cloud. Esta técnica pode ser utilizada em vários contextos e tem como propósito principal, fornecer ao utilizador uma visão geral do texto, através de uma representação gráfica das palavras mais frequentes. A imagem obtida inclui as diversas palavras do documento desordenadas, coloridas e com um tamanho proporcional à sua frequência no texto [67]. Para a aplicação deste método de observação de dados foi utilizada a biblioteca do Python *WordCloud* que permite obter a nuvem de termos baseada na sua frequência:

```
WordCloud.generate_from_frequencies(fdist)
```

A geração desta imagem, cujo exemplo pode ser observado no estudo de caso do capítulo 4, é possível através da importação da biblioteca *matplotlib*, que permite que outras bibliotecas executem os seus gráficos.

ÍNDICE TF-IDF

Apesar da frequência distribuída ser um valor importante para observar os termos mais relevantes no corpo de um único documento, não é a propriedade mais eficiente a indicar a importância destas palavras no documento, em relação aos restantes que constituem a coleção. Esta frequência tem em consideração apenas o texto que se está a analisar, e por isso, não é correto afirmar que as palavras com maior frequência são as que melhor permitem identificar o documento. Um termo que aparece um elevado número de vezes em quase todos os documentos não é importante na representação dos mesmos. Por este motivo, foi utilizado o índice *Tf-Idf,* para encontrar os *tokens* que melhor representam os documentos extraídos da editora BMC. Como referido na secção 2.3.1, este valor é o resultado da multiplicação entre a medida da frequência de termos *(Tf)* e o inverso da frequência de documentos *(Idf)*. Enquanto o primeiro consiste na divisão entre o número de ocorrências de um termo num documento e o total de termos do documento, a segunda variável é obtida através do logaritmo da divisão entre o total de documentos da coleção e o número de documentos onde o termo aparece.

Pela fórmula de cálculo deste índice, verifica-se que a importância de uma palavra numa publicação, ou seja, o valor do *Tf-Idf,* aumenta proporcionalmente ao número de vezes que o termo aparece no documento e diminui com o aumento da frequência do mesmo nas restantes publicações da coleção.

Este procedimento de cálculo do índice *Tf-idf* implicou que para cada documento, já tratado e armazenado na BD, fosse obtido o valor de todas as variáveis envolvidas na fórmula. Para estas operações foram utilizadas as diversas aplicações que o Python disponibiliza. A obtenção da frequência de cada um dos termos no documento, ou seja, o número de vezes que um determinado *token* se repete numa lista de *tokens*, foi possível através da aplicação da subclasse de dicionários *Counter().* O cálculo da quantidade total de termos na lista foi realizado através da função *len()* e a obtenção do número de documentos da coleção foi possível com a aplicação da função *count()* do PyMongo. Por fim, para que fosse encontrado o número de documentos que incluem o termo em questão, recorreu-se às funcionalidades dos dicionários no Python.

4. Estudo de Caso em Documentos Científicos sobre TOC

Pontos Chave:

- Os documentos em estudo, para além de pertencerem à editora BMC, abordam a síndrome: Transtornos Obsessivo Compulsivos.
- A recolha das publicações relacionadas com o tema resultou no armazenamento de 1794 documentos em, aproximadamente, 2 minutos.
- Aos textos em bruto armazenados, foram aplicadas técnicas de PLN que transformaram os dados em três formatos distintos possíveis: lista de *tokens* na forma de frases, lista de termos após a lematização e lista de termos com aplicação de *stemming*.
- Qualquer um dos formatos tratados, mais a versão original dos documentos, foram armazenados em diferentes coleções da BD, deixando-os disponíveis para um rápido e direto acesso.

Para melhor compreensão do sistema proposto, foi então realizado um estudo de caso que demonstra, na prática, todo o fluxo desenvolvido. Esta secção tem como finalidade, descrever todo o processo de pesquisa, extração, armazenamento, tratamento e análise de dados de um caso específico. Neste sistema, o utilizador tem um papel fundamental uma vez que é ele que decide o tema dos documentos que vão ser rastreados e extraídos. Para o caso de estudo de seguida exposto, a extração foi limitada a documentos com referência a Transtornos Obsessivo-Compulsivos (do inglês, *Obsessive Compulsive Disorders)*.

4.1 PROCURA E RECOLHA DE DOCUMENTOS SOBRE TOC

Tal como mencionado anteriormente, o repositório escolhido para extração dos documentos científicos biomédicos foi a editora BMC. Todos os *Journals* que esta entidade disponibiliza foram rastreados e de lá foram recolhidos os documentos pertencentes ao tópico escolhido.

Uma vez desenvolvido o sistema de *scraping*, o primeiro passo para a recolha e armazenamento dos documentos é a seleção do tema que se pretende ver abordado nas publicações. Uma das perturbações mentais que mais desperta o interesse dos investigadores do tema, e que afeta cerca de 2 a 3% da população em geral, são os Transtornos Obsessivo Compulsivos (TOC) [22]. Para que a documentação

extraída pelo sistema proposto incluísse informação sobre este tema, foi passado como parâmetro de entrada do *scraper* o tópico: *obsessive compulsive disorder*. Esta ação aplica um filtro nos documentos da BMC para que apenas sejam devolvidas as publicações relacionadas com TOC.

Todos os documentos alvo de recolha apresentam o seu conteúdo textual em Inglês e foram extraídos em outubro de 2019. As datas de publicação destes documentos encontram-se compreendidas entre os anos 1999 e 2019. A Tabela 4.1 resume algumas caraterísticas gerais dos documentos extraídos.

Tabela 4.1 - Caraterísticas gerais dos documentos utilizados como dados de estudo.

Número de documentos sobre TOC	1794
Idioma	Inglês
Intervalo de publicações	Anos 1999-2019
Tipo de documentos	Artigos Científicos, Erratas
Área de estudo dos documentos	Medicina e Saúde Pública, Biomedicina, Psicologia, Psiquiatria, Biologia, Genética, Hematologia, Oncologia, entre outros.

Para a execução do *scraper*, é indispensável o posicionamento da linha de comandos na diretoria do projeto Scrapy. Após a indicação do comando *crawl*, específico do Scrapy, é necessária a comunicação do nome do *spider* que se pretende correr e a atribuição dos respetivos parâmetros de entrada. No caso em estudo, para que fossem apenas devolvidos os documentos com conteúdos sobre a síndrome TOC, o campo passado como entrada do *spider* foi: *obsessive compulsive disorder*.

Sempre que o *scraper* é executado, são geradas mensagens de log, que detalham todas as etapas de concretização do processo e que são guardadas num ficheiro criado para o efeito. Neste relatório, é possível observar todos os endereços por onde o rastreador passou e, em caso de erro de execução, este fica registado para consulta. Para além destes dados, o histórico também guarda todas a configurações aplicadas no *scraper* e as diversas métricas que permitem conhecer tempos de execução e quantidade de páginas afetadas. Uma pequena parte do log de execução do *scraper* encontra-se exposta na Figura 4.1.

```
2019-10-08 09:52:49 [scrapy.utils.log] INFO: Scrapy 1.5.1 started (bot: BMCdocuments)
2019-10-08 09:52:49 [scrapy.utils.log] INFO: Versions: lxml 4.3.0.0, libxml2 2.9.9, cssselect 1.0.3, parsel 1.5.1, w3lib 1.20.0, Twisted 18.9.0,
                                             Python 3.6.8 |Anaconda, Inc.| (default, Dec 30 2018, 18:50:55) [MSC v.1915 64 bit (AMD64)],
                                             pyOpenSSL 18.0.0 (OpenSSL 1.1.1c  28 May 2019), cryptography 2.4.2,
                                             Platform Windows-10-10.0.18362-SP0
2019-10-08 09:52:49 [scrapy.crawler] INFO: Overridden settings: {'BOT_NAME': 'BMCdocuments', 'CONCURRENT_REQUESTS': 20, 'DOWNLOAD_TIMEOUT': 200,
                                                                 'LOG_FILE': 'bmc_scrapy_log.txt', 'LOG_STDOUT': True,
                                                                 'NEWSPIDER_MODULE': 'BMCdocuments.spiders', 'ROBOTSTXT_OBEY': True,
                                                                 'SPIDER_MODULES': ['BMCdocuments.spiders'],
                                                                 'USER_AGENT': 'BMCdocuments (+http://www.mydomain.com)'}
```

(a)

```
2019-10-08 09:54:55 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://bmcneurosci.biomedcentral.com/articles/10.1186/1471-2202-13-136>
2019-10-08 09:54:56 [scrapy.core.scraper] DEBUG: Scraped from <200 https://bmcneurosci.biomedcentral.com/articles/10.1186/1471-2202-13-136>
{'URL': 'https://bmcneurosci.biomedcentral.com/articles/10.1186/1471-2202-13-136'}
2019-10-08 09:54:56 [scrapy.log] DEBUG: Document added to MongoDB database!
```

(b)

```
2019-10-08 09:54:57 [scrapy.core.engine] INFO: Closing spider (finished)
2019-10-08 09:54:57 [scrapy.statscollectors] INFO: Dumping Scrapy stats:
{'downloader/request_bytes': 1018626,
 'downloader/request_count': 2813,
 'downloader/request_method_count/GET': 2813,
 'downloader/response_bytes': 120567023,
 'downloader/response_count': 2813,
 'downloader/response_status_count/200': 2808,
 'downloader/response_status_count/400': 5,
 'finish_reason': 'finished',
 'finish_time': datetime.datetime(2019, 10, 8, 9, 54, 57, 702515),
 'httperror/response_ignored_count': 5,
 'httperror/response_ignored_status_count/400': 5,
 'item_scraped_count': 1794,
 'log_count/DEBUG': 6402,
 'log_count/INFO': 29,
 'log_count/WARNING': 2,
 'response_received_count': 2813,
 'start_time': datetime.datetime(2019, 10, 8, 9, 52, 55, 832066)}
2019-10-08 09:54:57 [scrapy.core.engine] INFO: Spider closed (finished)
```

(c)

Figura 4.1 - Partes do log de execução do *scraper* desenvolvido.

Na Figura 4.1a, é possível observar que, em primeiro lugar, foi guardada no ficheiro toda a informação referente às configurações do *scraper,* mencionadas na Tabela 3.3*,* e válidas no momento em que o mesmo foi executado. Para além da reserva destes dados, foi também guardado um registo por cada visita do rastreador, por cada extração de página e para cada documento devidamente armazenado na BD (Figura 4.1b). Por fim, assim que terminada a execução do *scraper,* foram armazenados todos os dados estatísticos envolvidos no processo (Figura 4.1c). Observando estes últimos registos, foi possível perceber que, para o caso em estudo, foram solicitadas 2813 páginas, das quais 2808 responderam com sucesso e 5 devolveram erro, originados pela tentativa de rastreamento de *Journals* que não incluem qualquer publicação relacionada com a doença pesquisada. Das 2808 páginas visitadas apenas foram extraídos 1794 itens, valor que corresponde ao número de páginas com o conteúdo dos documentos. Quanto aos tempos de execução, verifica-se que para todo o processo de rastreamento de páginas, extração e armazenamento de conteúdos, foram necessários aproximadamente 2 minutos.

4.2 ARMAZENAMENTO DOS DOCUMENTOS

À medida que o *scraper* é executado e os vários documentos extraídos, o seu conteúdo é armazenando na base de dados MongoDB gerada: *dbBMCdocs2019*. Cada item que o *Item Pipeline* recebe do *Spider*, contendo o URL, título e conteúdo do documento, é inserido na forma de um documento JSON na coleção *original_publications* desta BD. Assim que terminada a execução do *scraper*, ou seja, quando concluída a procura e recolha das publicações sobre Transtornos Obsessivo Compulsivos, a totalidade dos documentos recolhidos passa a estar disponível na base de dados criada. Nesta fase, a *dbBMCdocs2019* inclui todos os documentos na sua forma original e não estruturada, tal como foram extraídos da página HTML.

O MongoDB disponibiliza uma interface gráfica, para que os utilizadores acedam de forma direta à informação armazenada. Esta GUI[8], denominada *MongoDB Compass*, permite, para além de observar e manipular a informação inserida, visualizar dados como o tamanho da coleção, o número total de documentos armazenados ou o tamanho médio de cada documento. Pela observação da Figura 4.2, é possível confirmar que o número de documentos armazenados na BD, relacionados com o tema TOC, coincide com o número de extrações realizadas pelo *scraper*. Os 1794 documentos ocupam cerca de 77.7MB e, em média, cada documento tem cerca de 44KB. A coleção contém apenas um índice, que corresponde ao código identificador do documento e que apresenta um tamanho de 44KB.

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size
original_publications	1,794	44.4 KB	77.7 MB	1	44.0 KB

Figura 4.2 - Interface gráfica MongoDB com os dados da coleção criada para armazenamento de documentos.

Dentro da coleção *original_publications*, ilustrada na Figura 4.3, é possível observar, para além do identificador gerado automaticamente pelo MongoDB (_id), todos os atributos inseridos pelo *scraper* : URL, *Title* e *Content*. Estes três campos, que chegaram à BD através do Item Pipeline, incluem dados do tipo *string* e sem limite de tamanho. Apesar dos atributos URL e *Title* serem preenchidos com valores relativamente pequenos, o campo *Content*, por sua vez, ocupa quase todo o espaço de cada registo.

[8] Do inglês: *Graphical User Interface*

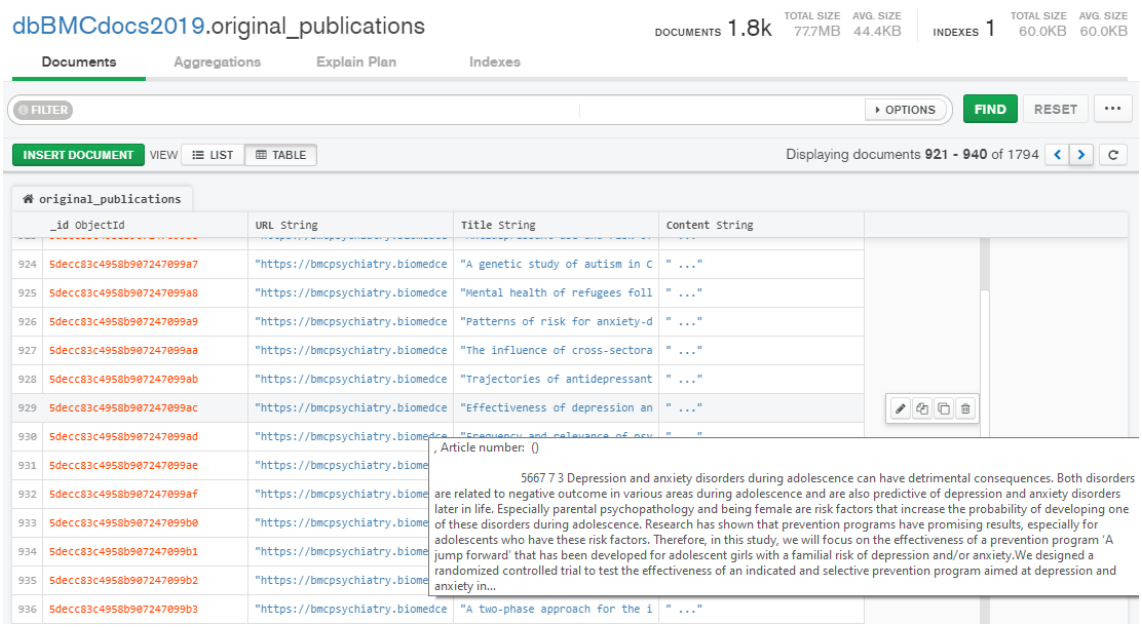


Figura 4.3 - Interface gráfica MongoDB com os campos que constituem cada documento da coleção *original_publications*.

4.3 TRATAMENTO E OBSERVAÇÃO DOS DADOS

Para que os dados de texto das publicações possam ser aproveitados, por exemplo, por sistemas complexos de Análise ou Mineração de Textos, é importante que estes se encontrem devidamente formatados e estruturados. Neste caso específico, para que investigadores dedicados ao estudo de textos relacionados com Transtornos Obsessivo Compulsivos, possam aplicar métodos e algoritmos de descoberta de conhecimento, é obrigatório que os dados de entrada estejam devidamente tratados.

Uma vez que o conteúdo dos documentos já se encontra reservado, a fase que se segue é a de tratamento do texto recolhido. Como os campos *URL* e *Title* da coleção de publicações desempenham uma função meramente informativa, o processamento de dados incide apenas no conteúdo textual do documento, armazenado no campo *Content*. Nesta etapa, todos os métodos descritos na secção 3.2.1 foram aplicados aos dados armazenados na coleção *original_publications*, ou seja, ao conteúdo das publicações do repositório da BMC que abordam TOC. A Figura 4.4 ilustra parte de um documento, no seu estado original, guardado na BD.

```
                                1315 6 Clinical and pharmacological studies of obsessive-compulsive disorder (OCD) have suggested
that the serotonergic systems are involved in the pathogenesis, while structural imaging studies have found some neuroanatomi
cal abnormalities in OCD patients. In the etiopathogenesis of OCD, few studies have performed concurrent assessment of geneti
c and neuroanatomical variables.We carried out a two-way ANOVA between a variable number of tandem repeat polymorphisms (5-HT
TLPR) in the serotonin transporter gene and gray matter (GM) volumes in 40 OCD patients and 40 healthy controls (HCs).We foun
d that relative to the HCs, the OCD patients showed significant decreased GM volume in the right hippocampus, and increased G
M volume in the left precentral gyrus. 5-HTTLPR polymorphism in OCD patients had a statistical tendency of stronger effects o
n the right frontal pole than those in HCs.Our results showed that the neuroanatomical changes of specific GM regions could b
e endophenotypes of 5-HTTLPR polymorphism in OCD.Obsessive-compulsive disorder (OCD) was made a disease independent of anxiet
y disorder in DSM-5. One of the reasons for this separation is that the biological bases of OCD and anxiety disorder are diff
erent [].Structural imaging studies have found neuroanatomical abnormalities in the cortico–striatal–thalamo–cortical (CSTC)
circuits in OCD patients []. A recent voxel-based morphometry (VBM) systematic review suggested that widespread structural ab
normalities may contribute to neurobiological vulnerability to OCD []. We previously found the presence of regional gray matt
er (GM) and white matter (WM) volume abnormalities in OCD patients [].Furthermore, positron emission tomography (PET) and fun
ctional magnetic resonance imaging (fMRI) have revealed abnormal activities in different nodes of the CSTC circuits in OCD pa
tients compared with healthy controls (HCs) [, ]. In our previous fMRI study, we found decreased activations in several brain
regions including the orbitofrontal cortex (OFC) [] and a specific relationship between fMRI activation and symptom subtypes
[].Meanwhile, family and twin studies have provided evidence for the involvement of a genetic factor in OCD. However, many li
nkage, association, and genome-wide association studies have failed to identify responsible genes [, ].Molecular genetic stud
```

Figura 4.4 - Parte de um dos documentos armazenados na BD na sua forma original [68].

Apesar do programa, que agrega todas as técnicas de processamento, receber como entrada um documento em bruto e retornar o texto já completamente tratado, nesta secção vão ser expostos os resultados obtidos após algumas das etapas principais de processamento desenvolvidas.

Em primeiro lugar, após a importação de todas as bibliotecas Python e a conexão desta linguagem à coleção da MongoDB, foi aplicada a eliminação de carateres especiais, a conversão de todo o texto para letras minúsculas e a expansão das expressões contraídas. Estes passos permitiram uma padronização dos dados, eliminando carateres sem valor e contribuindo para a uniformização de expressões com o mesmo significado. A Figura 4.5 mostra o resultado da aplicação destas técnicas no mesmo documento em bruto, representado pela Figura 4.4.

```
, article number:  () 1315 6 clinical and pharmacological studies of obsessive-compulsive disorder (ocd) have suggested that
the serotonergic systems are involved in the pathogenesis, while structural imaging studies have found some neuroanatomical a
bnormalities in ocd patients. in the etiopathogenesis of ocd, few studies have performed concurrent assessment of genetic and
neuroanatomical variables.we carried out a two-way a between a variable number of tandem repeat polymorphisms (5-httlpr) in t
he serotonin transporter gene and gray matter (gm) volumes in 40 ocd patients and 40 healthy controls (hcs).we found that rel
ative to the hcs, the ocd patients showed significant eased gm volume in the right hippocampus, and increased gm volume in th
e left precentral gyrus. 5-httlpr polymorphism in ocd patients had a statistical tendency of stronger effects on the right fr
ontal pole than those in hcs.our results showed that the neuroanatomical changes of specific gm regions could be endophenotyp
es of 5-httlpr polymorphism in ocd.obsessive-compulsive disorder (ocd) was made a disease independent of anxiety disorder in
dsm-5. one of the reasons for this ration is that the biological bases of ocd and anxiety disorder are different .structural
imaging studies have found neuroanatomical abnormalities in the cortico–striatal–thalamo–cortical (cstc) circuits in ocd pati
ents . a recent voxel-based morphometry (vbm) systematic review suggested that widespread structural abnormalities may contri
bute to neurobiological vulnerability to ocd . we previously found the presence of regional gray matter (gm) and white matter
(wm) volume abnormalities in ocd patients .furthermore, positron emission tomography (pet) and functional magnetic resonance
imaging (fmri) have revealed abnormal activities in different nodes of the cstc circuits in ocd patients compared with health
y controls (hcs) , . in our previous fmri study, we found eased activations in several brain regions including the orbitofron
tal cortex (ofc) and a specific relationship between fmri activation and symptom subtypes .meanwhile, family and twin studies
have provided evidence for the involvement of a genetic factor in ocd. however, many linkage, association, and genome-wide as
```

Figura 4.5 - Dados obtidos após eliminação de carateres especiais, conversão para minúsculas e expansão de contrações.

Nesta fase, ainda não foram removidos os sinais de pontuação como vírgulas e pontos finais, devido à sua relevância no processo de tokenização do texto em frase. Para esta etapa de segmentação, todo o documento resultante da aplicação dos métodos anteriores foi transformado numa lista com porções de texto, normalmente correspondentes a frases ou parágrafos. Esta divisão do documento não é feita, unicamente, com base nos sinais de pontuação, mas também tendo em consideração o contexto dos

dados textuais em causa. Cada fragmento, ou seja, cada *token* resultante representa uma unidade lógica de pensamento. Para ilustrar este procedimento, foram destacados os sete primeiros elementos do *array* de *tokens,* que resultou da tokenização do mesmo documento (Figura 4.6).

```
[', article number: () 1315 6 clinical and pharmacological studies of obsessive-compulsive disorder (ocd) have suggested that t
he serotonergic systems are involved in the pathogenesis, while structural imaging studies have found some neuroanatomical abno
rmalities in ocd patients.'
 'in the etiopathogenesis of ocd, few studies have performed concurrent assessment of genetic and neuroanatomical variables.we
carried out a two-way a between a variable number of tandem repeat polymorphisms (5-httlpr) in the serotonin transporter gene a
nd gray matter (gm) volumes in 40 ocd patients and 40 healthy controls (hcs).we found that relative to the hcs, the ocd patient
s showed significant eased gm volume in the right hippocampus, and increased gm volume in the left precentral gyrus.'
 '5-httlpr polymorphism in ocd patients had a statistical tendency of stronger effects on the right frontal pole than those in
hcs.our results showed that the neuroanatomical changes of specific gm regions could be endophenotypes of 5-httlpr polymorphism
in ocd.obsessive-compulsive disorder (ocd) was made a disease independent of anxiety disorder in dsm-5.'
 'one of the reasons for this ration is that the biological bases of ocd and anxiety disorder are different .structural imaging
studies have found neuroanatomical abnormalities in the corticostriatalthalamocortical (cstc) circuits in ocd patients .'
 'a recent voxel-based morphometry (vbm) systematic review suggested that widespread structural abnormalities may contribute to
neurobiological vulnerability to ocd .'
 'we previously found the presence of regional gray matter (gm) and white matter (wm) volume abnormalities in ocd patients .fur
thermore, positron emission tomography (pet) and functional magnetic resonance imaging (fmri) have revealed abnormal activities
in different nodes of the cstc circuits in ocd patients compared with healthy controls (hcs) , .'
 'in our previous fmri study, we found eased activations in several brain regions including the orbitofrontal cortex (ofc) and
a specific relationship between fmri activation and symptom subtypes .meanwhile, family and twin studies have provided evidence
for the involvement of a genetic factor in ocd.']
```

Figura 4.6 - Dados obtidos após a tokenização do texto em frases.

Uma grande parte dos sistemas de análise de textos necessita que os dados de entrada se apresentem neste formato, ou seja, em porções de texto lógicas e não apenas termos individualizados. Um exemplo de utilização deste tipo de dados é a marcação POS, que permite classificar os vocábulos de um texto de acordo com a sua estrutura lexical, tendo em consideração o contexto em que estes estão envolvidos.

Para que este formato de texto não fosse perdido e pudesse ser acedido a qualquer momento, foi gerada uma nova coleção na base de dados, denominada *sentences_publications,* onde foram armazenados todos os 1794 documentos sujeitos a este processo de tokenização (Figura 4.7).

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size
original_publications	1,794	44.4 KB	77.7 MB	1	60.0 KB
sentences_publications	1,794	40.0 KB	70.0 MB	1	60.0 KB

Figura 4.7 - Interface gráfica MongoDB com os dados da coleção inicial (*original_publications*) e da nova coleção gerada para armazenamento do resultado da tokenização em frases (*sentences_tokenization*).

Na etapa que se segue, foi novamente aplicada a tokenização do texto, mas desta vez em palavras individualizadas. O resultado obtido com a aplicação deste passo, foi a representação de cada documento por uma lista de termos e não uma lista de frases, tal como se verificou na etapa anterior. A Figura 4.8 permite observar o resultado deste procedimento, aplicado nos mesmo dados em que foi aplicada a tokenização em frases.

```
['clinical', 'and', 'pharmacological', 'studies', 'of', 'obsessivecompulsive', 'disorder', 'ocd', 'have', 'suggested', 'tha
t', 'the', 'serotonergic', 'systems', 'are', 'involved', 'in', 'the', 'pathogenesis', 'while', 'structural', 'imaging', 'stud
ies', 'have', 'found', 'some', 'neuroanatomical', 'abnormalities', 'in', 'ocd', 'patients', 'in', 'the', 'etiopathogenesis',
'of', 'ocd', 'few', 'studies', 'have', 'performed', 'concurrent', 'assessment', 'of', 'genetic', 'and', 'neuroanatomical', 'v
ariableswe', 'carried', 'out', 'a', 'twoway', 'a', 'between', 'a', 'variable', 'number', 'of', 'tandem', 'repeat', 'polymorph
isms', 'httlpr', 'in', 'the', 'serotonin', 'transporter', 'gene', 'and', 'gray', 'matter', 'gm', 'volumes', 'in', 'ocd', 'pat
ients', 'and', 'healthy', 'controls', 'hcswe', 'found', 'that', 'relative', 'to', 'the', 'hcs', 'the', 'ocd', 'patients', 'sh
owed', 'significant', 'eased', 'gm', 'volume', 'in', 'the', 'right', 'hippocampus', 'and', 'increased', 'gm', 'volume', 'in',
'the', 'left', 'precentral', 'gyrus', 'httlpr', 'polymorphism', 'in', 'ocd', 'patients', 'had', 'a', 'statistical', 'tendenc
y', 'of', 'stronger', 'effects', 'on', 'the', 'right', 'frontal', 'pole', 'than', 'those', 'in', 'hcsour', 'results', 'showe
d', 'that', 'the', 'neuroanatomical', 'changes', 'of', 'specific', 'gm', 'regions', 'could', 'be', 'endophenotypes', 'of', 'h
ttlpr', 'polymorphism', 'in', 'ocdobsessivecompulsive', 'disorder', 'ocd', 'was', 'made', 'a', 'disease', 'independent', 'o
f', 'anxiety', 'disorder', 'in', 'dsm', 'one', 'of', 'the', 'reasons', 'for', 'this', 'ration', 'is', 'that', 'the', 'biologi
cal', 'bases', 'of', 'ocd', 'and', 'anxiety', 'disorder', 'are', 'different', 'structural', 'imaging', 'studies', 'have', 'fo
und', 'neuroanatomical', 'abnormalities', 'in', 'the', 'corticostriatalthalamocortical', 'cstc', 'circuits', 'in', 'ocd', 'pa
tients', 'a', 'recent', 'voxelbased', 'morphometry', 'vbm', 'systematic', 'review', 'suggested', 'that', 'widespread', 'struc
tural', 'abnormalities', 'may', 'contribute', 'to', 'neurobiological', 'vulnerability', 'to', 'ocd', 'we', 'previously', 'fou
nd', 'the', 'presence', 'of', 'regional', 'gray', 'matter', 'gm', 'and', 'white', 'matter', 'wm', 'volume', 'abnormalities',
'in', 'ocd', 'patients', 'furthermore', 'positron', 'emission', 'tomography', 'pet', 'and', 'functional', 'magnetic', 'resona
```

Figura 4.8 - Dados obtidos após a tokenização do texto em palavras.

Uma vez que o texto já se encontra dividido em termos, ou *tokens*, segue-se a fase de eliminação das palavras sem significado. Esta remoção das *stopwords* contribui para a obtenção de um texto com maior significância, já que as palavras sem pouca ou nenhuma relevância no texto são removidas através desta etapa. O resultado deste procedimento pode ser visualizado pela Figura 4.9. Comparando estes dados com os da figura anterior, verifica-se que antes da eliminação das *stopwords* (Figura 4.8), o texto incluía palavras como *"the", "of"* ou *"that",* que acabaram por ser removidas da lista devido ao seu escasso valor semântico.

```
['clinical', 'pharmacological', 'studies', 'obsessivecompulsive', 'disorder', 'ocd', 'suggested', 'serotonergic', 'systems',
'involved', 'pathogenesis', 'structural', 'imaging', 'studies', 'found', 'neuroanatomical', 'abnormalities', 'ocd', 'patient
s', 'etiopathogenesis', 'ocd', 'studies', 'performed', 'concurrent', 'assessment', 'genetic', 'neuroanatomical', 'variablesw
e', 'carried', 'twoway', 'variable', 'number', 'tandem', 'repeat', 'polymorphisms', 'httlpr', 'serotonin', 'transporter', 'ge
ne', 'gray', 'matter', 'gm', 'volumes', 'ocd', 'patients', 'healthy', 'controls', 'hcswe', 'found', 'relative', 'hcs', 'ocd',
'patients', 'showed', 'significant', 'eased', 'gm', 'volume', 'right', 'hippocampus', 'increased', 'gm', 'volume', 'left', 'p
recentral', 'gyrus', 'httlpr', 'polymorphism', 'ocd', 'patients', 'statistical', 'tendency', 'stronger', 'effects', 'right',
'frontal', 'pole', 'hcsour', 'results', 'showed', 'neuroanatomical', 'changes', 'specific', 'gm', 'regions', 'could', 'endoph
enotypes', 'httlpr', 'polymorphism', 'ocdobsessivecompulsive', 'disorder', 'ocd', 'made', 'disease', 'independent', 'anxiet
y', 'disorder', 'dsm', 'one', 'reasons', 'ration', 'biological', 'bases', 'ocd', 'anxiety', 'disorder', 'different', 'structu
ral', 'imaging', 'studies', 'found', 'neuroanatomical', 'abnormalities', 'corticostriatalthalamocortical', 'cstc', 'circuit
s', 'ocd', 'patients', 'recent', 'voxelbased', 'morphometry', 'vbm', 'systematic', 'review', 'suggested', 'widespread', 'stru
ctural', 'abnormalities', 'may', 'contribute', 'neurobiological', 'vulnerability', 'ocd', 'previously', 'found', 'presence',
'regional', 'gray', 'matter', 'gm', 'white', 'matter', 'wm', 'volume', 'abnormalities', 'ocd', 'patients', 'furthermore', 'po
sitron', 'emission', 'tomography', 'pet', 'functional', 'magnetic', 'resonance', 'imaging', 'fmri', 'revealed', 'abnormal',
'activities', 'different', 'nodes', 'cstc', 'circuits', 'ocd', 'patients', 'compared', 'healthy', 'controls', 'hcs', 'previou
s', 'fmri', 'study', 'found', 'eased', 'activations', 'several', 'brain', 'regions', 'including', 'orbitofrontal', 'cortex',
'ofc', 'specific', 'relationship', 'fmri', 'activation', 'symptom', 'subtypes', 'meanwhile', 'family', 'twin', 'studies', 'pr
ovided', 'evidence', 'involvement', 'genetic', 'factor', 'ocd', 'however', 'many', 'linkage', 'association', 'genomewide', 'a
```

Figura 4.9 - Dados obtidos após a eliminação das *stopwords* da lista de *tokens*.

Para finalizar o tratamento do texto, falta a aplicação dos métodos que contribuem para a redução da variabilidade de palavras com a mesma origem morfológica: a lematização e *stemming*. Devido às semelhanças de funcionalidade de ambas as técnicas, a aplicação destes métodos foi realizada em separado nos mesmos dados de entrada.

Com a aplicação de técnicas de lematização no texto resultante da eliminação das *stopwords* (Figura 4.9), obteve-se a mesma lista de *tokens*, mas com alguns dos seus termos apresentados na forma

canónica. Tomando como exemplo a terceira palavra da lista anterior, *"studies",* é possível observar, através da Figura 4.10, que após a sua lematização, este *token* foi transformado no termo: *"study".*

```
['clinical', 'pharmacological', 'study', 'obsessivecompulsive', 'disorder', 'ocd', 'suggested', 'serotonergic', 'system', 'in
volved', 'pathogenesis', 'structural', 'imaging', 'study', 'found', 'neuroanatomical', 'abnormality', 'ocd', 'patient', 'etio
pathogenesis', 'ocd', 'study', 'performed', 'concurrent', 'assessment', 'genetic', 'neuroanatomical', 'variableswe', 'carrie
d', 'twoway', 'variable', 'number', 'tandem', 'repeat', 'polymorphism', 'httlpr', 'serotonin', 'transporter', 'gene', 'gray',
'matter', 'volume', 'ocd', 'patient', 'healthy', 'control', 'hcswe', 'found', 'relative', 'hcs', 'ocd', 'patient', 'showed',
'significant', 'eased', 'volume', 'right', 'hippocampus', 'increased', 'volume', 'left', 'precentral', 'gyrus', 'httlpr', 'po
lymorphism', 'ocd', 'patient', 'statistical', 'tendency', 'stronger', 'effect', 'right', 'frontal', 'pole', 'hcsour', 'resul
t', 'showed', 'neuroanatomical', 'change', 'specific', 'region', 'could', 'endophenotypes', 'httlpr', 'polymorphism', 'ocdobs
essivecompulsive', 'disorder', 'ocd', 'made', 'disease', 'independent', 'anxiety', 'disorder', 'dsm', 'one', 'reason', 'ratio
n', 'biological', 'base', 'ocd', 'anxiety', 'disorder', 'different', 'structural', 'imaging', 'study', 'found', 'neuroanatomi
cal', 'abnormality', 'corticostriatalthalamocortical', 'cstc', 'circuit', 'ocd', 'patient', 'recent', 'voxelbased', 'morphome
try', 'vbm', 'systematic', 'review', 'suggested', 'widespread', 'structural', 'abnormality', 'may', 'contribute', 'neurobiolo
gical', 'vulnerability', 'ocd', 'previously', 'found', 'presence', 'regional', 'gray', 'matter', 'white', 'matter', 'volume',
'abnormality', 'ocd', 'patient', 'furthermore', 'positron', 'emission', 'tomography', 'pet', 'functional', 'magnetic', 'reson
ance', 'imaging', 'fmri', 'revealed', 'abnormal', 'activity', 'different', 'node', 'cstc', 'circuit', 'ocd', 'patient', 'comp
ared', 'healthy', 'control', 'hcs', 'previous', 'fmri', 'study', 'found', 'eased', 'activation', 'several', 'brain', 'regio
n', 'including', 'orbitofrontal', 'cortex', 'ofc', 'specific', 'relationship', 'fmri', 'activation', 'symptom', 'subtypes',
'meanwhile', 'family', 'twin', 'study', 'provided', 'evidence', 'involvement', 'genetic', 'factor', 'ocd', 'however', 'many',
'linkage', 'association', 'genomewide', 'association', 'study', 'failed', 'identify', 'responsible', 'gene', 'molecular', 'ge
```

Figura 4.10 - Dados obtidos após a aplicação do método de lematização na lista de *tokens*.

Para além desta técnica, a outra abordagem alternativa à lematização é a aplicação do método de *stemming,* novamente, nos mesmos dados obtidos com a eliminação das *stopwords*. Com este método, os *tokens* da lista foram reduzidos ao *stem* origem, verificando-se uma significante diferença entre os resultados da aplicação de *stemming* e a lista de *tokens* passada como entrada. Observando na Figura 4.11, o resultado do processo de *stemming*, verifica-se que o mesmo *token* do exemplo anterior, nesta situação, foi transformado no termo *"studi".*

```
['clinic', 'pharmacolog', 'studi', 'obsessivecompuls', 'disord', 'ocd', 'suggest', 'serotonerg', 'system', 'involv', 'pathoge
nesi', 'structur', 'imag', 'studi', 'found', 'neuroanatom', 'abnorm', 'ocd', 'patient', 'etiopathogenesi', 'ocd', 'studi', 'p
erform', 'concurr', 'assess', 'genet', 'neuroanatom', 'variablesw', 'carri', 'twoway', 'variabl', 'number', 'tandem', 'repea
t', 'polymorph', 'httlpr', 'serotonin', 'transport', 'gene', 'gray', 'matter', 'volum', 'ocd', 'patient', 'healthi', 'contro
l', 'hcswe', 'found', 'rel', 'hc', 'ocd', 'patient', 'show', 'signific', 'eas', 'volum', 'right', 'hippocampu', 'increas', 'v
olum', 'left', 'precentr', 'gyru', 'httlpr', 'polymorph', 'ocd', 'patient', 'statist', 'tendenc', 'stronger', 'effect', 'righ
t', 'frontal', 'pole', 'hcsour', 'result', 'show', 'neuroanatom', 'chang', 'specif', 'region', 'could', 'endophenotyp', 'httl
pr', 'polymorph', 'ocdobsessivecompuls', 'disord', 'ocd', 'made', 'diseas', 'independ', 'anxieti', 'disord', 'dsm', 'one', 'r
eason', 'ration', 'biolog', 'base', 'ocd', 'anxieti', 'disord', 'differ', 'structur', 'imag', 'studi', 'found', 'neuroanato
m', 'abnorm', 'corticostriatalthalamocort', 'cstc', 'circuit', 'ocd', 'patient', 'recent', 'voxelbas', 'morphometri', 'vbm',
'systemat', 'review', 'suggest', 'widespread', 'structur', 'abnorm', 'may', 'contribut', 'neurobiolog', 'vulner', 'ocd', 'pre
vious', 'found', 'presenc', 'region', 'gray', 'matter', 'white', 'matter', 'volum', 'abnorm', 'ocd', 'patient', 'furthermor',
'positron', 'emiss', 'tomographi', 'pet', 'function', 'magnet', 'reson', 'imag', 'fmri', 'reveal', 'abnorm', 'activ', 'diffe
r', 'node', 'cstc', 'circuit', 'ocd', 'patient', 'compar', 'healthi', 'control', 'hc', 'previou', 'fmri', 'studi', 'found',
'eas', 'activ', 'sever', 'brain', 'region', 'includ', 'orbitofront', 'cortex', 'ofc', 'specif', 'relationship', 'fmri', 'acti
v', 'symptom', 'subtyp', 'meanwhil', 'famili', 'twin', 'studi', 'provid', 'evid', 'involv', 'genet', 'factor', 'ocd', 'howe
v', 'mani', 'linkag', 'associ', 'genomewid', 'associ', 'studi', 'fail', 'identifi', 'respons', 'gene', 'molecular', 'genet',
'studi', 'focus', 'structur', 'includ', 'receptor', 'transport', 'protein', 'serotonerg', 'dopaminerg', 'systembas', 'transpo
rt', 'imag', 'find', 'pet', 'studi', 'found', 'eas', 'serotonin', 'transport', 'bind', 'insular', 'cortex', 'ocd', 'patient',
```

Figura 4.11 - Dados obtidos após a aplicação do método de *stemming* na lista de *tokens*.

A escolha da técnica a aplicar depende das necessidades dos sistemas que utilizam os dados e dos resultados obtidos com cada uma delas. Por este motivo, foi gerado para cada um dos métodos, uma nova coleção na mesma base de dados. A coleção que armazena os dados resultantes da lematização denomina-se *lemma_words_publications* e a que inclui o resultado do processo de *stemming* designa-se *stem_words_publications.*

Em suma, na Figura 4.12, encontram-se representadas todas as coleções geradas para a base de dados *dbBMCdocs2019.* Para além da coleção original, que reserva o conteúdo dos documentos sobre TOC extraído das páginas web, foram criadas três coleções com diferentes estágios de processamento: tokenização do documento em frases (*sentences_publications*), tokenização do texto em termos e aplicação da lematização (*lemma_words_publications*) e, por último, utilização do *stemming*, como método alternativo à lematização *(stem_words_publications).*

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size
stem_words_publications	1,794	37.3 KB	65.4 MB	1	64.0 KB
sentences_publications	1,794	40.0 KB	70.0 MB	1	60.0 KB
original_publications	1,794	44.4 KB	77.7 MB	1	60.0 KB
lemma_words_publications	1,794	41.5 KB	72.8 MB	1	72.0 KB

Figura 4.12 - Interface gráfica MongoDB com os dados de todas as coleções geradas após a execução do *scraper* e o tratamento dos documentos incluídos no estudo de caso.

Após o tratamento e armazenamento de todos os documentos recolhidos, foi possível a observação de algumas caraterísticas dos dados. Através da implementação da técnica POS *tagging*, foi possível obter a marcação gramatical de todos os termos dos textos reservados na coleção *sentences_publications*, onde se encontram organizados os documentos em listas de frases. A cada uma das listas, de cada um dos documentos, foi aplicado o algoritmo responsável por este tipo de categorização de termos. A Figura 4.13 exemplifica uma frase de um dos documentos, com os seus termos devidamente marcados. Neste exemplo, as *tags* apresentam o seguinte significado: JJ – adjetivo, NNS – nome no singular, IN – preposição, CC - conjunção coordenativa, VBP – verbo no presente, VBN – verbo no particípio passado, RB - advérbio, VBG – verbo no gerúndio.

```
[('general', 'JJ'), ('indicators', 'NNS'), ('such', 'JJ'), ('as', 'IN'), ('restlessness', 'NN'), (',', ','), ('nervousness', 'N
N'), (',', ','), ('and', 'CC'), ('tension', 'NN'), ('are', 'VBP'), ('included', 'VBN'), ('here', 'RB'), (',', ','), ('as', 'I
N'), ('are', 'VBP'), ('additional', 'JJ'), ('somatic', 'JJ'), ('c', 'NN'), ('signs', 'NNS'), ('(', '('), ('for', 'IN'), ('examp
le', 'NN'), (',', ','), ('trembling', 'VBG'), (')', ')'), ('.', '.')]
```

Figura 4.13 - Frase de um dos documentos da coleção *sentences_publications* após a aplicação da técnica de POS *tagging*.

Com este tipo de classificação de palavras foi possível a aplicação de um filtro nos dados, a fim de serem encontradas apenas as palavras de determinadas classes. Tomando como exemplo um filtro pelas

categorias nomes e adjetivos, foram selecionados todos os termos responsáveis por identificar pessoas, lugares, entidades ou coisas em geral e possíveis atributos que os caraterizem. Após esta limitação do texto, foi implementado o conceito de coocorrência de palavras e encontrado o nome ou adjetivo que mais vezes se aproxima de outro num documento. Para a concretização desta tarefa, foi em primeiro lugar, feita uma contagem do número de vezes que as palavras ocorrem muito próximas umas das outras e, em seguida, usados os resultados obtidos para estimar o grau de associação entre elas. Para cada termo selecionado numa frase, foram examinadas as cinco palavras seguintes e contadas as suas ocorrências nesse contexto. Este tipo de observação dos dados foi possível, recorrendo aos módulos *FreqDist* e *ConditionalFreqDist* da biblioteca NLTK.

Na Tabela 4.2 pode-se observar um exemplo do resultado da implementação deste método num documento. Neste modelo, foram encontradas as palavras que, no mesmo documento, surgiram mais vezes associadas aos termos: *patients, disorder, compulsive, study* e *symptoms.*

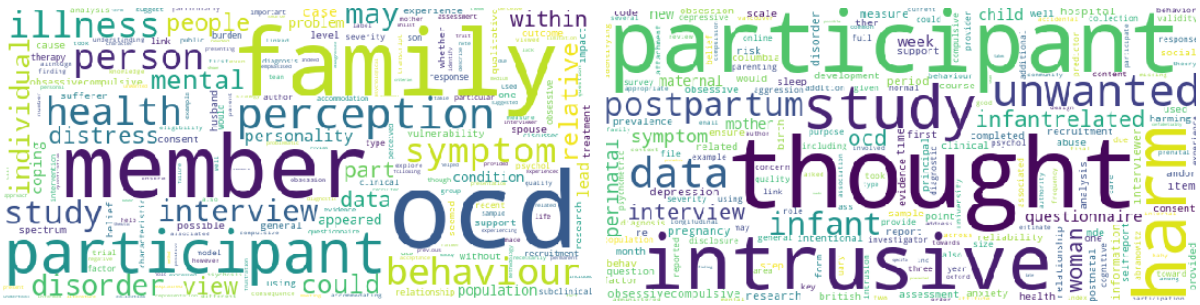
Tabela 4.2 – Associação entre nomes ou adjetivos do mesmo documento, com base no número de ocorrências dos cinco termos mais próximos

Título do documento	Termo pesquisado	Termo associado
Subtyping patients with heroin addiction at treatment entry: factor derived from Self-Report Symptom Inventory [69].	*patients*	*heroin*
	disorder	*young*
	compulsive	*behavior*
	study	*hypothesis*
	symptoms	*patients*
Prevalence of body-focused repetitive behaviors in three large medcal colleges of karachi: a cross-sectional study [70].	*patients*	*dermatology*
	disorder	*sindrome*
	compulsive	*skin*
	study	*students*
	symptoms	*anxiety*

Observando os dados obtidos com este procedimento, verifica-se que, nenhuma das associações encontradas é compatível nos dois documentos, o que se justifica pelos divergentes tópicos abordados nos textos. Apesar de nem todas as associações serem conclusivas, no primeiro documento observa-se uma possível relação entre pacientes e a heroína ou transtornos e indivíduos jovens. No segundo

documento, são atribuídas associações entre pacientes e a área da dermatologia, entre o ato compulsivo e a pele, e entre os sintomas e a ansiedade. Esta forma de estudar os dados é bastante útil para encontrar de forma rápida e sem esforço do utilizador, possíveis relações entre termos nos diversos documentos.

Para além deste método de marcação gramatical, para que se conheça melhor um texto, é importante o estudo de indicadores como a frequência das palavras que este inclui. Um método útil para obter, de forma gráfica, esta caraterística nos textos é a *WordCloud*. Estudando as imagens geradas por esta técnica, é possível uma confrontação de termos pela sua frequência. Na Figura 4.14, observam-se duas *WordClouds*, geradas a partir de documentos da coleção de texto organizado numa lista *lemmas (lemma_words_publications)*.



a) b)

Figura 4.14 - *WordClouds* de frequência associadas a dois documentos armazenados na coleção *lemma_words_sentences*.

A primeira *WordCloud* (Figura 4.14a) mostra que no documento em causa, as palavras com maior frequência são *family*, *member*, *participant* e *ocd,* por se tratar dos termos com maior dimensão na imagem. Por comparação de tamanho, pode-se verificar que palavras como *illness* ou *behaviour* não são tão frequentes no mesmo documento. Na imagem ao lado (Figura 4.14b), os termos que mostram maior número de ocorrências no documento são *participant*, *thought*, *intrusive* e *harm*.

Apesar desta metodologia permitir uma apelativa visualização dos dados, a frequência dos termos num texto não constitui uma medida fiável para encontrar a importância de uma palavra na representação do documento. Neste âmbito, surge o índice Tf-Idf, que tal como mencionado nos capítulos anteriores, devolve a significância de um termo num documento, inserido numa coleção de vários. Quanto maior for valor deste índice, superior é a relevância do termo nesse contexto.

Aos documentos apresentados na Figura 4.14, foram aplicadas as fórmulas de cálculo do Ti-Idf e extraídos os 15 termos com o índice mais elevado. Os títulos dos documentos em causa e as listas de *tokens* resultantes deste passo encontram-se representadas na Figura 4.15.

Title:

Separating obsessive-compulsive disorder from the self. A qualitative study of family member perceptions

Terms with the highest Tf-Idf:

['member', 'perception', 'family', 'husband', 'sufferer', 'spouse', 'daddy', 'person', 'behaviour', 'accommodating', 'seemed', 'coping', 'appeared', 'accommodation', 'illness']

(a)

Title:

Maternal unwanted and intrusive thoughts of infant-related harm, obsessive-compulsive disorder and depression in the perinatal period: study protocol

Terms with the highest Tf-Idf:

['infantrelated', 'intrusive', 'unwanted', 'postpartum', 'harm', 'infant', 'thought', 'perinatal', 'harming', 'pregnancy', 'maternal', 'intentional', 'mother', 'columbia', 'interviewer']

(b)

Figura 4.15 - Listas dos quinze termos com Tf-Idf mais elevado em dois documentos da coleção *lemma_words_sentences*.

Comparando os termos de cada uma das listas com os de maior frequência nas respetivas *WordClouds*, verificam-se algumas divergências. Nem todas as palavras de maior dimensão na *WordCloud*, estão incluídas na lista de palavras com maior Ti-Idf. Analisando o primeiro documento, observa-se que apesar de palavras como *ocd* e *participants* se destacarem na WordCloud, estas não surgem na lista de termos com Tf-Idf mais elevado. Uma vez que este índice é diretamente proporcional à frequência do termo num documento e inversamente proporcional à quantidade de documentos da coleção onde a palavra surge, esta diferença pode-se justificar pelo elevado número de publicações onde os termos aparecem. É normal que numa coleção de documentos sobre transtornos obsessivo compulsivos, o termo *ocd*[9] surja numa grande quantidade de documentos, reduzindo o valor do Tf-Idf deste termo. Relativamente ao segundo documento, observa-se que palavras como *pregnancy*, *intentional* ou *husband* apresentam um Tf-Idf elevado, mas não se destacam na *WordCloud*. Apesar de não serem os termos mais frequentes nos documentos, são considerados relevantes para a identificação do documento, pois não surgem com tanta frequência nas restantes publicações da coleção de documentos sobre TOC.

[9] Do inglês: *Obsessive Compulsive Disorders*

5. CONCLUSÕES

Neste último capítulo é sintetizado o trabalho desenvolvido ao longo da dissertação e apresentadas as conclusões mais relevantes. Antes de terminar, são também apontados alguns aspetos que poderão ser alvo de trabalho futuro.

5.1 CONCLUSÃO

O trabalho desenvolvido focou-se, principalmente, na implementação de um sistema automático de recolha online de documentos científicos completos, pertencentes ao setor da saúde, e o tratamento dos dados textuais extraídos. Esta automatização facilita todo o trabalho manual de pesquisa, recolha e limpeza de documentos, realizado por muitos investigadores na área da biomedicina.

Com o passar dos anos, foi-se verificando um considerável aumento no número e diversidade de documentos publicados neste setor. No sentido de permitir o acompanhamento deste crescimento bibliográfico, acabaram por surgir vários sistemas, que se dedicam à automatização do processo de análise de textos e descoberta de conhecimento, como é o caso da Mineração de Textos. No entanto, antes de qualquer estudo automático dos dados, é fundamental que a recolha dos textos não seja o resultado do procedimento manual, de pesquisa por um tema num motor de busca e extração dos inúmeros documentos devolvidos. Para melhorar esta fase de extração de documentos da web, podem ser utilizados web *scrapers*, que tornam automático todo processo de busca e recolha de documentos. Para além desta simplificação na extração de publicações médicas, é essencial que os conteúdos recolhidos sejam devidamente tratados e estruturados. A implementação de métodos de PLN nos textos extraídos permite a formatação e padronização dos dados, contribuindo assim, para melhores resultados na descoberta de conhecimento.

Os desenvolvimentos realizados neste trabalho dividem-se em duas etapas principais. A primeira fase, onde foi indispensável a conexão à internet, passou pela construção de um *Scraper*, para rastreamento, extração e armazenamento dos documentos médicos. Antes de qualquer ação, este *bot*, que teve por base a estrutura Scrapy, foi restringido ao domínio da *BioMed Central*, garantindo-se assim, que a extração de dados estivesse limitada a repositórios da biomedicina que cedessem o acesso ao conteúdo completo das publicações científicas. Após um rastreamento pelos vários *hyperlinks* contidos nas páginas visitadas, seguiram-se as fases de recolha do título, conteúdo e URL dos documentos e o armazenamento dessa informação numa base de dados adequada. Como se trata de uma grande quantidade de dados

textuais e não existe a necessidade de obter relacionamentos entre eles, optou-se por guardar os documentos recolhidos numa base de dados não relacional e orientada a documentos.

Assim que reservados os documentos na sua forma original, segue-se a segunda etapa principal do sistema. Nesta fase, foram aplicados alguns dos conceitos de Processamento de Linguagem Natural, para limpeza e tratamento dos dados armazenados. Primeiramente, foram utlizados métodos para a eliminação de carateres especiais, conversão de todo o texto para minúsculas e a expansão de palavras contraídas, seguindo-se a tokenização dos documentos nas duas formas possíveis: frases ou termos. Ao resultado da tokenização em termos foram ainda aplicadas técnicas para eliminação das palavras sem significado (*stopwords*) e por fim utilizada uma das metodologias de encurtamento de palavras: lematização ou *stemming*. Como resultado deste tratamento dos dados, foram obtidas três novas coleções na base de dados: uma com todos os documentos resultantes da tokenização em frases, outra com o produto da lematização das palavras e por fim, uma coleção com a lista de termos após a aplicação da técnica de *stemming*.

Terminado o processamento dos textos, foram ainda observadas algumas caraterísticas nos documentos. Recorrendo a metodologias de POS *tagging*, ou seja, de marcação de partes do discurso, foram categorizadas gramaticalmente as várias palavras incluídas na coleção resultante da tokenização em frases. Com o produto desta marcação, foi possível filtrar os dados pelas classes nomes e adjetivos, e encontrar dentro destas as palavras que surgem próximas mais vezes num documento.

A frequência dos termos num documento e o índice Tf-Idf são medidas importantes para o estudo da relevância das palavras. Com a primeira métrica, foi possível obter a WordCloud de cada um dos documentos armazenados na coleção de *lemmas*, onde os termos com maior dimensão estão associados a uma frequência superior. Comparando estes dados com os termos que apresentam um Tf-Idf mais elevado, confirma-se que nem sempre uma frequência alta significa que o termo representa bem o documento. A sua importância num documento também é afetada pelo número de ocorrências na restante coleção.

Por ser uma área de bastante interesse para os investigadores no setor da saúde e por se tratar de um distúrbio que afeta milhares de pessoas no mundo, foi apresentado o resultado da recolha e tratamento de documentos científicos relacionados com Transtornos Obsessivo-Compulsivos. Este estudo de caso foi fundamental para a demostração do funcionamento do sistema desenvolvido. Como resultado da primeira fase do sistema, obteve-se a extração e armazenamento numa BD de 1794 publicações relacionadas com TOC e sem qualquer formatação. Com a aplicação da segunda fase, foram obtidas

três novas coleções na BD, com os documentos organizados e estruturados de forma diferente em cada uma delas. A escolha da coleção a utilizar varia consoante a necessidade dos utilizadores e com o propósito que se pretende dar aos dados de texto.

5.2 TRABALHO FUTURO

Para complementar o sistema de recolha e tratamento de documentos projetado e desenvolvido, pode ser considerada como perspetiva futura, a aplicação de técnicas de descoberta de conhecimento nos dados resultantes. Com um estudo mais aprofundado das áreas de Mineração de Texto, Extração de Informação ou *Machine Learning*, pode ser possível a implementação de métodos e algoritmos de descoberta de novos factos, apropriados ao contexto do problema.

De entre as várias tarefas incluídas por estas áreas, a que se destaca, pela sua utilidade no estudo em questão, seria a aplicação de Regras de Associação nos dados. Estes métodos, bastante utilizados na Extração de Informação e na Mineração de Dados estruturados, permitem encontrar correlações, muitas vezes desconhecidas, entre diferentes elementos dentro de um conjunto de dados. No caso prático em análise, com a aplicação desta técnica, podem vir a ser encontradas associações entre termos que ocorrem nos diferentes documentos, dando origem à descoberta de relações entre entidades como fármacos, doenças, genes, tratamentos, sintomas, entre outros.

Para além desta metodologia de previsão de relações, seria também relevante a aplicação de tarefas de *Clustering* nos textos recolhidos. Este método seria capaz de contribuir para a organização automática de um elevado número de documentos científicos, possibilitando a atribuição de cada documento a um, ou mais, *Clusters,* deixando-os organizados em grupos.

Como ponto de melhoria do sistema desenvolvido, numa fase futura, a arquitetura de rastreamento e recolha de documentos poderia ser alargada a outros domínios de informação. Garantido apenas, que o conteúdo das publicações científicas é totalmente disponibilizado e que a área de estudo dos documentos se foca na biomedicina, podem ser rastreados mais repositórios de publicações científicas. Quanto maior for a amostra de documentos extraídos, mais completo será o estudo de determinado tema.

REFERÊNCIAS

1. Sarkar, D.: Text Analytics with Python. Apress (2016). https://doi.org/10.1007/978-1-4842-4354-1.

2. Erhardt, R.A.A., Schneider, R., Blaschke, C.: Status of text-mining techniques applied to biomedical text, (2006). https://doi.org/10.1016/j.drudis.2006.02.011.

3. McKinney, W.: Data Structures for Statistical Computing in Python. Proc. 9th Python Sci. Conf. (2010).

4. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: From data mining to knowledge discovery in databases. AI Mag. 17, 37 (1996).

5. Zhao, B.: Web Scraping. In: Encyclopedia of Big Data. pp. 1–3 (2019). https://doi.org/10.1007/978-3-319-32001-4.

6. Madnani, N.: Getting started on natural language processing with Python. Crossroads. 13, 5–5 (2007). https://doi.org/10.1145/1315325.1315330.

7. Raybaut, P.: Spyder - Documentation. Spyder 2.3 Doc. (2009). https://doi.org/http://dx.doi.org/10.1016/0388-0001(94)90016-7.

8. Wang, J., Guo, Y.: Scrapy-based crawling and user-behavior characteristics analysis on Taobao. Proc. 2012 Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discov. CyberC 2012. 44–52 (2012). https://doi.org/10.1109/CyberC.2012.17.

9. Jurafsky, D., H. Martin, J.: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, (2019).

10. Rossum, G. Van, Drake, F.L.: Python Tutorial, http://docs.python.org/tutorial/, (2019). https://doi.org/10.1111/j.1094-348X.2008.00203_7.x.

11. Borgmann, H., Wölm, J.H., Vallo, S., Mager, R., Huber, J., Breyer, J., Salem, J., Loeb, S., Haferkamp, A., Tsaur, I.: Prostate Cancer on the Web—Expedient Tool for Patients' Decision-Making? J. Cancer Educ. (2017). https://doi.org/10.1007/s13187-015-0891-3.

12. Topaloglu, H., Gumussoy, C.A., Bayraktaroglu, A.E., Calisir, F.: The relative importance of usability and functionality factors for E-health web sites. Hum. Factors Ergon. Manuf. (2013). https://doi.org/10.1002/hfm.20319.

13. Agre, G.H., Mahajan, N. V.: Keyword focused web crawler. In: 2nd International Conference on Electronics and Communication Systems, ICECS 2015 (2015).

https://doi.org/10.1109/ECS.2015.7124749.

14. Talib, R., Kashif, M., Ayesha, S., Fatima, F.: Text Mining: Techniques, Applications and Issues. Int. J. Adv. Comput. Sci. Appl. 7, 414–418 (2016). https://doi.org/10.14569/ijacsa.2016.071153.

15. Spasic, I., Ananiadou, S., McNaught, J., Kumar, A.: Text mining and ontologies in biomedicine: Making sense of raw text. Brief. Bioinform. 6, 239–251 (2005). https://doi.org/10.1093/bib/6.3.239.

16. Bollacker, K.D., Lawrence, S., Giles, C.L.: System for automatic personalized tracking of scientific literature on the Web. Proc. ACM Int. Conf. Digit. Libr. 105–113 (1999). https://doi.org/10.1145/313238.313270.

17. Uramoto, N., Matsuzawa, H., Nagano, T., Murakami, A., Takeuchi, H., Takeda, K.: A text-mining system for knowledge discovery from biomedical documents a. Ibm Syst. J. 43, (2004).

18. Lu, Z.: PubMed and beyond: A survey of web tools for searching biomedical literature. Database. 2011, 1–13 (2011). https://doi.org/10.1093/database/baq036.

19. Pellizzon, R. de F., Población, D.A., Goldenberg, S.: Pesquisa na área da saúde: seleção das principais fontes para acesso à literatura científica. Acta Cir. Bras. 18, 493–496 (2003). https://doi.org/10.1590/s0102-86502003000600002.

20. Balan, P.F., Gerits, A., Vanduffel, W.: A practical application of text mining to literature on cognitive rehabilitation and enhancement through neurostimulation. Front. Syst. Neurosci. 8, 1–14 (2014). https://doi.org/10.3389/fnsys.2014.00182.

21. Drubach, D.A.: Obsessive-Compulsive Disorder. Contin. Lifelong Learn. Neurol. 21, 783–788 (2015). https://doi.org/10.1212/01.CON.0000466666.12779.07.

22. Juang, Y., Liu, C.: Phenomenology of obsessive compulsive disorder in Taiwan. Psychiatry Clin. Neurosci. 55, 623–627 (2001).

23. Setti, S., Wanto, A.: Analysis of Backpropagation Algorithm in Predicting the Most Number of Internet Users in the World. J. Online Inform. 3, 110 (2019). https://doi.org/10.15575/join.v3i2.205.

24. AbuKausar, M., S. Dhaka, V., Kumar Singh, S.: Web Crawler: A Review. Int. J. Comput.

Appl. 63, 31–36 (2013). https://doi.org/10.5120/10440-5125.

25. Wang, J., Geng, X., Gao, K., Li, L.: Study on topic evolution based on text mining. In: Proceedings - 5th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2008 (2008). https://doi.org/10.1109/FSKD.2008.417.

26. vanden Broucke, S., Baesens, B.: Practical Web Scraping for Data Science. Pract. Web Scraping Data Sci. 155–172 (2018). https://doi.org/10.1007/978-1-4842-3582-9.

27. Ahuja, M.S., Bal, J.S., Varnica: Web Crawler : Extracting The Web Data. Int. J. Comput. Trends Technol. (2014).

28. Yadav, M., Goyal, N.: Comparison of Open Source Crawlers- A Review. Int. J. Sci. Eng. Res. 6, 1544–1551 (2015).

29. Parvez, M.S., Tasneem, K.S.A., Rajendra, S.S., Bodke, K.R.: Analysis of Different Web Data Extraction Techniques. 2018 Int. Conf. Smart City Emerg. Technol. ICSCET 2018. 1–7 (2018). https://doi.org/10.1109/ICSCET.2018.8537333.

30. Ghosh Dastidar, B., Banerjee, D., Sengupta, S.: An Intelligent Survey of Personalized Information Retrieval using Web Scraper. Int. J. Educ. Manag. Eng. 6, 24–31 (2016). https://doi.org/10.5815/ijeme.2016.05.03.

31. Mahto, D.K., Singh, L.: A dive into Web Scraper world. Proc. 10th INDIACom; 2016 3rd Int. Conf. Comput. Sustain. Glob. Dev. INDIACom 2016. 689–693 (2016).

32. S.C.M. de S Sirisuriya: A Comparative Study on Web Scraping. Proc. 8th Int. Res. Conf. KDU. 135–140 (2015).

33. Nisafani, A.S., Hendrawan, R.A., Wibisono, A.: Eliciting Data From Website Using Scrapy : an. Semin. Nas. Teknol. Inf. dan Multimed. 1–8 (2017).

34. Scrapy: Scrapy Documentation Release 1.6.0. (2019).

35. Ignatow, G., Mihalcea, R.: Web Crawling and Scraping. Text Min. A Guideb. Soc. Sci. 34–41 (2018). https://doi.org/10.4135/9781483399782.n3.

36. Kouzis-Loukas, D.: Learning Scrapy. (2016).

37. Farooq, B.: Available Online at www.ijarcs.info CRAWLING OF JAPANESE REAL-ESTATE WEBSITES USING SCRAPY. 9, 64–67 (2018).

38. Manuel, J., Santos, A.: Real Estate Market Data Scraping and Analysis for Financial Investments. (2018).

39. Han, J., Haihong, E., Le, G., Du, J.: Survey on NoSQL database. Proc. - 2011 6th Int. Conf. Pervasive Comput. Appl. ICPCA 2011. 363–366 (2011). https://doi.org/10.1109/ICPCA.2011.6106531.

40. Acharya, B., Jena, A. K., Chatterjee, J. M., Kumar, R., & Le, D.N.: NoSQL Database Classification: New Era of Databases for Big Data. Int. J. Knowledge-Based Organ. 9, 50–65 (2019).

41. Abramova, V., Bernardino, J.: NoSQL databases: MongoDB vs Cassandra. ACM Int. Conf. Proceeding Ser. 14–22 (2013). https://doi.org/10.1145/2494444.2494447.

42. Henricsson, R.: Document Oriented NoSQL Databases: Document Oriented NoSQL Databases A comparison of performance in MongoDB and CouchDB using a Python interface, (2011).

43. Kumar, L., Bhatia, P.K.: Available Online at www.jgrcs.info TEXT MINING : CONCEPTS , PROCESS AND APPLICATIONS. 4, 36–39 (2013).

44. Perkins, J.: Python 3 Text Processing With NLTK 3 Cookbook. (2014). https://doi.org/10.1017/CBO9781107415324.004.

45. Lamurias, A., Couto, F.M.: Text Mining for Bioinformatics Using Biomedical Literature. (2019). https://doi.org/10.1016/b978-0-12-809633-8.20409-3.

46. Fan, J. wei, Prasad, R., Yabut, R.M., Loomis, R.M., Zisook, D.S., Mattison, J.E., Huang, Y.: Part-of-speech tagging for clinical text: wall or bridge between institutions? Annu. Symp. Proc. 2011, 382–391 (2011).

47. Jusoh, S., Alfawareh, H.M.: Techniq Techn iq ues , Applications and Challenging Issue in Text Mining. (1988).

48. Nie, B., Sun, S.: Using Text Mining Techniques to Identify Research Trends: A Case Study of Design Research. Appl. Sci. (2017). https://doi.org/10.3390/app7040401.

49. Natarajan, M.: Role of Text Mining in Information Extraction and Information Management. DESIDOC Bull. Inf. Technol. (2014). https://doi.org/10.14429/dbit.25.4.3663.

50. Berner, M., Frick, K., Kriston, L., Loessl, B., Gann, H., Batra, A., Mann, K., Medicine, A.:

Finding the ideal place for a psychotherapeutic intervention in a stepped care approach – a brief overview of the literature and preliminary results from the Project PREDICT. Int. J. Methods Psychiatr. Res. 17, 60–65 (2008). https://doi.org/10.1002/mpr.

51. Kotu, V., Deshpande, B., Kotu, V., Deshpande, B.: Chapter 9 – Text Mining. In: Predictive Analytics and Data Mining (2015). https://doi.org/10.1016/B978-0-12-801460-8.00009-4.

52. Feldman, R., Dagan, I.: Knowledge Discovery in Textual Databases (KDT) Data Structure : the Concept Hierarchy Concept Distributions. Proc. 1st Int. Conf. Knowl. Discov. Data Min. 112–117 (1995).

53. Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E.D., Gutierrez, J.B., Kochut, K.: A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques. (2017).

54. Fleuren, W.W.M., Alkema, W.: Application of text mining in the biomedical domain, (2015). https://doi.org/10.1016/j.ymeth.2015.01.015.

55. Sathya, S., Rajendran, N.: A Review on Text Mining Techniques. Int. J. Comput. Sci. Trends Technol. 3, 274–284 (2013).

56. Jensen, L.J., Saric, J., Bork, P.: Literature mining for the biologist: From information retrieval to biological discovery. Nat. Rev. Genet. 7, 119–129 (2006). https://doi.org/10.1038/nrg1768.

57. Harpaz, R., Callahan, A., Tamang, S., Low, Y., Odgers, D., Finlayson, S., Jung, K., LePendu, P., Shah, N.H.: Text Mining for Adverse Drug Events: the Promise, Challenges, and State of the Art. Drug Saf. (2014). https://doi.org/10.1007/s40264-014-0218-z.

58. Zheng, S., Dharssi, S., Wu, M., Li, J., Lu, Z.: Text mining for drug discovery. Methods Mol. Biol. 1939, 231–252 (2019). https://doi.org/10.1007/978-1-4939-9089-4_13.

59. Aggarwal, C.C., Zhai, C.X.: Mining text data. (2013). https://doi.org/10.1007/978-1-4614-3223-4.

60. Swanson, D.R.: Fish oil, Raynaud's syndrome, and undiscovered public knowledge. Perspect. Biol. Med. 30, 7–18 (1986).

61. Zhou, S., Palma, M.: A Web Scraper For Forums. 1 (2017).

62. BioMed Central Ltd: BMC, research in progress, https://www.biomedcentral.com/.

63. BioMed Central Ltd: We're the pioneers of open access publishing, https://www.biomedcentral.com/about/open-access.

64. BMC Part of Springer Nature, https://www.biomedcentral.com/.

65. Fountoulakis, K.N., Iacovides, A., Ioannidou, C., Bascialla, F., Nimatoudis, I., Kaprinis, G., Janca, A., Dahl, A.: Reliability and cultural applicability of the Greek version of the International Personality Disorders Examination. BMC Psychiatry. (2002). https://doi.org/10.1186/1471-244X-2-6.

66. Diedrich, A., Sckopke, P., Schwartz, C., Schlegl, S., Osen, B., Stierle, C., Voderholzer, U.: Change in obsessive beliefs as predictor and mediator of symptom change during treatment of obsessive-compulsive disorder - a process-outcome study. BMC Psychiatry. (2016). https://doi.org/10.1186/s12888-016-0914-6.

67. Heimerl, F., Lohmann, S., Lange, S., Ertl, T.: Word cloud explorer: Text analytics based on word clouds. In: Proceedings of the Annual Hawaii International Conference on System Sciences (2014). https://doi.org/10.1109/HICSS.2014.231.

68. Honda, S., Nakao, T., Mitsuyasu, H., Okada, K., Gotoh, L., Tomita, M., Sanematsu, H., Murayama, K., Ikari, K., Kuwano, M., Yoshiura, T., Kawasaki, H., Kanba, S.: A pilot study exploring the association of morphological changes with 5-HTTLPR polymorphism in OCD patients. Ann. Gen. Psychiatry. (2017). https://doi.org/10.1186/s12991-017-0126-6.

69. Maremmani, I., Pani, P.P., Pacini, M., Bizzarri, J. V., Trogu, E., Maremmani, A.G.I., Gerra, G., Perugi, G., Dell'Osso, L.: Subtyping patients with heroin addiction at treatment entry: Factor derived from the Self-Report Symptom Inventory (SCL-90). Ann. Gen. Psychiatry. (2010). https://doi.org/10.1186/1744-859X-9-15.

70. Siddiqui, E.U., Naeem, S.S., Naqvi, H., Ahmed, B.: Prevalence of body-focused repetitive behaviors in three large medical colleges of karachi: A cross-sectional study. BMC Res. Notes. (2012). https://doi.org/10.1186/1756-0500-5-614.