Universidade do Minho
Escola de Engenharia

Tiago Alexandre Pinto Alves

Development of text mining tools for information retrieval and extraction from patents

Dissertação de Mestrado

Mestrado em Bioinformática

Trabalho efetuado sob a orientação/supervisão de:

Professor Doutor Miguel Francisco Pereira Rocha, Departamento de Informática, Universidade do Minho

Hugo Samuel Oliveira Costa, Head of Development for Text Mining, SilicoLife lda.

Outubro de 2016

ACKNOWLEDGMENTS

Ao professor Miguel Rocha, por toda a disponibilidade demonstrada ao longo do desenvolvimento deste projeto para esclarecer as minhas dúvidas, ajustando a minha conduta e propondo soluções para todos os problemas que fui encontrando e que, à primeira vista, pareciam não ter solução possível. Agradeço também todos os contributos feitos para que fosse conferido maior rigor científico a esta dissertação.

Ao Hugo Costa, o grande "supervisor na empresa" que através de toda a sua disponibilidade e interesse no projeto permitiu que muitos problemas fossem ultrapassados, quer relativos aos primeiros passos no processo de aprendizagem de programação em Java, quer relativos à organização de todo o trabalho efetuado. De igual forma, agradeço também todos os contributos para o melhoramento de algumas afirmações menos rigorosas que por vezes surgiram na escrita desta dissertação.

Agradeço também a toda a equipa da SilicoLife pela rápida integração que proporcionaram e pela oportunidade de trabalhar em conjunto com excelentes pessoas com um sentido de humor muito apurado e que contribuíram com energia positiva para qualquer dia mais árduo ao longo deste percurso. Em especial, gostaria de agradecer ao Rúben toda as dúvidas de programação que me foi esclarecendo com uma prontidão incrível e à Ana, por solucionar alguns problemas "logísticos" que complicariam o desenvolvimento desta dissertação, sendo uma excelente amiga e colega de casa.

Ao Nuno, pelo companheirismo e por toda a boa disposição, estando sempre disponível para ouvir algumas dúvidas que me iam atormentando e por ir dando a sua opinião, mesmo não estando dentro do projeto.

Aos meus amigos, por me perdoarem todas as ausências justificadas com todo o trabalho inerente à elaboração deste projeto e por saberem fazer-me levantar a cabeça, sorrindo sempre.

Ao Jorge, por ser um amigo impecável que muitas vezes se prejudicou para conseguir ajudar-me com alguns dos vários problemas que encontrei ao longo deste trabalho, tendo sido fundamental com as suas inúmeras opiniões acerca da escrita desta dissertação.

À Camila, por todo o apoio incondicional e por estar sempre disponível para ouvir as minhas lamúrias, argumentando sempre com o brilho nos olhos que não consegue deixar-me indiferente. Agradeço-lhe toda a alegria que tem depositado na minha vida e por ser um porto seguro repleto de boas energias.

Por último e não menos importante, agradeço à minha família por terem incutido em mim todos os valores que me transformaram naquilo que sou hoje e por me apoiarem em tudo aquilo que está e no que não está ao alcance deles. São, sem duvida alguma, das pessoas mais importantes para mim e, sem eles, muito provavelmente esta dissertação não existiria.

A todos, muito obrigado!

RESUMO

A literatura biomédica é constituída por um número alargado e em crescimento de publicações escritas em linguagem natural. As patentes, uma fração integrante das referidas publicações, têm vindo a ser consideradas importantes fontes de informação, uma vez que possuem informação curada resultante do seu processo de atribuição. Apesar de serem consideradas verdadeiras bibliotecas tecnológicas, a sua informação não estruturada transforma a procura de informação nesses textos uma tarefa deveras desafiante. A mineração de textos biomédicos é um campo científico que explora esta tarefa, criando metodologias para a pesquisa de informação estruturada em literatura biomédica.

A obtenção de informação é uma tarefa integrante do processo de mineração de textos biomédicos, na qual a informação relevante é obtida de uma extensa coleção de documentos usando diversas metodologias. O processo de obtenção de toda a informação contida numa patente requer o *download* do respetivo ficheiro PDF que posteriormente é convertido em texto passível de ser lido por máquinas recorrendo a tecnologias de processamento tais como o reconhecimento ótico de carateres (OCR).

Neste projeto, um sistema de obtenção de informação e um sistema de conversão de PDF em texto foram desenvolvidos dando origem a uma ferramenta de tratamento de patentes que foi integrada no @note2, uma plataforma computacional de código aberto usada para a mineração de textos biomédicos. A *pipeline* elaborada pode ser desintegrada em quatro diferentes funções: pesquisa de patentes, obtenção de meta-informação das mesmas, obtenção dos seus ficheiros em formato PDF e a extração de todo o texto desses documentos.

Um conjunto de patentes do desafio BioCreative V CHEMDNER foi usado para testar a ferramenta desenvolvida, avaliando o seu desempenho e a sua real capacidade de obtenção das patentes e todo o processo de extração de informação das mesmas. Os resultados são promissores, aproximando a comunidade científica da informação disponibilizada nas patentes publicadas, permitindo a posterior implementação de outros processos da mineração de textos biomédicos a esses documentos.

Palavras-chave: Mineração de textos biomédicos; patentes, obtenção de informação, reconhecimento ótico de caracteres, @note2

ABSTRACT

Biomedical literature is composed of a large and ever increasing number of publications, written in natural language. Patents are a relevant fraction of these publications, considered important sources of information due to all the curated information available in the documents, from the granting process. Although being real technological libraries, their unstructured data turns the search of information within these documents a challenging task. Biomedical text mining is a scientific field that explores this task, creating methodologies to search and structure the information in the biomedical literature.

Information retrieval is one of the biomedical text mining tasks, in which the relevant information is obtained from an extensive collection of documents using several text retrieval methodologies. Getting all the information available on a patent document requires the download of the respective PDF document, that is then converted into a machine-readable text by technologies as *Optical Character Recognition* (OCR).

In this project, an information retrieval, and a PDF to text conversion system were developed building a "patent pipeline" which was integrated into @note2, an open-source computational framework for biomedical text mining. The patent pipeline can be disintegrated into four different tasks: the patent search, the retrieval of patent metadata, the retrieval of their PDF files, and the extraction of all the information from these documents.

A set of patents from the BioCreative V CHEMDNER task was used to test the developed pipeline, evaluating the framework performance and the real capacity to retrieve the requested patents and extract their unstructured information. The results were promising, bringing to the scientific community the published patent information and allowing the posterior implementation of other biomedical text mining processes over these documents.

Keywords: Biomedical text mining, patents, information retrieval task, *Optical Character Recognition*, @note2

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Acronyms

API - Application Programming Interfaces

BioTM - Biomedical Text Mining

CEMP - Chemical Entity Mentions in Patents

CM - Corpora Module.

CPD - Chemical Passage Detection

DP - Dynamic Programming

DPI - Dots Per Inch

EPO - European Patent Office

FBPN - Feed Forward Back Propagation Neural Network

FN - False Negative

FP - False Positive

GPRO - Gene and Protein Related Object

HTTP - Hypertext Transfer Protocol

IE - Information Extraction

IPC - International Patent Classification

IR - Information Retrieval

IUPAC - International Union of Pure and Applied Chemistry

JPO - Japan Patent Office

JSON - JavaScript Object Notation

ML - Machine-Learning

MVC - Model-View-Controller

NCBI - National Center for Biotechnology Information

NER - Named Entity Recognition

NLP - Natural Language Processing

OCR - Optical character recognition

OPS - Open Patent Services

OSCAR3 - Open-Source Chemistry Analysis Routines 3

PCT - Patent Cooperation Treaty

PDF - Portable Document Format

PHA - Polyhydroxyalkanoate

PHB - Polyhydroxybutyrate

PHH - Polyhydroxyhexanoate

PMM - Publication Manager Module

PNN - Probabilistic Neural Network

POS - Part-of-Speech

PPV - Positive Predictive Value

RE - Relation Extraction

REST - Representational State Transfer

RM - Resources Module

SMILE - Simplified Molecular Input Line Entry System

SOAP - Simple Object Access Protocol

SVD - Singular Value Decomposition

TN - True Negative

TP - True Positive

URI - Uniform Resource Identifiers

URL - Uniform Resource Locators

USPTO - United States Patent and Trademark Office

WIPO - World Intellectual Property Organization

1. INTRODUCTION

1.1. Context and Motivation

Nowadays, we are living in a digital era with lots of textual sources to look for, what makes this process a challenging task. Looking only at the life sciences, we get a huge number of new publications, research reports and patents every year (Klinger *et al.*, 2008).

Biomedical Text Mining (BioTM) techniques have been applied to different types of publications to automate the extraction of high quality and structured information from these texts, being in the recent years, a growing interest in patents (Cohen and Hunter, 2008).

Patents are important sources of information since they have detailed descriptions of inventions rarely replicated in other publications visible to the general public, being real technological libraries (WIPO, 2015c). For instance, the number of registered patents by year on World Intellectual Property Organization (WIPO) database increased from 1702900 in 2005 to 2680900 in 2014 (WIPO, 2014; WIPO, 2015d). Handling these large amounts of information turns the search and extraction of data into a difficult and time-consuming task. So, it has become a necessity to bring up machine assisted knowledge techniques to this kind of sources (Oldham *et al.*, 2013).

There are several available patent databases such as Espacenet, that is provided by the European Patent Office (EPO) or PATENTSCOPE[1], that is provided by the WIPO. Using only these two databases, it is possible to search information or to download the patent published files for more than 90 million patents.

Patent databases are organized into specific areas of knowledge which make the search process easier. The information from these patent databases can be accessed in text format or in Portable Document Format (PDF) files. Those files have often encoded information on images with complex backgrounds or even different sizes and orientations, making text extraction a difficult task to do (Patel *et al., 20*12).

[1] (https://patentscope.wipo.int/search/en/search.jsf)

Optical character recognition (OCR) tools have many applications based on the conversion of printed text into editable text. This method mimics the human optical mechanism and the ability to read documents even when the words are in complex environments (Patel *et al.*, 2012).

BioTM techniques become necessary to automate the search for structured information in these natural language oriented texts since this is a task that made by hand becomes quite time-consuming, similar to what happens with other sources of biomedical literature (Shatkay and Feldman, 2003).

Over the last few years, the Biosystems research group (University of Minho) and the SilicoLife company have worked in the BioTM field, developing @Note2, a Java multi-platform BioTM Workbench which uses a relational database (as MySQL) and is based on a plug-in architecture, allowing the development of new tools/methodologies in the text mining field, both in information retrieval or information extraction (Lourenço *et al.*, 2009). This software and its core libraries, in its current version @Note2, will be used as the basis for this work, once it can be upgraded and improved.

1.2. Aims

In this work, the main goal is the development of a text mining pipeline that enables the search of patents for the given input keywords including the retrieval of texts from distinct patent databases/repositories and evaluates different options to conduct the PDF to text conversion.

In detail, the scientific/ technological objectives are:

- Review some available search engines that allow the searching for patents;
- Review some tools used for patent retrieval (metadata and PDF files);
- Develop tools using the available search engines for the patent searching process;
- Develop tools for the retrieval of patents data from relevant databases;
- Improve PDF to text conversion using OCR tools;
- Build a framework named “patent pipeline” using the developed tools to search and retrieve patents data as well as the improved PDF to text conversion.
- Validation with gold standard corpora and if possible in real world scenarios (e.g. BioCreative Challenge).

2. STATE OF THE ART

2.1. Biomedical Text Mining

Huge amounts of information are generated every day due to technology evolution (Liu *et al.*, 2015; Faro *et al.*, 2012). Looking only at the life sciences, the already huge number of texts available on public databases increases a lot producing an exponential growth of information in the last few years (Figure 1) (Faro *et al.*, 2012; Hunter and Cohen, 2006).

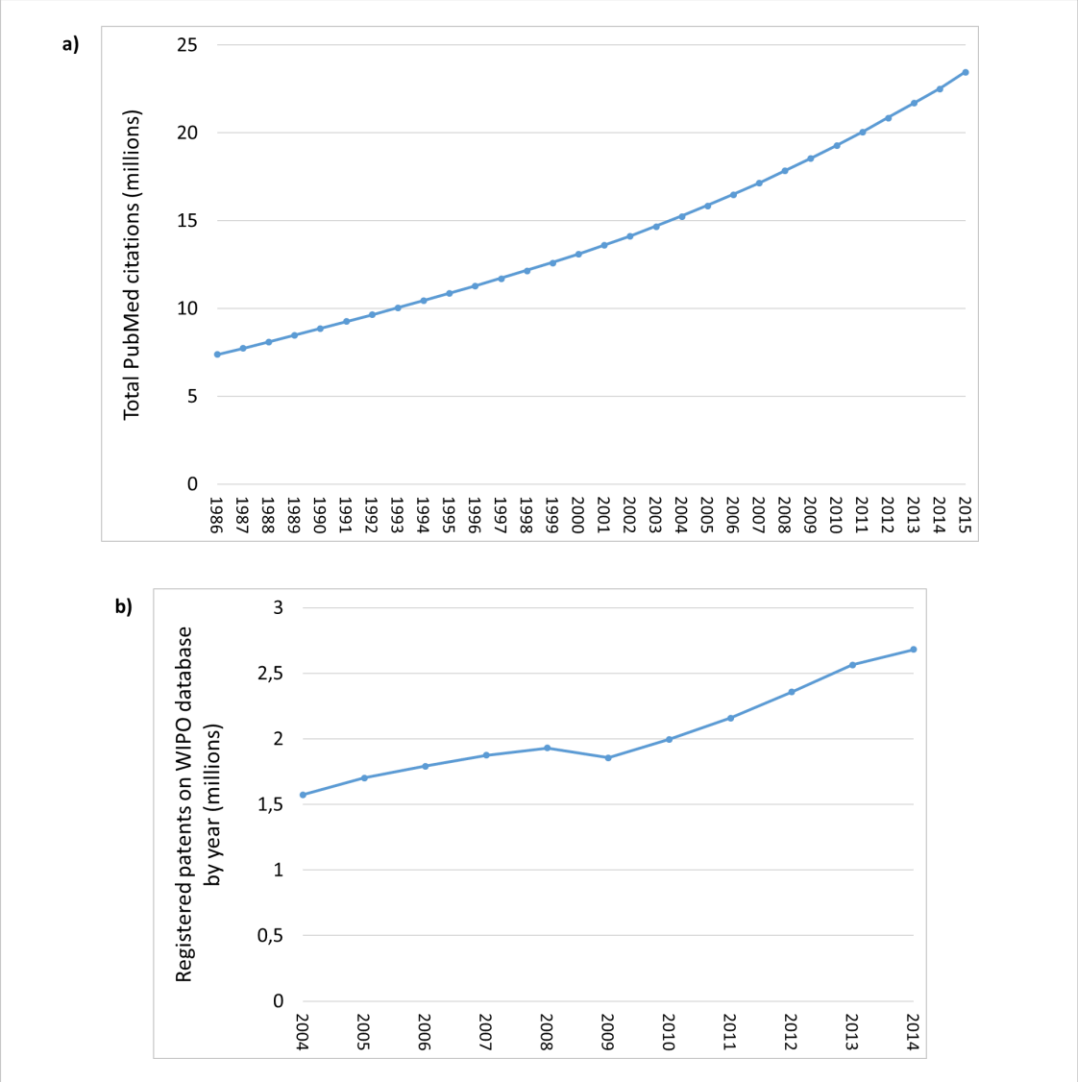


Figure 1 - Growth of scientific publications. a) represents the growth of the total PubMed citations from 1986 to 2015 ((USNLM, 2016) and B) represents the growth of the registered patents on WIPO by year (WIPO, 2014; WIPO, 2015d).

For instance, the World Intellectual Property Organization (WIPO), that is one of the most well-known world patent databases, has an annual increasing rate around 5% and PubMed, which is the most used database concerning biomedical literature with almost 27 million records, has an annual increasing rate around 4% (Wu *et al.*, 2015; Lu, 2011; WIPO, 2015d). However, the unstructured nature of that information makes impossible to analyze all the available literature manually (Wu *et al.*, 2015; Zweigenbaum *et al.*, 2007).

To exploit this information, a field named Biomedical Text Mining emerged. That field allows the automation of text processing and the extraction of meaningful knowledge from it (Cohen and Hunter, 2008).

2.1.1. Concepts and definitions

Text Mining is based on many different knowledge areas such as statistics, artificial intelligence, computer science, management science, machine learning (ML), among others (Miner *et al.*, 2012). As we can see in Figure 2, Text Mining is the combined result of these external fields of knowledge with numerous text analytics components as Information Retrieval (IR), Information Extraction (IE), Natural Language Processing (NLP), among others (Miner *et al.*, 2012).

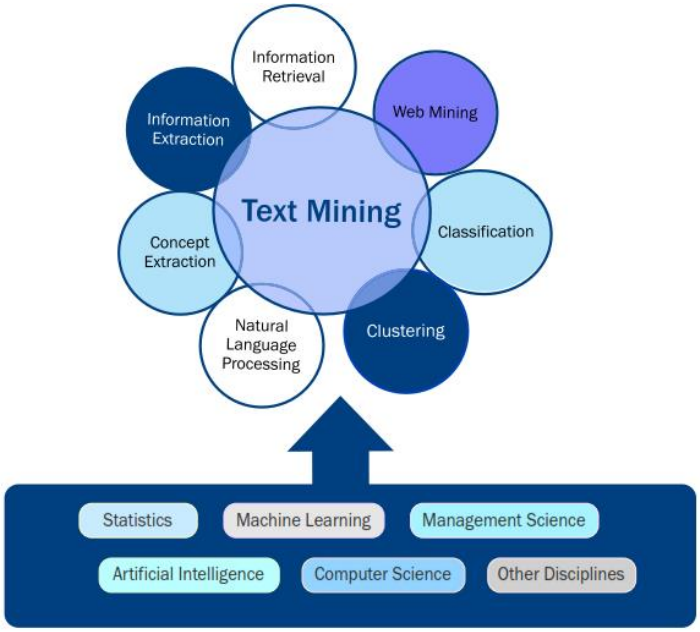


Figure 2 - Text mining, the text analytical components contribute and the knowledge gained from many external areas. Adapted from (Miner *et al.*, 2012).

The Text Mining field applied to molecular biology or biomedical literature is named Biomedical Text Mining (BioTM). BioTM explores all the literature available on public repositories with information related to almost every gene, protein, and other molecules intervenient in biological processes (Mathiak and Eckstein, 2004; Shatkay and Feldman, 2003).

BioTM can improve the research process either by the discovery of new facts, the data interpretation or even the design of experiments. It correlates genes or proteins with a specific disease, reconstructing or even predicting different pathways, finding specific biological functions for genes, among others (Shatkay and Feldman, 2003).

BioTM can be segmented in information retrieval (IR), which allows obtaining relevant information resources (e.g. papers or patents) from an extensive collection of documents, and information extraction (IE) which allows the extraction of pertinent information from the documents on unstructured form through the use of computer processing (Krallinger and Valencia, 2005).

As can be seen in Table 1, the scientific community has been developing several BioTM platforms based on the use of several programming languages using IR, and IE methodologies (Table 1).

Table 1 - Some BioTM platforms, their programming languages and a brief review on BioTM functionalities.

BioTM platform	Programming Language	BioTM functionalities
@note (Lourenço *et al.*, 2009)	Java	Information retrieval and extraction of biomedical information from literature, detecting entities and relations between them in raw text.
iHOP (Hoffmann and Valencia, 2004)	PHP, AJAX, and Javascript	Information retrieval from MEDLINE abstracts related to protein interactions and genes information.
Neji (Campos *et al.*, 2013)	Java	Identification of bio-entities in free text (information extraction).
ABNER (Settles, 2005)	Java	Tagging genes, proteins and other bio-entities in free text (information extraction).
GATE (Cunningham *et al.*, 2013)	Java	Identification of chemical and drug bio-entities as well as chemical formulas in free text (information extraction).
MyMiner (Salgado *et al.*, 2012)	PHP, AJAX, and Javascript	Web application for bio-entity tagging (information extraction).

2.1.2. Text retrieval (IR process)

When handling textual data, the first thing to do is to access a database with a large collection of literature and extract only the relevant documents for the specific case study. Shatkay and Feldman (2003) classified text retrieval in four different types of searching: Boolean queries and index structures; similarity queries and the vector model; latent semantics indexing; and text categorization (Shatkay and Feldman, 2003). Using Boolean queries, the user/machine gives different keywords connected by a Boolean term like "and", "or", "not" (among others) as input. This type of search is supported by an index of all terms in all documents allocated on the database. When a given term is searched, the documents where the input keyword (or a combination of words) appears are organized by the query frequency (Figure 3). This type of search has several limitations since it returns a great number of documents that can be irrelevant and may not retrieve all relevant documents.

The second search type, based on similarity queries and the vector space model, is more concise than the Boolean query, since each term (which can be a single word or a longer phrase like "blood pressure") is introduced into a vector denoted by ***M***. Every document is represented as a vector with dimension ***M,*** where each element is the weight for a given term on that document.

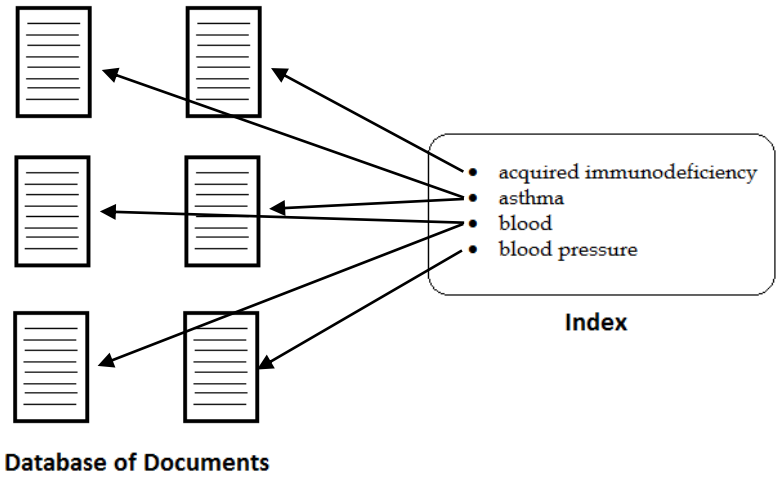


Figure 3 - Index relating the searched terms to the documents in which they occur. Adapted from (Shatkay and Feldman, 2003).

The calculation is performed by term frequency or significance and normalized for document length, enabling the addition of small but quite informative documents to the results. The frequency of documents containing a specific term is also taken into account in the weight calculation, making possible to favor rare query terms, since they are more likely to be informative than the most frequent terms. After this normalization, it becomes possible to predict the distance between the query vector and the multiple document vectors. For that purpose, different metrics can be used, being the Euclidean distance the most well-known. Finally, using distance metrics, it is possible to achieve similarity through the cosine of the angles between two vectors (Figure 4).

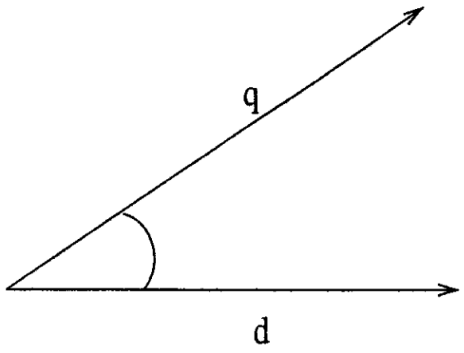


Figure 4 - The angle between two vectors, queries (q) and documents (d). From (Shatkay and Feldman, 2003).

Latent semantics indexing type of search is used to overtake problems as polysemy and synonymy, where a given term can have multiple meanings or where different terms can have the same meaning, respectively.

This type of search uses also ***M***-dimensional vectors representing the documents in relation to the input terms, similarly to the previously described search type. However, these vectors are aggregated into a matrix which is the base for the application of an algebraic operator (singular value decomposition (SVD)), which allows the identification of relations between the terms and documents using the semantic of document words.

Finally, on the text categorization search type, the user can define a set of rules, allowing the classification of documents in different categories, or using machine learning to automatically perform this classification (Shatkay and Feldman, 2003).

Krallinger and Valencia (2005) rank the text retrieval search in a more simplistic way with only two different types of search: the document-based search and the query-based search. On document-based search, the input query is a document and the algorithm searches for documents which are similar to the input document, returning a list ordered by similarity. The query-based search type uses an algorithm with keywords as input query and searches the literature for documents with a high frequency of these keywords. It returns a list based on that frequency similar to the Boolean queries described above. The used keywords can be combined with Boolean operators turning the query more complex and precise

Nowadays, there are several retrieval tools that implement the described search types to provide the requested information to the user/machine. So, to search for biomedical literature, the most used retrieval tool is the GQuery, a global cross-database provided by US National Center for Biotechnology Information (NCBI)[2] which is accomplished with a text-based search mechanism and a retrieval system which supports the document-based and query-based search described by Krallinger and Valencia (2005). Similar to GQuery, *Google Scholar* provided by Google[3] is also a widely used retrieval tool, while other literature search engines as CrossrefSearch[4] or the Nature Publishing Group search engine[5] are less used. These tools can be evaluated by their output in two different parameters: sensitivity (number of relevant documents retrieved on all relevant documents existents) and specificity (number of relevant documents retrieved on all documents retrieved) (Hunter and Cohen, 2006).

2.1.3. Information extraction (IE process)

After the IR process getting a machine-readable text, the next key step is to analyze it and try to understand the provided information. This process can be done by different techniques of IE, which essentially work in three stages: recognition of terms; finding the structure of the sentence; or finding meaning and logic relations between terms or sentences. For that, IE techniques are often accomplished by first applying Natural Language Processing (NLP) processes (Thessen *et al.*, 2012).

[2] (http://www.ncbi.nlm.nih.gov/)
[3] (http://scholar.google.com)
[4] (http://www.crossref.org/crossrefsearch.html)
[5] (http://www.nature.com/search/?sp_a=sp1001702d&sp_t=advanced&sp_x_1=ujournal&sp-p=all&sp)

NLP includes a set of computational and linguistic methods that allows the application of NLP on several systems as speech-to-text transcription, spelling correction, or even the indexing of documents on a database allowing the search process as described in the previous section. Applied to BioTM field, NLP processes are commonly used for the extraction of information from natural language texts allowing the identification of words and the true meaning of them, the assignment of the syntactic structure of expressions, among so many others (Clark *et al.*, 2010).

From the numerous NLP tasks, the more frequently used within IE methodologies are the *sentence splitting task*, which allows the delimitation of the sentences boundaries, splitting the text using expression rules to analyze punctuation, word capitalization or breaks (Bach and Badaskar, 2007); the *tokenization* process, which breaks raw text into words (tokens) through splitting strings by whitespaces and then by symbols or punctuation; and the *part-of-speech (POS)* or the *lemmatization* processes, where each token is tagged into their lexical group (verb, adjective, noun, among others) or by their canonical form, respectively.

2.2. Patents

A patent is a validated document representing the intellectual property rights of an invention granted by a national government or international organization, giving to the author exclusive rights to use, manufacture or sell that for a determined period of time. In this kind of document, there is information that may not be found anywhere else, due to novelty nature implicit on patents and due to the fact that only a small fraction of that information is subsequently disclosed in journal articles or other scientific literature (WIPO, 2015a; Latimer, 2005). This fact makes patent data critical to understanding several technological areas (WIPO, 2015a).

Patent documents are organized into several sections. Following a more general classification, patents are divided into three main sections: a Front Page with bibliographic data, providing the document number, the applicant (the inventor) or the person with inventor's right (the assignee), the inventor itself, the filing and publication dates, the classification (scientific or technological area), citations and codes (e.g. country codes); a Description, providing a brief, clear and concise summary of the invention's technical background and how that invention solves a specific problem, describing the essential features of the invention, which may be represented

through drawings; and a Claims section, the definition of the matter for which protection is required (WIPO, 2015c; WIPO, 2015a).

Inventions from the biological field can be, for instance, kits for the manipulation or use of nucleic acids or proteins, diagnostic kits, chemicals for the pharmacological industry, software for bioinformatics analysis, production of food, medicine methodologies, among others (Latimer, 2005).

Patents are classified into hierarchical categories by patent offices. The International Patent Classification (IPC) is the most common system with patents grouped into eight sections : "A" representing "human necessities"; "B" representing "performing operations and transporting"; "C" representing "chemistry and metallurgy"; "D" representing "textiles and paper"; "E" representing "fixed constructions"; "F" representing "mechanical engineering, lighting, heating, weapons and blasting", "G" representing "physics" and "H" representing "electricity" (WIPO, 2015b). Inside each section, patents are divided into classes (e.g., A61 representing medical or veterinary science/hygiene) and each class contains subclasses (e.g., A61K representing preparations for medical, dental or toilet purposes). These subclasses are also grouped into main groups (e.g., A61K 38/00 representing medicinal preparations containing peptides) and some of these main groups can be divided into subgroups (e.g. A61K 38/16 representing peptides having more than 20 amino acids) (Eisinger *et al.*, 2013).

2.2.1. Patent databases and retrieval systems

There are several ways of getting patent documents since they are available in numerous databases. The most used databases are described in more detail in Table 2.

There are several methods that can be used for patent extraction from the presented patent databases. The webpage links provided in Table 2 can be useful for the extraction of a small number of patents files in an easy way, depending only on how comprehensive the research is desired. For localized searches, databases as j-PlatPat from JPO or as PatFT from USPTO are suitable to use but for searches in worldwide patent documents, databases as PATENTSCOPE from WIPO or esp@cenet from European Patent Office (EPO) are more appropriate (Table 2).

Table 2 - Some patent databases, the correspondent URL, and a brief description. Adapted from (Latimer, 2005).

Name	Description	URL
United States Patent and Trademark Office (USPTO) Patent Full-Text and Image Database (PatFT)	For searching and printing full-text and full-page image US patents and published US applications.	http://www.uspto.gov/patft/index.html
esp@cenet from European Patent Office (EPO)	For searching and printing worldwide patents and patent publications	http://worldwide.espacenet.com/
j-PlatPat from Japan Patent Office (JPO)	For searching and printing Japanese patents and patent publications	https://www.j-platpat.inpit.go.jp/web/all/top/BTmTopEnglishPage
PATENTSCOPE from World Intellectual Property Organization (WIPO)	For searching and printing international Patent Cooperation Treaty (PCT) applications	https://patentscope.wipo.int/search/en/search.jsf
Google Patents	For searching and printing world patents and published applications	https://patents.google.com/

However, various authors have attempted to use alternative approaches for the extraction of large volumes of data due to the difficulty imposed by this process. Heifets and Jurisica (2012) built SCRIPDB (a chemical structure database) which used USPTO bulk download from Google servers available since 2010 (Heifets and Jurisica, 2012). Papadatos *et al.* (2016) used patent data from USPTO, EPO, WIPO besides titles and abstracts from the JPO processed in XML format to build SureChEMBL a database with compounds. For patent data processing, Papadatos *et al.* (2016) used IFI Claims[6], a third-party global patent database aggregating patent information from over 40 national and international data sources into a single repository (Papadatos *et al.*, 2016).

Another alternative is using secondary databases containing only portions of all patent documents normally in a limited space of time (Pasche *et al.*, 2014). A practical example of this is NBER[7] database.

[6] (http://www.ificlaims.com/index.php)
[7] (http://www.nber.org/patents/)

2.2.2. Text mining over patents

Given the characteristics presented by patents and the interest that their information awakened on the research and development branch, several studies were triggered using patent information. As examples, Laffleur (2016) described mucoadhesion, mucoadhesive polymers and mucoadhesive drug delivery systems using patent literature as one of the information sources (Laffleur, 2016). Chen and colleagues (2015) presented a search strategy for the identification and classification of a vaccine to Influenza Virus using EPO patents information (Chen *et al.*, 2015).

Although the great informative capacity of patents, processing all the information available on them manually becomes a quite time-consuming task (Jeong and Kim, 2014). To work around that situation, text mining tools have been applied to texts from patents in the same way as for the biomedical literature in scientific fields such as biology and chemistry. Oldham *et al.* (2013) used text mining tools to find Latin species names of a 6 million terms list on 11 million patents published between 1976 and 2010 in WIPO and EPO (Oldham *et al.*, 2013).

The information described in the free text about chemical compounds, drugs, and other chemical entities have been the target of interest by the scientific community, both from the academic and industrial points of view (Xu *et al.*, 2015). That information allows the identification of unique chemicals, indexing bibliographic chemical databases or even the discovery of which chemical compounds are related to a given biological processes. Due to this growing interest, there are an increasing number of publications addressing patent text mining in the chemical field.

To validate the performance of patent text mining methods, there is an increased interest in manually annotated patent corpora and in suitable training/test data (Akhondi *et al.*, 2014). Therefore, to achieve that goal, the BioCreative CHEMDNER tasks were created (Campos *et al.*, 2015; Krallinger *et al.*, 2015; Krallinger *et al.*, 2013).

The BioCreative CHEMDNER tasks are based on the implementation of IE processes to several chemical publications, finding chemical mentions on papers or even, more recently, on patent documents. In these new CHEMDNER approaches there are three possible tasks: the recognition of chemical named entities in patents (Chemical Entity Mentions in Patents task (CEMP)); the classification of chemical-related patent titles and abstracts (Chemical Passage Detection task (CPD)); and the recognition of genes and proteins in patent abstracts (Gene and Protein Related

Object task (GPRO)). These tasks are run using a development, a training, and a test corpus, which were created previously by domain experts and classified by recall (*r*), precision (*p*) and *F1* measures (Krallinger *et al.*, 2013).

Recall, also named sensitivity or coverage, the percentage of correctly classified positive results (True Positive (TP)) over all positive set (True Positive + False Negative (TP+FN)) is defined by the equation:

$$r = \frac{TP}{TP + FN}$$

Precision, also named Positive Predictive Value (PPV), is the percentage of correctly labeled positive results over all examples classified as positive (True Positive + False Positive (TP+FP)):

$$p = \frac{TP}{TP + FP}$$

The F1 measure is the harmonic mean between precision and recall and is used as overall classifier:

$$F1 = 2 * \frac{p * r}{p + r}$$

For these tasks, some contributions were made by Akhondi *et al.* (2015) which presented a system able to recognize chemical entities in patents and classify chemical-related patent titles and abstracts using an ensemble system combining dictionaries with some improvements to tmChem, a ML-based system for chemicals developed by Leaman *et al.* (2015).

Concerning the manually annotated patent corpora creation, some contributions also have been made. Akhondi *et al.* (2014) developed a large gold standard chemical patent corpus using 200 full-text patents from WIPO and EPO, annotating chemicals and all their sub-entities like diseases, targets or modes of action of compounds.

Similarly, Kiss *et al.* (2012) developed a chemical patent corpus using the claims section of 313 patents from the class A61K, which includes preparations for medical, dental or toilet purposes. The subsequent analysis was restricted to 62 claims (Kiss *et al.*, 2012).

These curated corpora can then be used to test the text mining tools enforced on patents as the tool described by Papadatos *et al.* (2016) (the SureChEMBL). In this system, after getting patent information, the full text is tokenized and analyzed for named chemical entities using IE classifiers.

After the IE process, the system submits the chemical entities to name-to-structure tools to get the structure of each individual chemical. That allows the users to search for chemicals using a chemical structure or substructure as a query or even a combination of both, allowing the retrieval of the relevant patent literature (Papadatos *et al.*, 2016). Developing a system like that requires that the entire process has to be tested. That is why gold standard corpora are so important.

Other related works were conducted. Kemp and Lynch (1998) described a system which is capable of extracting generic structure descriptions from the text of patent abstracts in an autonomous way using a dictionary matching system after tokenizing and then a series of heuristics to fulfill the rules of nomenclature systems as the International Union of Pure and Applied Chemistry (IUPAC), created to simplify the writing process and standardize all chemical names (Kemp and Lynch, 1998; Campos *et al.*, 2015).

Sayle *et al.* (2012) applied text mining techniques in patents with pharmacological interest getting over typographical problems (human spelling errors, hyphenation or even line breaking) associated with compounds as IUPAC names. After the IE process, chemical names are used as input in a name-to-structure software similar to SureChEMBL (Sayle *et al.*, 2012).

Heifets and Jurisica presented SCRIPDB, a portal which allows the extraction of chemical structures from patents using text mining without losing important information, as chemical relationships (Heifets and Jurisica, 2012). Jessop *et al.* (2011) developed a system named PatentEye. Although this system is a prototype, it allows the extraction of chemical reactions from the patent literature creating representations able to be understandable by any machine. Initially, patents are identified from EPO online archive depending on user's search and downloaded in XML format type. After that, these documents are semantically enhanced and reactions were extracted using OSCAR3 (Open-Source Chemistry Analysis Routines 3), a tool for chemistry-specific parsing of chemical documents (Jessop *et al.*, 2011).

Not always the information extracted from the patent sections comes in natural language format. Frequently, this information comes from encrypted file formats, usually images in BMP, TIFF, PNG or GIF. When it is wanted to extract any kind of information from this type of file such as chemical structures, it is necessary to convert these images into structured representations of standard chemical files format (such as Simplified Molecular Input Line Entry System (SMILE)

strings) (Park *et al.*, 2009). To convert that encrypted text to machine-readable text, methodologies as Optical Character Recognition (OCR) can be used.

2.3. OCR

OCR is a computer science technology that mimics the human capacity of seeing things, associating the image of a character to its symbolic identity. OCR uses pattern recognition, artificial intelligence and computer vision to the conversion of encrypted file formats as images, printed text or even handwritten text into machine-coded, readable, editable and searchable data widening the content of these files. These data can be stored in a more compact way and used in processes as machine translation, text-to-speech or even text mining (Asif *et al.*, 2014).

One of the first OCR approaches was developed in 1914 by Emanuel Goldberg. He created a device for blind people that read characters and converted them into standard telegraph code (text-to-speech method) (Asif *et al.*, 2014). More recently, with the same concern, Cutter and Manduchi (2015) developed an experimental software to analyze the acquisition of OCR-readable images by a blind person using her smartphone's camera. That software gives inferences to the user about camera position, proximity to paper and other variables that can inhibit a successfully taken well-framed image (Cutter and Manduchi, 2015).

The character recognition may be done in two different ways (online and offline), being OCR a system which uses an offline approach, in which the recognition is done after the completion of writing or printing and not done when characters are draws (Álvarez *et al.*, 2015).

2.3.1. OCR Methods

OCR systems are based on patterns. Patterns can be numbers, letters or any punctuation character. The first thing to do is teach the machine about these patterns. For that, there are training sets with all characters grouped into classes. These classes are then used by the machine to analyze and compare new examples, assigning them to the best-scored class (Eikvil, 1993).

The complete process can be resumed to two main processes: the character extraction where the learned patterns are applied to delimit words or individual letters that are then identified by the character recognition process (Holley, 2009).

OCR Character extraction

After getting the document ready to be used as input, it is necessary to locate the words or characters inside the document and distinguish them from figures or other page sections without any information. This process is called segmentation (Eikvil, 1993). Most of the times, in a first approach, segmentation algorithms identify words which are then fragmented into single characters (Eikvil, 1993). However, there are several problems associated with this recognition process since single characters can be linked to each other; background noise can be identified as dots or accents; figures or graphics can be identified as text, among others (Figure 5).

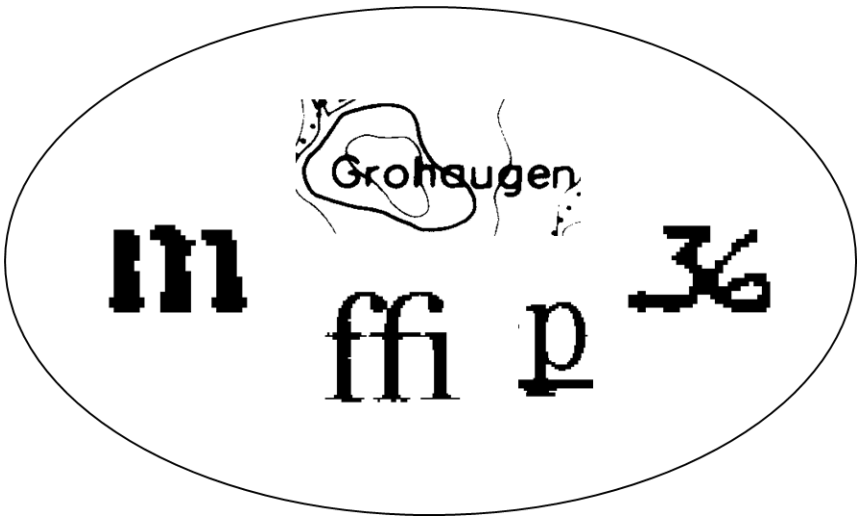


Figure 5 – Representation of degraded symbols. Adapted from (Eikvil, 1993).

After segmentation and correspondent text extraction, the resulting image of each character may contain some noise. So, a preprocessing phase is required. For that, small gaps are filled and the width of lines is reduced allowing the increase of each character intelligibility. In addition, normalization can be applied to make size, slant and rotation uniform. After this, the feature

extraction step is applied. In this process, the information that is more relevant for classification purposes is extracted from the preprocessed images. This allows minimizing the variability between the different created classes (Trier *et al.*, 1996). This is a very complex process since there is a great variability in the extracted features due to the variability of the available data or even due to recognition problems (Hossain *et al.*, 2012).

As evidenced by Eikvil (1993), there are several techniques for feature extraction that are evaluated by their robustness (sensitivity to noise, distortions, style variation and translation or rotation movements) and practical use (speed of recognition, the complexity of implementation and independence of other supplementary techniques).

Some feature extraction techniques as *template matching and correlation*, *the statistical distribution of points*, *transformations*, and *structural analysis* will be described.

In *template matching and correlation* techniques there is no true feature extraction. In this method, a matrix containing the image of the input character is used as "feature vector". Then, it is compared with the training examples used to represent each possible class, allowing the calculation of a distance or a similarity value. The character is assigned to the class where the distance value is lowest and below a specific threshold or where the similarity value is the greatest and above a specific threshold (Trier *et al.*, 1996).

Based on *statistical distributions of points* there is a feature selection using different statistical approaches. The features can be related to the density of the black points within some regions resultant from the specific division of the rectangle that delimits the character; with the density of black points surrounding a selected center; with the vectors in different directions (rows and columns), with different lengths (when distance is in use) created by the number of transitions from the character background to foreground, and vice-versa; with the relative joint occurrence of black and white points; or even with the number of intersections of the vectors created for each character point (Hossain *et al.*, 2012; Eikvil, 1993).

The complexity of these two previous described techniques allows some tolerance to distortions and style variations.

Other techniques as *transformations* and series expansions use the curve that describes the delineation of the characters through Fourier domain frequency, allowing the tolerance to the

background noise of this approach (Granlund, 1972). When the character has finer details, the Fourier domain frequency is high and for the basic shape the Fourier domain is low (Hossain *et al.*, 2012).

Finally, there are *structural analysis* techniques, where the input character is partitioned. In these techniques, for each partition, all the features extracted are the geometric and topological structures of characters, allowing the extraction of features with high tolerance to noise and style variations but weakly tolerant to rotation or translations.

OCR Character Recognition

After feature extraction, the next step is the assignment of each character to the correspondent class. Two approaches can be made: the first includes all methods that use a numerical vector as a feature to achieve the similarity value for each class and the second includes the methods that compute the similarity function using the physical structure of the character, comparing it with the physical structure of the class (Eikvil, 1993).

On the first approach, there are several classifiers: *minimum distance*, *neural networks*, and the *statistical* classifiers. *Minimum distance* classifiers use different metrics to calculate the distance value between each class and a specific character, being the Euclidean distance the most common one. A practical example of this classifier is the k-Nearest Neighbor classifier which allows obtaining competitive results depending on the *k* value and on the used metric to the distance calculation (Hossain *et al.*, 2012).

Neural network classifiers are made by layers of interconnected elements whereby the feature vector can be used as input. In every element of a given layer, a new value is calculated to the input vector through a non-linear function adjusted in the training phase. A practical example of this classifier is the feed forward back propagation neural network (FBPN) (Eikvil, 1993; Hossain *et al.*, 2012).

Statistical classifiers use Bayesian methods that calculate the probabilities of a given character belongs to any class. Then, that input character is assigned to a class with higher probability. To optimize the classification results, the Normal distribution is used as density function and it is assumed that all classes have the same occurrence probability. This classification scheme

gives the lowest probability of making classification errors, in terms of average. A practical example of this classifier is the probabilistic neural network (PNN) (Eikvil, 1993; Hossain *et al.*, 2012).

The second approach uses structural methods, being the methods that use syntactic analysis the most used ones (Eikvil, 1993). On these methods, the grammar of each class is compared with structural components from an unknown character in a practical way.

After all characters being classified into their respective classes, a post-processing phase is required. In this phase, the individual characters are associated into words or numbers based on their location within the document. In the post-processing phase, the context is used to find errors from the classification process and correct them. This process can be done by analyzing the probability of a given sequence of characters appearing together (e.g. a "k" after an "h" is an error, in the English language) or using dictionaries for all words that must appear in the text (Eikvil, 1993).

The steps of the overall OCR process regarding the character extraction and recognition processes are presented in Figure 6.

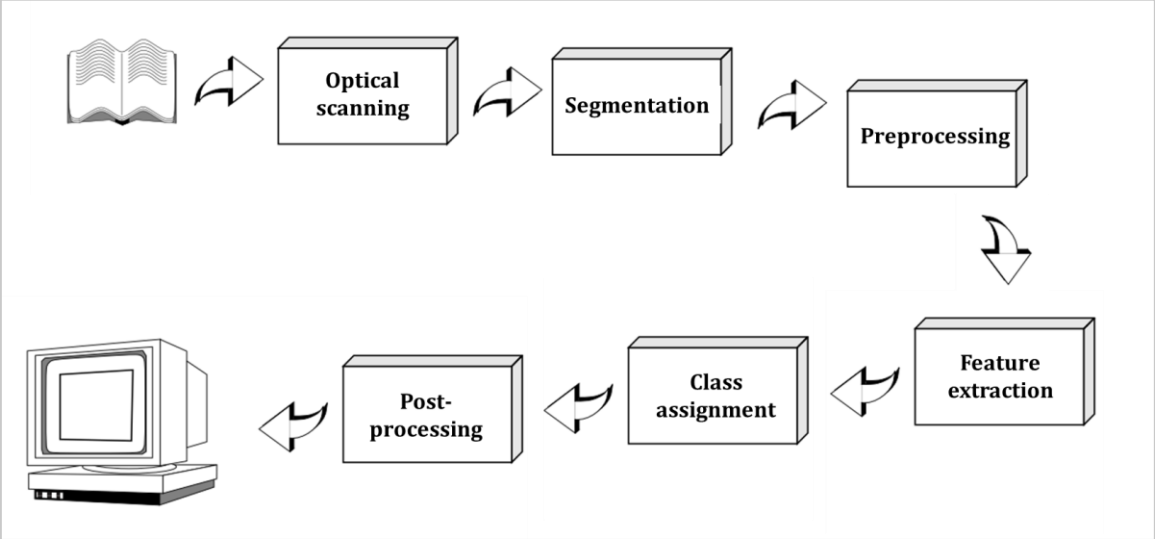


Figure 6 – Components of an OCR-system. Adapted from (Eikvil, 1993).

2.3.2. Available OCR systems

There are several types of OCR software available in the market depending on the users' needs. Generally, OCR tools can be divided into two different types – online services and desktop software. However, only a few of them are open source and available for free use. Some of the most used OCR software were represented in Table 3.

Table 3 – Some of the most used OCR systems and respective URL, type and the programming language used to apply the system.

OCR software	Programming Language	Type	URL
Tesseract	C++	Desktop (open source)	https://github.com/Tesseract-ocr
OCRopus	Python	Desktop (open source)	https://github.com/tmbdev/ocropy
GOCR (JOCR)	JavaScript	Desktop (open source)	http://jocr.sourceforge.net/
ABBYY Finereader	Not applicable	Online/Desktop	http://www.abbyy.com/
Adobe Acrobat Professional version	Not applicable	Desktop	http://www.adobe.com/products/acrobatpro.html

Tesseract is a free and open source software, written in the C++, so it is platform independent and can be applied to other languages as Java. *Tesseract* works in a step by step manner following the steps presented earlier with some particularities. First, the image is converted into binary images. The character outlines are extracted and converted into blobs. These blobs are analyzed and divided, by the spaces (even the little-fuzzed ones), into words. In a first attempt, these words are used as input and the recognized ones are used as training data for the adaptive classifier. After this training phase, the classifier is used to identify and classify the other words (Patel *et al.*, 2012; Smith, 2007).

OCRopus is another open source OCR system (Breuel, 2008). This system is written in the Python programming language and has three major components: the physical layout analysis, text

line recognition, and statistical language modeling. In physical layout analysis text columns, text blocks, text lines and reading order are identified to allow the recognition on each text line for posterior recognition of sentence text. To do that, a prior knowledge of the language, vocabulary, grammar, and the domain of the document is previously created. As a statistical method, the *OCRopus* software uses Bayesian classifiers for the classification process (Breuel, 2008).

There are more available systems but less used such as *GOCR (JOCR)* or paid ones, as *ABBYY Finereader software* or *Adobe Acrobat Professional version*.

2.4. @note2

The @note project (www.anote-project.com) is a Biomedical Text Mining platform created by the Centre of Biological Engineering at the University of Minho, in collaboration with the company SilicoLife[8] that currently maintains it. The first version of this project has been released in 2009 (Lourenço *et al.*, 2009), being now on the version 2.0.

The @note2 platform is totally written in Java which gives portability, programming facilities and robustness to the system, allowing run the code on any computer, catch runtime exception for the errors that may occur and even use several application programming interfaces (API) easily configurable and user-friendly (Hugunin, 1997).

Structurally, @Note2 is organized into core libraries and GUI tools. The core libraries are organized mainly in three main functional modules: the Publication Manager Module (PMM), the Resources Module (RM) and the Corpora Module (CM) (Figure 7).

The PMM module is subdivided into two different processes: the IR Search and the IR Crawling processes. The IR Search process allows searching documents in online repositories: PubMed and Springer. From this process, a list of documents is retrieved according to some configurable parameters for each online repository. The obtained list is a data structure designated as *Query*.

[8] (http://www.silicolife.com/)

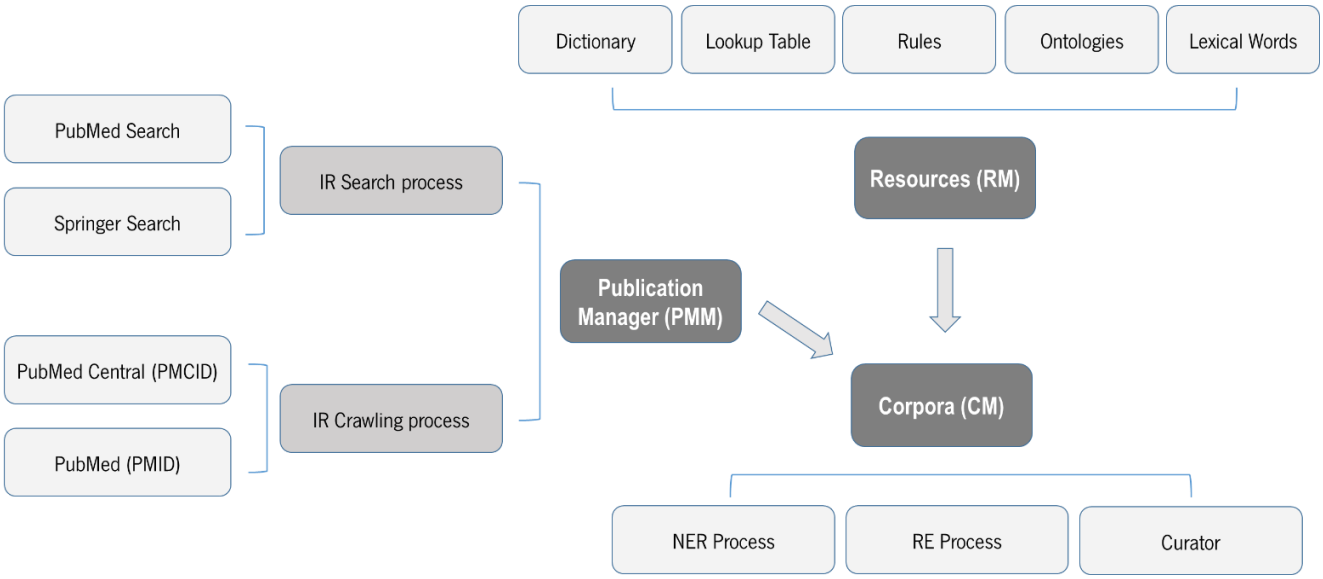


Figure 7 - @note2 structure.

The IR Crawling process is used to try to complete the documents information previously downloaded by the IR Search process with the respective full-text content. For that, specific database IDs are used. On @note2, there are two ways to achieve full-text documents: using PMCIDs from PubMed Central database or using PMIDs from PubMed database (Figure 7).

The CM allows the corpora management, being responsible for the corpora creation. That corpora can be obtained from the publications retrieved by the PMM on the IR Search process, from PDF files imported from a folder or from the evaluation corpora loading process. To each corpus, can be applied several IE processes allowing the identification and extraction of biological entities and their relationships, creating an IE schema with all the obtained results that is associated with the respective corpus were IE process was applied. The applied IE schema depends only on the IE process applied: Named Entity Recognition (NER) processes give origin to NER schemas and Relation Extraction (RE) processes give origin to RE schemas. CM also encompasses the Curator process, a powerful environment which allows the manual curation of the produced IE schemas by the addition, edition or removal of entities or relations generated previously.

RM comprises several interfaces that allow the creation and management of dictionaries, rules, lookup tables, ontologies or lexical words that are the lexical resources that can be used in IE processes by CM.

@note2 uses a relational database (as the MySQL database) and is built over the AIBench framework (Lourenço *et al.*, 2009). Consequently, it presents specific characteristics of this framework: interoperability, by allowing the model-view-controller (MVC) design pattern with different functionalities, which may come from the integration of components of other open-source platforms wrapping that content into standardized formats; flexibility, by allowing new arrangements and different configurations to get distinct applications; and, modularity, by promoting a plugin-based platform, allowing the development and integration of new components by establishing dependencies between them.

For developers, @note2 can be the base for the development of new services, algorithms or graphical components following the clear separation between core libraries and the GUI tools, the graphical section of @note. Therefore, developers can create new @note2 functionalities which can be applied to BioTM field. At the same time, the components of @note2 can be used in the development of other applications.

3. PATENT PIPELINE DEVELOPMENT

This chapter comprises all conceptual structure as well as the Java implementation of the developed pipeline.

All development process can be organized into several components that were introduced on several components integrated on the @note2 as part of the PMM and CM modules. Alongside with the patent pipeline description, this chapter encompasses several information about every used component, such as its operability and configuration.

3.1. Overview

Taking into account the proposed objectives for this project, a pipeline encompassing several steps was designed. That pipeline is referred as "patent pipeline" and can be organized into four different tasks. The first task is related to patent ID searching, being followed by patent metadata retrieval, while the third encompasses the published patent PDF file downloading, and, finally, the fourth deals with the application of PDF to text conversion methodologies to the PDF files.

The patent metadata is related to its bibliographic data, comprising the invention title, authors, publication date, an external link to a patent database entry (if available), the abstract and a short patent description (sometimes these are merged).

The patent pipeline uses different information sources grouped by their purpose. Sources to search and retrieve patent IDs, sources to search for metadata about each patent and sources capable of returning the patent file(s) in PDF format were put in the *search sources module*, *metainformation sources module* and *retrieval sources module*, respectively. The used PDF to text conversion methodologies were organized in the *PDF conversion module.* With that approach, on the patent ID searching process, the total number of patents is the result of the sum of all used components while for the other implemented processes, the patents that were not processed by a given component are used as input on the next component and so on until all patents being processed or all the components were used. The overall view of the conceptual architecture is provided in Figure 8.

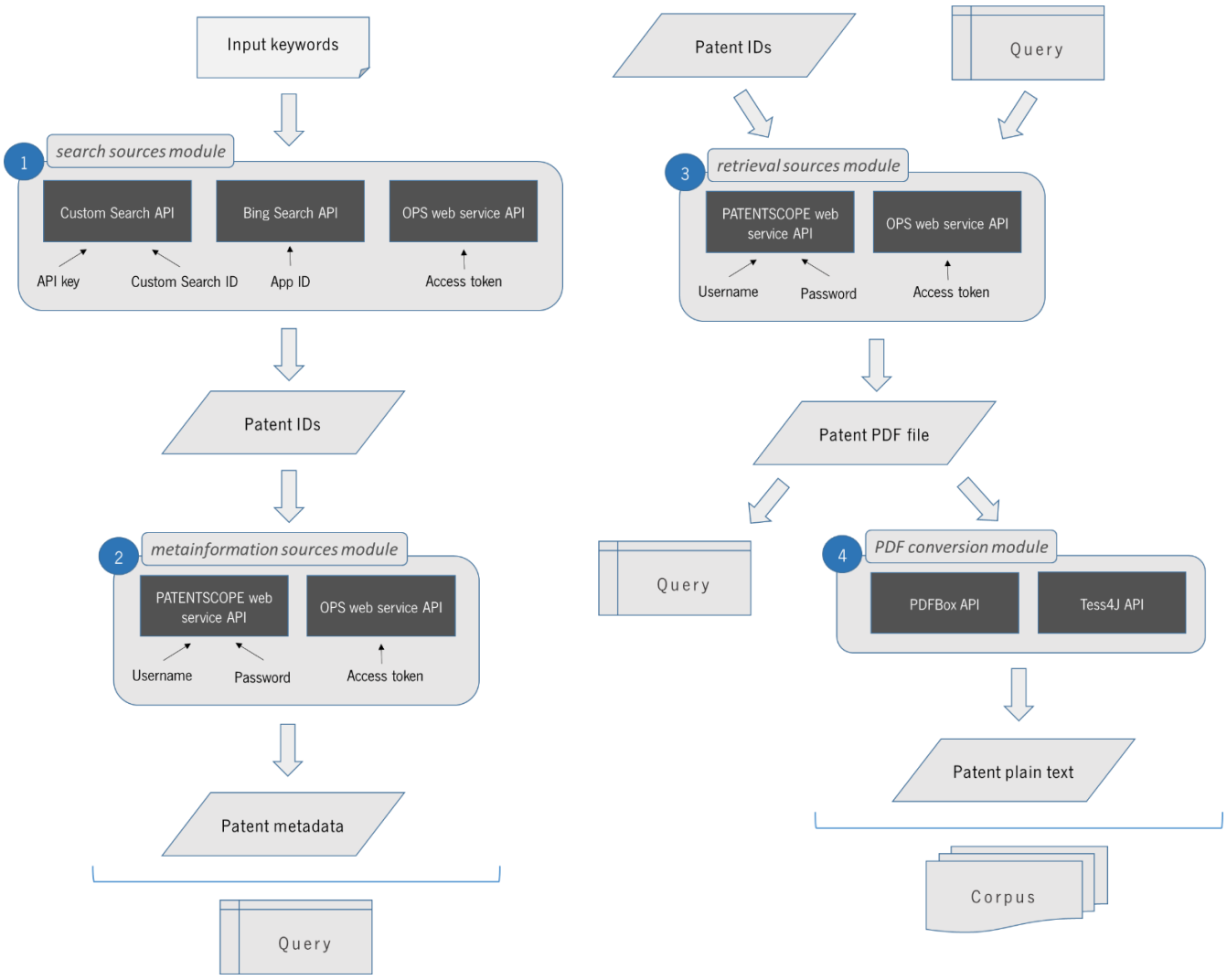


Figure 8 - Graphical summary of the designed patent pipeline. The numbers represent the pipeline flow.

As seen in Figure 8, to start the search process, input keyword(s) are needed. Those keywords can be biomedical entities as chemicals, genes, diseases, among others.

After the input definition, the keywords are processed by the *search sources module.* In this project, this module is constituted by several APIs of popular search engines, namely the *Custom Search* API from Google, the *Bing Search* API from Microsoft and the *Open Patent Services (OPS) web services* API from EPO. Each one of these module components requires specific access keys resulting from the respective services registration. These access keys along with the given input keywords are the base for getting access to the server and returning a set of patent IDs (Figure 8).

The *metainformation sources module* uses the returned patent IDs from the *search sources module* and the component specific access keys as input, returning metadata related to bibliographic information for the given patents. In this project, two different APIs were chosen to be the constituents of this module: the *PATENTSCOPE web service* API and the *OPS web service* API.

The first two modules of the patent pipeline (*metainformation sources module* and the *search sources module*) can be grouped, leading to the query creation. The query is a data structure implemented on @note2 used to store information about documents resulting from a search process.

The *retrieval sources module* takes the patent IDs from the query generated by the two previous modules of the patent pipeline (or a list of patent IDs added manually) and the component specific access keys as input, returning the PDF files of the published patents, which are added to the query data. In this project, that module is constituted by the same APIs used on *metainformation sources module* (the *PATENTSCOPE web service* API and the *OPS web service* API) using different configurations.

The PDF conversion module takes all the PDF files returned from the retrieval sources module or from the manual adding, extracting all printed text from these patent files and transforming it into a machine-readable and editable text. That text allows the creation of a corpus data structure, allowing to run IE methods, for instance, NER or RE (Figure 9).

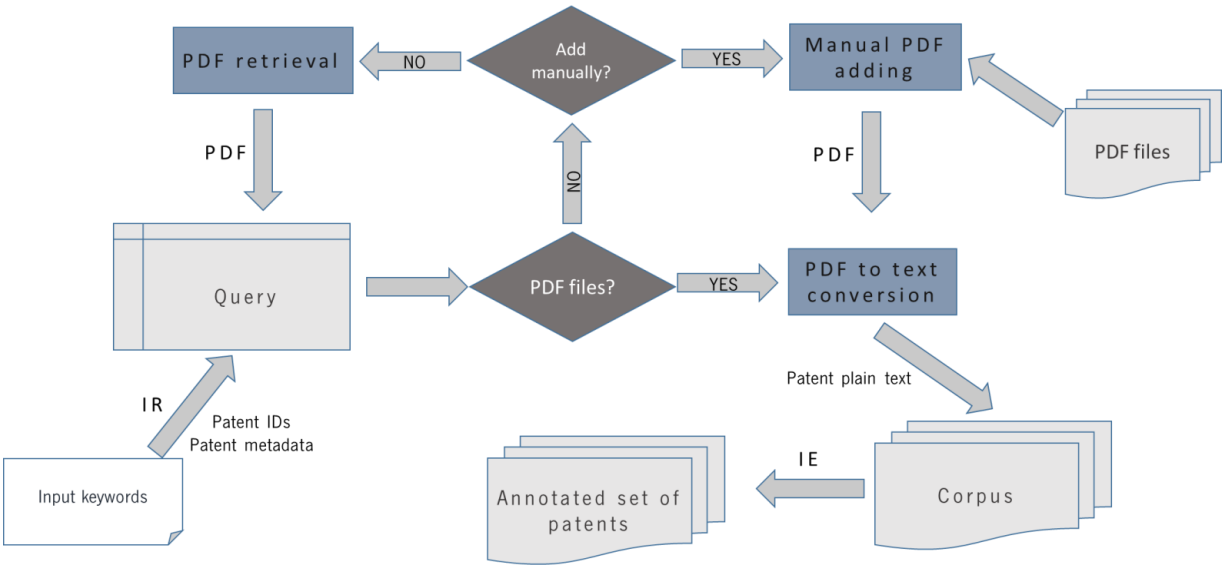


Figure 9 - Graphical summary of patent pipeline's necessary and alternative steps to get a patent corpus to be annotated.

Due to the patent pipeline modules aggregation, the patent ID searching, as well as patent metadata retrieval, are integrated on "Patent Search", a new @note2 IR Search process. The modules responsible for these tasks are detailed below in sections 3.2 and 3.3.

The patent PDF file downloading is integrated on @note core libraries as an IR Crawling process (Figure 10).

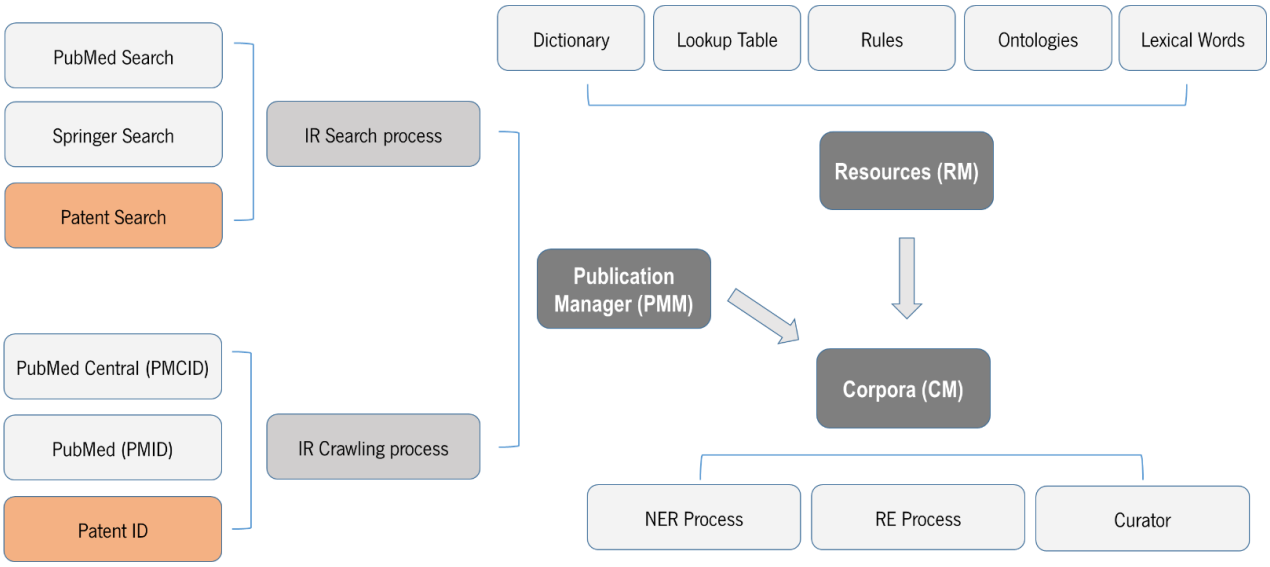


Figure 10 - @note structure with patent pipeline implementations. The orange boxes represent the new components added.

The PDF to text conversion methodologies are incorporated into the corpora module as a pre-process to the corpus creation process using a query from PMM or whenever PDF files are loaded from a folder.

The modules relative to patent retrieval and PDF to text conversion tasks are described in sections 3.4 and 3.5.

3.2. Patent IDs search process

The first stage of the patent pipeline corresponds to the input of different keywords on the *search sources module*, the respective processing and the consecutive returning of patents IDs associated with the given terms.

The given input can be simple with only a search term or comprising several terms. As an example, to search all patent IDs associated with polyhydroxybutyrate (PHB), a polymer with interesting biodegradable properties for application in the industry (Reddy *et al.*, 2003), the input can be simple, with terms such as "polyhydroxybutyrate" or just "PHB". However, since PHB is a polyhydroxyalkanoate (PHA) (polyesters synthesized by bacteria of various hydroxyalkanoates) another possible search can be all PHA who were not PHB, such as polyhydroxyhexanoate (PHH) polymers. In this case, the input must be expressed in more than one term combined by search operators (e.g. "PHA NOT PHB").

It is also possible to search for patent IDs using different PHA polyesters in combination with the search operators AND or OR. The AND operator is used only when patents containing references to both searched terms are required (e.g. "PHH AND PHB"); the OR operator is used to return patent IDs of patents containing at least one of the search terms (e.g. "PHH OR PHB"). To differentiate the search operators from other search words, they should always be entered in upper case.

All components of the *search sources module* are based on information exchanges through the internet using different Uniform Resource Identifiers (URI), a compact sequence of characters that identifies an internet resource (Berners-Lee *et al.*, 2005). A well-known subset of URIs are the Uniform Resource Locators (URLs), which give the resource location along with the resource identification.

The keywords are introduced on the specific part of the component URI. Since any URI is composed by a limited set of characters, an encoding step is needed to convert the invalid input characters to a valid ASCII or UTF-8 format. That encoding process is called percent-encoding or URL-encoding. So, after input expression definition, the invalid characters are replaced by the correspondent numerical value of character's binary code (details about the percent-encoding process applied to the search sources module components are available in Appendix 1).

Each component of the *search sources module* has specific invalid characters. For the *Custom Search* API and *Bing Search* API, almost all modifications are applied using the *URLEncoder*[9], a Java utility class for the HTML form encoding. For the *OPS web services* API only some modifications are required (EPO, 2015).

[9] (https://docs.oracle.com/javase/7/docs/api/java/net/URLEncoder.html)

3.2.1. Custom Search API

The *Custom Search* API is provided by Google and returns the same results of Google's web search through code implementation. This API works through a *Representational State Transfer* (REST) architecture (Google, 2015). As mentioned by Fielding (2000), REST systems consist of a coordinated set of components, connectors, and data elements, being the underlying architectural principle of the web. It allows complex interactions between server and client without exposing server's information besides what was requested and without the server having or saving any information about the client. This allows returning the same results for two different requests from the same client asking for the same server's information.

Systems using REST architecture and agreeing with its principles are designated as RESTful systems. Those systems communicate with the server over Hypertext Transfer Protocol (HTTP) calls, as GET (to obtain information), POST (to create a new entry with new information), PUT (to replace an entry information), and DELETE (to delete information) (Fielding, 2000).

The *Custom Search* API can retrieve results including the URL, result's title, text snippets, information about the requested search (e.g. the time that the server took to return the results or the number of results) and some information about the used *Custom Search* engine (Google, 2015). The results can be returned in two different formats: *JavaScript Object Notation* (JSON) or *Atom*. The JSON format assembles all the data about results in a collection of name-value pairs or in an ordered list of values using a language-independent data interchange format (international standard ECMA-404) (ECMA, 2013). The *Atom* format is an XML-based document format that contains all results grouped in lists of information, known as "feeds". Every "feed" illustrates a different result constituted by different data grouped on different "entries" (Nottingham and Sayre, 2005).

To use this API, Google requires two different access tokens: search engine ID and API key. The search engine ID allows the control of daily queries and the number of the returned results, while the API key allows the client identification (Figure 8). To get access to the server, the API uses a GET command through a URI allowing the access to resources (Figure 11).

Figure 11 - Example of the used URI on Custom Search API to retrieve patent IDs from the server related to PHBs.

The protocol used to communicate with the server, the host name, the Google's service and version part of URI are invariable (Figure 11). The "parameters" section is replaced by different available parameters that can be used to improve and filter results being always separated by the "&" symbol on the URI (Google, 2013). The "q", the "key" and the "cx" parameters are mandatory, while the others are optional (Table 4).

Table 4 - The most important and the most used parameters of the *Custom Search* API (Google, 2013).

Parameters	Description
q=	Defines the input expression to search for.
key=	Identifies the API key.
cx=	Defines the used *Custom Search* engine ID.
exactTerms=	Defines words that all results must contain.
excludeTerms=	Defines words that exclude results containing them.
orTerms=	Defines additional terms to search. Results with at least one of the defined additional terms are returned.
relatedSite=	Defines a site which filters the results for only those that have relationships with it.
siteSearch=	Defines a site to search returning only the results within that domain.
num=	Defines how many results per page will be returned. The maximum allowed number is 10.
start=	Defines the index of the first result presented on a given page.
alt=	Defines the response data format. It can be represented as "JSON" or "*Atom*".
cr=	Restricts results to documents originated on client country.
dateRestrict=	Restricts results by choosing a number of days (d[number]), weeks (w[number]), months (m[number]) or years (y[number]) for results' date.

Besides the API parameters shown in Table 4, the input expression can be a combination of keywords and symbols, designated as search operators. The API uses the same search operators as Google's search engine (Table 5).

Table 5 - Brief description of the most important *Custom Search* API search operators (Google, 2016).

Symbol	Description
+	It allows adding new mandatory keywords, returning only results that contain all of them.
OR	Return results that contain at least one of the given terms.
-	Allows adding new terms that exclude results containing them.
""	Defines a filter returning only results containing the exact words within the marks.
*	It is used as a placeholder of unknown terms on a phrase.
..	It is used to get results with numbers in a range.
~	Returns synonyms for a given term.
site:	Defines a domain for which all results must be included.
define:	Returns only definitions for the given term.
filetype:	Defines a file extension for results.

After the search process, the obtained results are URL addresses to patent pages from the *Google Patents* database. To get only the patent IDs, a result transformation is required (Figure 12). All patent IDs are composed of one or two letters followed by a sequence of numbers, ending with the kind code (one letter, in many cases, followed by a number). So, a regular expression is applied to every result removing all unnecessary URL parts (Figure 12).

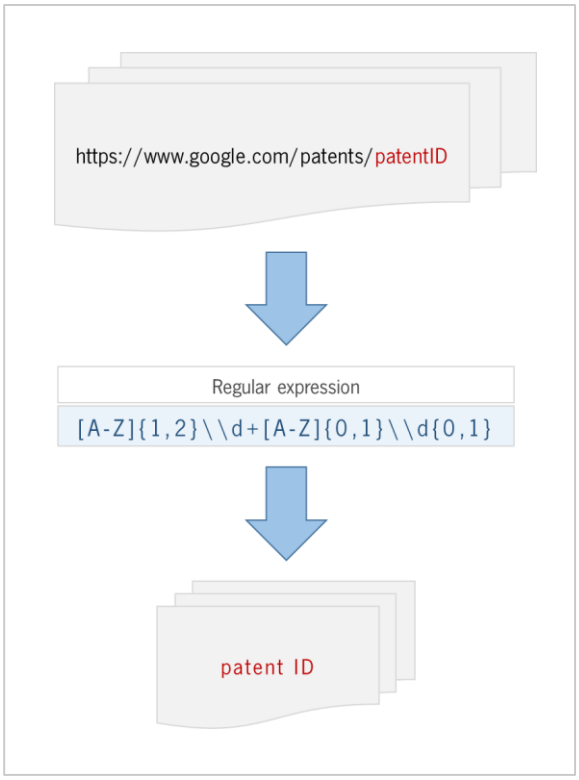


Figure 12 - Post-process applied to search process results.

3.2.2. Bing Search API

The *Bing Search* API programmatically allows obtaining the same results provided by *Bing Search engine* from Microsoft through the same output formats of the *Custom Search* API: JSON or *Atom*. Using this API, it is possible to search on many different information types using the same input keywords: the web, images, news, video or even related searches and spelling suggestions (Microsoft, 2012).

To communicate with the server, this API uses a *Hypertext Transfer Protocol Secure* (HTTPS) which is an encrypted communication protocol which combines HTTP with *Secure Sockets Layer* (SSL) certificate, preventing non-authorized access to information (Singh and Kumar, 2016).

Similar to the *Custom Search* API, the *Bing Search* API allows the introduction of some parameters/filters as URI parts (Figure 13) and search operators applicable to the keywords.

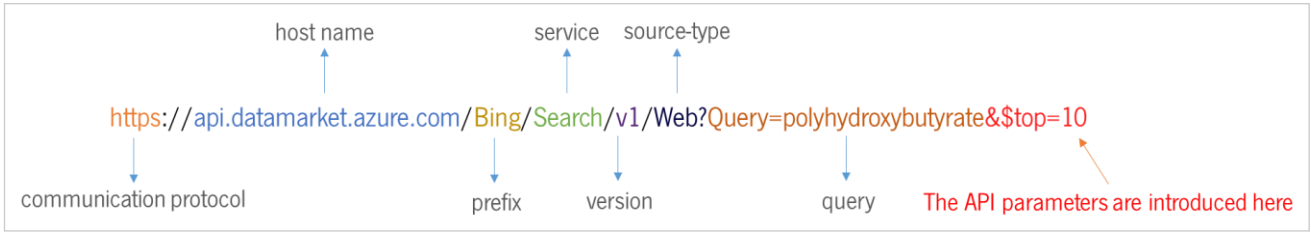


Figure 13 – Example of the used URI on *Bing Search API* to search for 10 patent IDs related with PHBs.

The most used API parameters and query search operators are presented in Table 6 alongside a brief description for each one.

Table 6 – The description of the most relevant URI parameters and the most used search operators of *Bing Search* API (Microsoft, 2012). The API parameters in italic are presented through the exact URI representation.

	Parameter	**Description**
API parameters (URI modifications)	query	Defines the input keywords to the search process.
	source-type	Allows the selection of the type of results to be retrieved. The predefined value is "Web" but it can be also "Image", "Video", "News", "Spell" and "RelatedSearch".
	$format=	Defines the server's response format. It can be in *Atom* or JSON.
	$top=	Defines the number of results to return per page. The maximum number allowed is 50.
	$skip=	Defines the index of the first result returned.
	market=	Defines the language and the country to focus results on.
Search operators (keywords modifications)	AND or &	Add terms to query, returning only results that contain all terms.
	NOT or –	Exclude results that contain the given terms.
	OR or |	Return results that contains one of the given words.
	site:	Return only web pages inside the given domain.
	""	Returns only the results which contain the exact words within speech marks on the webpage content.
	+	Return webpages which contain all given terms preceded by this symbol.
	prefer:	Help to give emphasis to a given term. The first results returned are those containing the defined term.
	filetype: or ext:	Returns only the results with the extension given.

Although the API has the ability to return results with several information types (Table 6), only web results are considered here. Each result provides a title, a brief description and the specific patent external URL which redirects to the invention's web page from the *Google Patents* database, similarly to the *Custom Search* API (Microsoft, 2012). After getting that URL, a post-processing step is required to extract only the patent ID. Since the web pages have the same URL structure than the obtained on *Custom Search* API, the result modifications are the same for the two APIs (Figure 12).

3.2.3. OPS web service API

The *OPS web service* API, as the name implies, is a web service accessed by programming in a machine-to-machine service based requests. *OPS web services* provide access to EPO's raw data from the same sources as *Espacenet* (EPO, 2015).

The results are obtained in the XML format, but can also be obtained in JSON and the communication with the server is made using a RESTful architecture similar to Google's *Custom Search* API. Contrarily to the previously discussed APIs, the *OPS web service* API allows obtaining directly patent IDs avoiding the further processing of the results. This API allows returning other patent information as metadata (bibliographic information, legal status data, EPO's register, patent classification date, among others) or even page images of the published patent document (EPO, 2015). Due to that diversity, the URI used to the server's request can be represented in different manners allowing the use of this API on different patent pipeline modules (Table 7).

For patent ID searches, the used URI is a simple variation of what is shown in Figure 14.

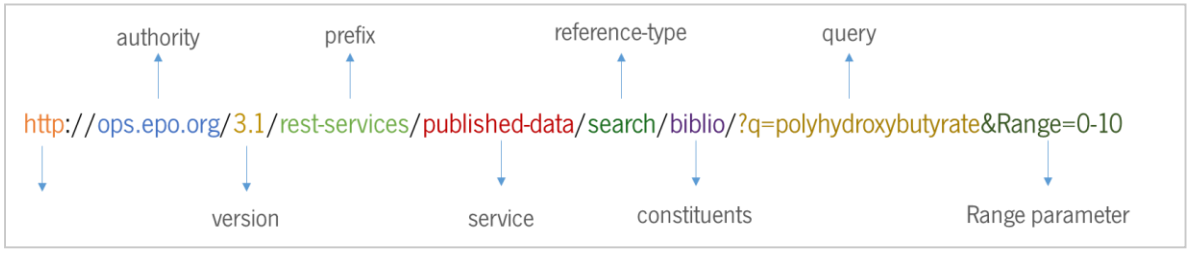


Figure 14 – Example of the used URI on *OPS web service API* to retrieve patent IDs related with PHBs.

As seen in Figure 14, the protocol, authority, version, prefix and service parts are invariable. The constituents part can be "biblio" for bibliographic data searches, "abstract" for abstract searches or "full-cycle" for getting all possible information about each result (bibliographic data and/or abstract).

Table 7 - Description of URI parts of *OPS web service* API for search and retrieval operations (EPO, 2015).

		OPS URI part	Description
Common to both operations	Invariable	protocol	Communication protocol
		authority	The host name
		version	OPS version
		prefix	The distinguishing of OPS RESTful services
		service	OPS service responsible for retrieving data
	Variable	reference-type	Type of request made to the server to define which documents associated with a given patent ID the server must return.
Search operation		constituents	Type of patent information that will be returned from the server for each search result
		query	Input definition. It can be simple (with a simple keyword) or composed by different keywords coupled with AND/OR connectors.
Retrieval operation		input-format	Represents the input patent ID number format.
		endpoint	Defines the information that must be returned.

The query is introduced after the "*?q=*" designation. To filter results, alongside with the API parameter (*Range=*), search operators can be used (Table 8).

Table 8 – Brief description of the API parameters and search operators that can be used on *OPS web services* API to search for patent IDs (EPO, 2016). The API parameters are presented in italic since that is the exact URI representation.

Parameter	Type	Description
?q=	API parameter	Defines the input expression.
Range=	API parameter	Defines the range for the returned results. The maximum range value between the given values is 100.
AND	Search Operator	Add terms to query, returning only the results that contain all terms.
OR	Search Operator	Return results that contains one of the given words.
NOT	Search Operator	Exclude results that contain the given terms.
“”	Search Operator	Returns only the results which contain the exact words within speech marks.
=	Search Operator	Allows the relation definition between two terms, returning only the patent IDs where this relation is implicit.
All	Search Operator	Defines that all terms entered within quotes after it will be found on retrieved patents.
Any	Search Operator	Defines that at least one of the terms entered within quotes after it will be found on retrieved patents.
within	Search Operator	Allows the definition of the date range (years) for patent publication. The date must be inside quotes after parameter, separated by a comma.
>=	Search Operator	Allows the definition of a lower limit to the year of patent publication.
<=	Search Operator	Allows the definition of an upper limit to the year of patent publication.

3.3. Patent metadata download process

The second step of the patent pipeline is the metadata retrieval process done by the *metainformation sources module*. On that process, a data structure containing bibliographic information about every patent ID is obtained, related to the patent title, publication date, authors, abstract and description, if available. The data structure is created within the framework of @note2 for all the patent IDs retrieved, being called “query”.

There are two different components on the *metainformation sources module*: *PATENTSCOPE web service* and *OPS web service*.

3.3.1. PATENTSCOPE web service API

The *PATENTSCOPE web service* API is a data service provided by WIPO, integrating the PATENTSCOPE search engine and PATENTSCOPE database, through a Java API using *Simple Object Access Protocol* (SOAP) protocol (WIPO, 2016). That protocol, similar to REST, is used to exchange structured information with WIPO servers in a machine-to-machine communication system (Curbera *et al.*, 2002). As a platform-independent and secure system, the SOAP protocol requires the use of XML as the source for messaging and HTTP or *Simple Mail Transfer Protocol* (SMTP) for message transmission through application layer protocols (Curbera *et al.*, 2002).

The *PATENTSCOPE web service* API can access bibliographic data for all published patent applications through the WIPO server, returning information in the XML format. Also, this web service allows the retrieval of the published patent file. There are several predefined methods available for use on this API depending on what information is required (Table 9).

Table 9 – Different *PATENTSCOPE web service* API predefined methods and respective description (Waring, 2012). The necessary parameters for each method are presented in bold.

***PATENTSCOPE web service* method**	**Description**
getAvailableDocuments	Returns a list of document IDs of all available documents on PATENTSCOPE database for a given **application number (patent ID)**.
getDocumentContent	Returns the binary content of a given **document ID**.
getDocumentOcrContent	Returns the binary content of a given **document ID** in text-based PDF format.
getDocumentTableOfContents	Returns all the page IDs for a given **document ID**.
getDocumentContentPage	Returns the binary content of a given a **page ID** of a patent document and the respective **document ID**. The document pages can be in TIFF or XML format.
getIASR	Returns the bibliographic data of a given **application number** in XML format.

To retrieve patent metadata using the *PATENTSCOPE web service*, there are two possible methodologies. The first is obtaining an XML file through the application of the method *getIASR* giving a patent ID as input. Another methodology can be used through the use of the *getAvailableDocuments* method, followed by the *getDocumentTableOfContents* method, and finally,

the *getDocumentContentPage* method applied only to the XML file returned by *getDocumentTableOfContents* method. After getting the XML file, a parser can be used to get the metadata associated with the patent (Figure 15).

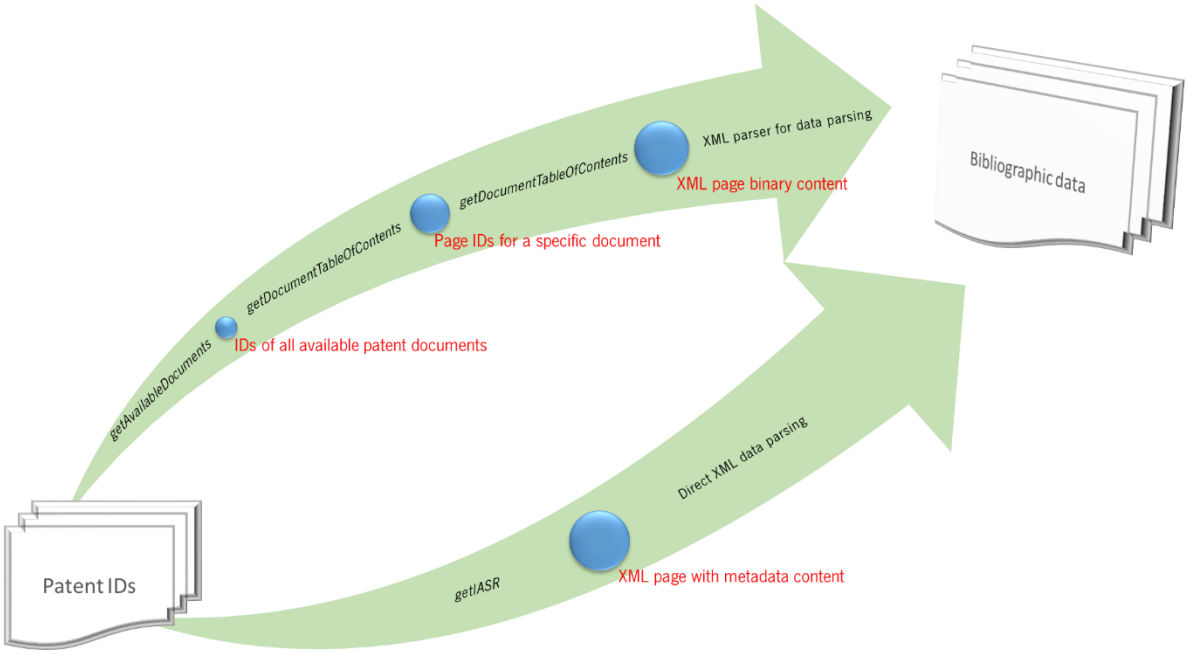


Figure 15 – Schema about the possible alternatives for *PATENTSCOPE web service* patent metadata retrieval process.

3.3.2. OPS web services API

The other component of the *metainformation sources module* is used to retrieve metadata is the *OPS web services* API. As described previously, there are many requests that can be made through this web service allowing the patent IDs searching process and the patent data retrieving (Table 7). However, a different URI is needed for every request type. In the metadata retrieval process, the used URI parts refer to patent data retrieving request type: the reference-type, the input-format and the endpoint (Table 7).

The reference-type part of the URI used on the *OPS web service* refers to the different documents that can be available for the same patent ID. This is due to the fact that obtaining a

public patent document is a slow process, and there are modifications throughout different stages of that process. It can be “publication”, for the published patent application with all bibliographic data updated at the time of publication; “application”, for the “middle-stage” documents representing the formalities filled by the applicant to obtain the patent (description and claims are included); or “priority”, for the data filled by applicant on the first filling of their patent application (EPO, 2015).

The input-format represents the format of the input application number. It can be either “epodoc” for country code, number and kind code concatenated (e.g. DE200720016308U), “docdb” for all application number parts separated with dots being the country code, number and kind code mandatory (e.g. DE.202007016308.U) or “original” for unformatted patent ID number as found on original patent document (e.g. DE.(20 2007 016,308.8)) (EPO, 2015). The *search source module* components only return patent IDs through “epodoc” format, so all OPS retrieval services use that format as input.

Finally, the endpoint part of the URI in published-data request structure defines the access to the worldwide patent data and what information should be downloaded. The endpoint has a default value and if not defined, “biblio” is automatically the chosen value. The “biblio” value allows downloading all patent metadata but some other options are available to download only parts of the bibliographic data: “abstract” to get only the patent abstract; “description” to get only patents description; “claims” to download only the patent claims; and “fulltext” to download patent claims and description, if both are available (EPO, 2015).

Taking advantage on that variability of options, the reference-type used on our patent pipeline for metadata retrieval is defined as “publication”, the input-format as “epodoc” and the endpoint as “biblio” (Figure 16).

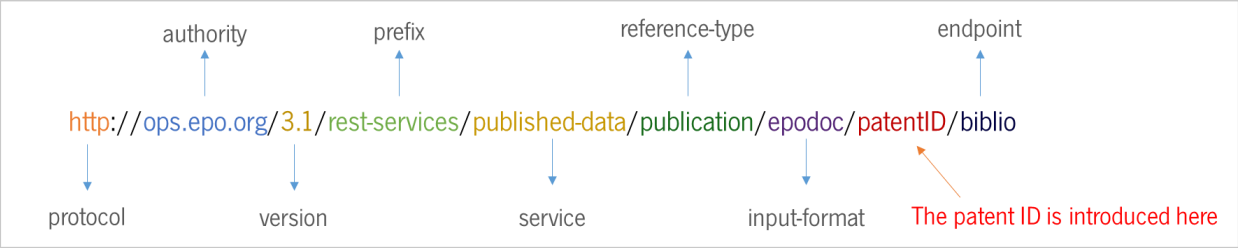


Figure 16 – Representation of the used URI to retrieve metadata on *OPS web service* API, given a patent ID on "epodoc" format.

As seen in Figure 16, after replacing the URI part, noted by "patentID" for the respective patent ID of interest, this URI can be used to communicate with the server, returning an XML or JSON file that can be easily parsed to extract available bibliographic information.

3.4. Patent PDF file retrieval process

The third step of the patent pipeline is the patent retrieval using the components of *retrieval sources module*. For each patent ID, a PDF file with all pages as seen in the published patent application is returned. There are two web services that can be used to download that PDF files: PATENTSCOPE and OPS. Similar to patent metadata retrieval, these two act together, allowing the download of a great number of patents. The patent PDF retrieval process, although being a corpus creation process component is interlinked with the @note query data structure, previously generated, since the absolute path to the PDF file is added to the query in the process. That allows correctly getting PDF files.

3.4.1. PATENTSCOPE web services

As previously described, the *PATENTSCOPE web service* has several Java methods which can be combined to request patent data. After searching for available documents using the *getAvailableDocuments* method, the PDF file can be obtained using two different methodologies. The first and the quickest way is the direct returning of the PDF file with the use of

getDocumentOcrContent method providing a patent text file with high quality (Figure 17). The other way is downloading every page separately. For that purpose, with the use of *getDocumentTableOfContents* after *getAvailableDocuments* method, a list of all page IDs for a patent document ID is provided being the patent pages downloaded using the *getDocumentContentPage* method iteratively for every page (Figure 17). That alternative method requires a posterior merge of all pages into a single PDF file, with tools as *PDFMergerUtility* class from *PDFBox*[10] API, whose version 2.0.1 was used in this work.

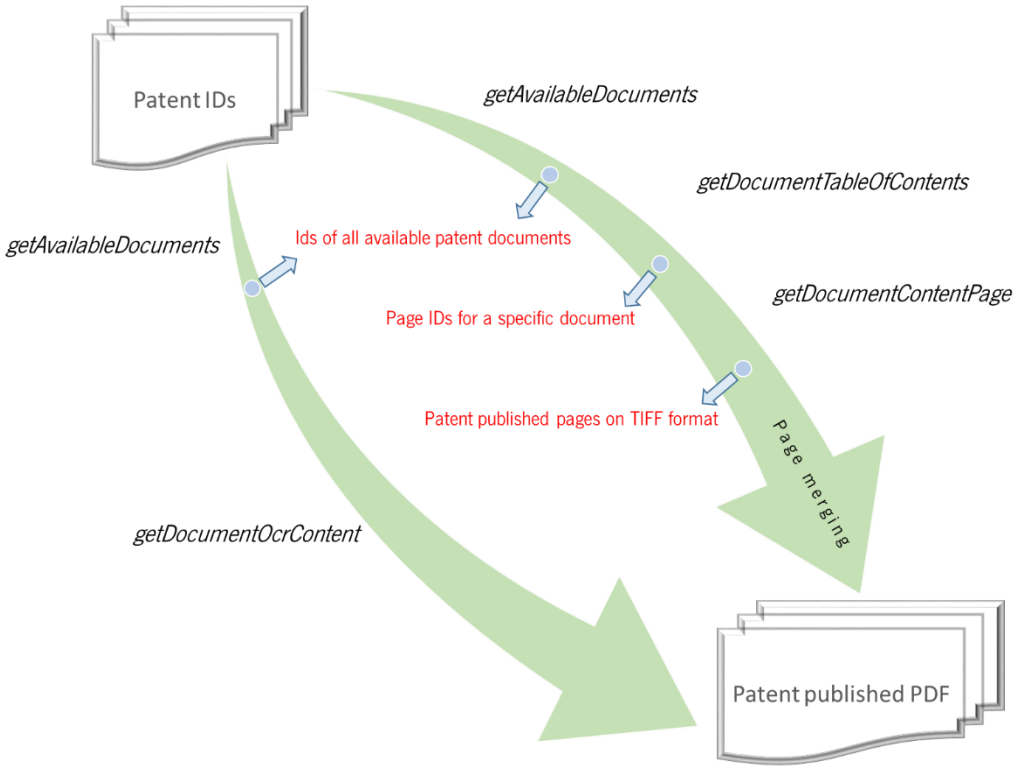


Figure 17 - Schematic view of the used *PATENTSCOPE web service* methods to patent PDF file downloading given a patent ID.

[10] (https://pdfbox.apache.org/docs/2.0.1/javadocs/org/apache/pdfbox/multipdf/PDFMergerUtility.html)

3.4.2. OPS web services

Using the *OPS web service*, to retrieve patent images from the server, the request is made using a URI similar to the used for patent metadata download. However, although reference-type and input-format parts have no variation, the endpoint part must be defined as "images" (Figure 18).

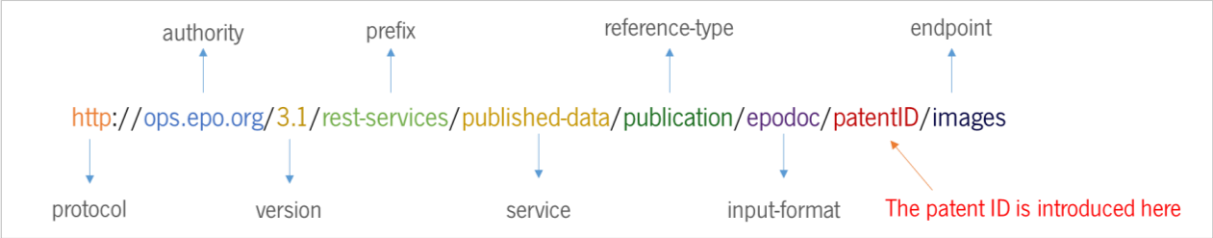


Figure 18 - Representation of the used URI for retrieve patent images on *OPS web service*.

To return a complete PDF, after replacing the URI part notated as "patentID" on the model represented on Figure 18 by the ID of the wanted patent, the request to server must be made iteratively to each page, downloading and merging them into a file using the *PDFMergerUtility* class, similarly to *PATENTSCOPE web services* API.

3.5. Text extraction from PDF files

The last step of the patent pipeline is relative to the parsing of PDF files, obtained from *retrieval sources module* components using the *PDF conversion module*. The components of this module allow the extraction of the text that is printed on the PDF pages into an editable and searchable text format. Since PDF files can have several encoding formats, different methodologies were used to achieve the main goal of this patent pipeline step (Figure 19).

The first approach tried on the *PDF conversion module* was using the version 2.0.0 of the *Apache PDFBox* library[11]. That library is an open source Java tool that allows the PDF documents manipulation. When patent PDF files have Unicode text, this tool allows the content extraction. However, the most of patent PDF files have all text printed into images, which are not detected by the *Apache PDFBox* tool requiring a more accurate method.

[11] (https://pdfbox.apache.org/index.html)

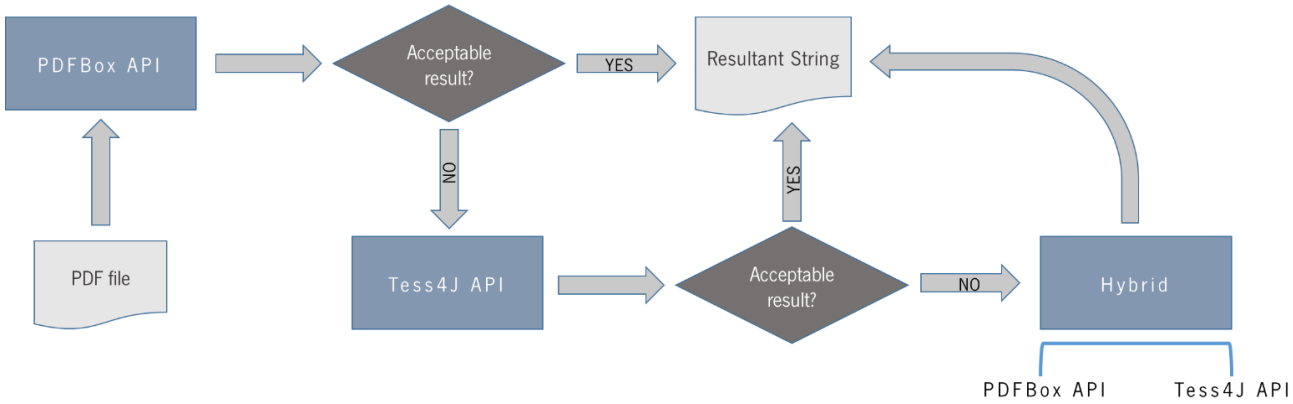


Figure 19 – Flowchart for PDF to text conversion process using *PDF conversion module* components.

Tesseract, as previously described in the previous chapter, is an OCR tool with great capabilities. Despite being coded in C++, it is the ideal tool to be used on PDF to text conversion due to platform independency and to the existence of Java wrappers (Smith, 2007). The wrapper used on this pipeline was the *Tess4J*[12] version 3.2.1 developed by Quan Nguyen and published on May 29, 2016.

The *Tesseract* system is a flexible system with many configurations that allow the algorithm behavior changing. Those parameters can be related to the loading step made by API at the resources level or even at the image processing level (*init* only parameters); to mathematical and algorithmic aspects which have an influence on algorithm application (general parameters); or even to debugging options (debug parameters), which allow the text printing or the algorithm's graphical output for the used *Tesseract* configurations or for the bugs founded (Details about *Tesseract* parameters for the three type of configurations can be accessed on Appendix II).

Using the *Tesseract* capabilities by the *Tess4J* wrapper, OCR can be successfully applied to patent PDF files. When the OCR process returns a null or an insufficient result, another approach must be used. So, using a hybrid process combining *PDFBox* and *Tess4J*, it is possible to extract all PDF pages and iteratively convert them into images that can be then processed using OCR algorithm (Figure 19).

[12] (http://tess4j.sourceforge.net/)

3.5.1. OCR evaluation through Dynamic Programming (DP)

A good way to get an OCR evaluation metric is using Dynamic Programming (DP) algorithms, which allow comparing the result of an OCR with a known result used as a gold standard (e.g. an abstract of a paper kept in a database such as PubMed).

DP is a mathematical algorithm that can be used to solve computational optimization problems commonly used in sequence analysis (Reiners, 2008). As described by Parberry and Gasarch (2002), DP uses a sequential approach, calculating the solution for all the possible sub problems and storing their values in a matrix.

Applied to computational biology, the most practical and well-known examples of this type of problem that can be solved used DP algorithms are the local or global alignments of two different DNA (or protein) sequences. There are two different algorithms that can be used for this purpose: Needleman-Wunsch, for global alignments and the Smith-Waterman for local alignments. Both algorithms represent an optimization problem with a processing time defined by $O(m * n)$ (Reiners, 2008).

The objective function which describes both algorithms is represented by the following equation:

$$S_{i,j} = max[S_{i-1,j-1} + s(a_i b_j), S_{i,j-1} + w, S_{i-1,j} + w]$$

where $S_{i,j}$ is the cell with the (i,j) coordinates on the matrix S; $s(a_i b_j)$ represents the match/mismatch score of a given cell in the matrix; and w represents the score assigned to the gap penalty. The gap represents a "space" in the alignment between the two sequences that in biological terms can be relative to a mutation by a nucleotide insertion or deletion in one of the sequences (Edgar, 2004).

To solve DP algorithms, it is important to define which values will be used for the match, mismatch and gap penalty. For the given examples, the value 1 is assigned to match, the value -1 to mismatch and the value -2 to the gap penalty (Reiners, 2008). Using that values, the array of *m+1,n+1* dimensions can be built. The *m* and *n* characters correspond to the size of the two

sequences that are going to be aligned (Figure 20). Using the Smith-Waterman algorithm, negative values are not allowed in the array, being automatically replaced by 0 (Reiners, 2008).

To obtain an alignment, after array values assignment, the main goal is searching for all possible local alignments and then selecting the one with the highest score as the best possible alignment (Reiners, 2008). The best local alignment is the one that ends with the highest scored cell which is the beginning of a process called traceback. The traceback process consists on the array iteration starting on that array cell and ending with a cell with 0 as a value representing the beginning of the alignment (Figure 20).

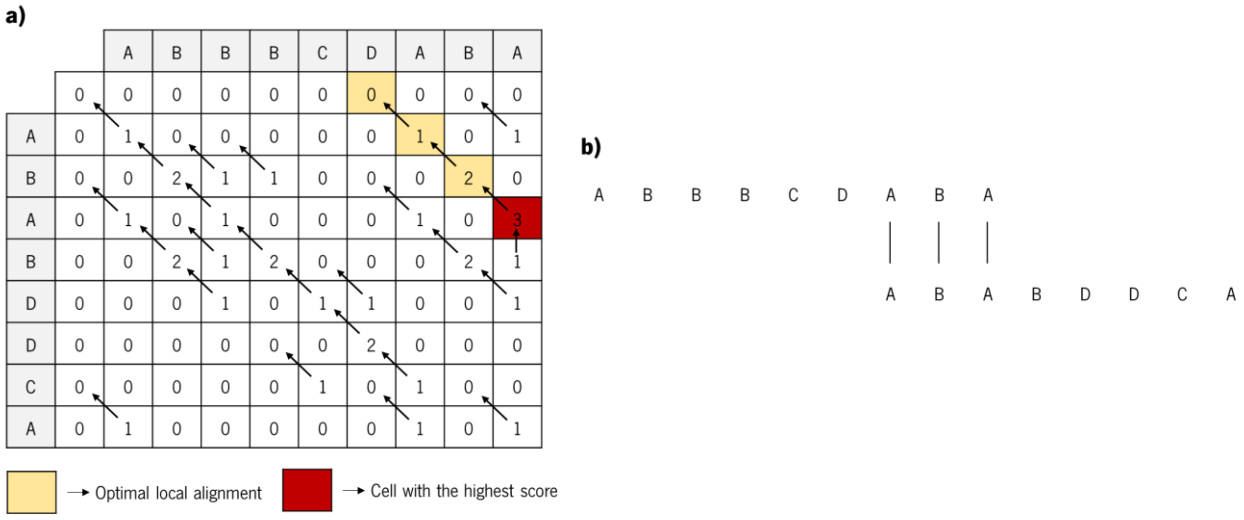


Figure 20 - Local alignment results of two sequences (ABBBCDABA and ABABDDCA). a) represents the traceback process on alignment array; b) is the alignment's schematic view.

For each iteration, starting from the cell ($S_{m,n}$) and looking then at the $S_{m-1,n-1}$ cell value, if it corresponds to a match or mismatch value transition, the traceback process continues through this cell. Otherwise, it is necessary to look at the above ($S_{m-1,n}$) and the left cell ($S_{m,n-1}$) searching for a gap penalty value transition. This iterative process is performed until reaching the alignments initial cell.

As can be seen in Figure 20, the best local alignment of the sequences ABBBCDABA and ABABDDCA has a total score of 3 corresponding to the sum of 3 matches (3 times the value 1).

In the given example, the comparison is made for the individual letters of two different sequences but it can be applied to numbers, words or something that can be compared iteratively. Applying this process to OCR texts, each matrix row and column can be represented by tokens that constitute the texts from OCR and a curated corpus, respectively. After building the array with these tokens, DP algorithms can be applied, evaluating the OCR performance thought the number of tokens that match exactly on the two texts (Figure 21).

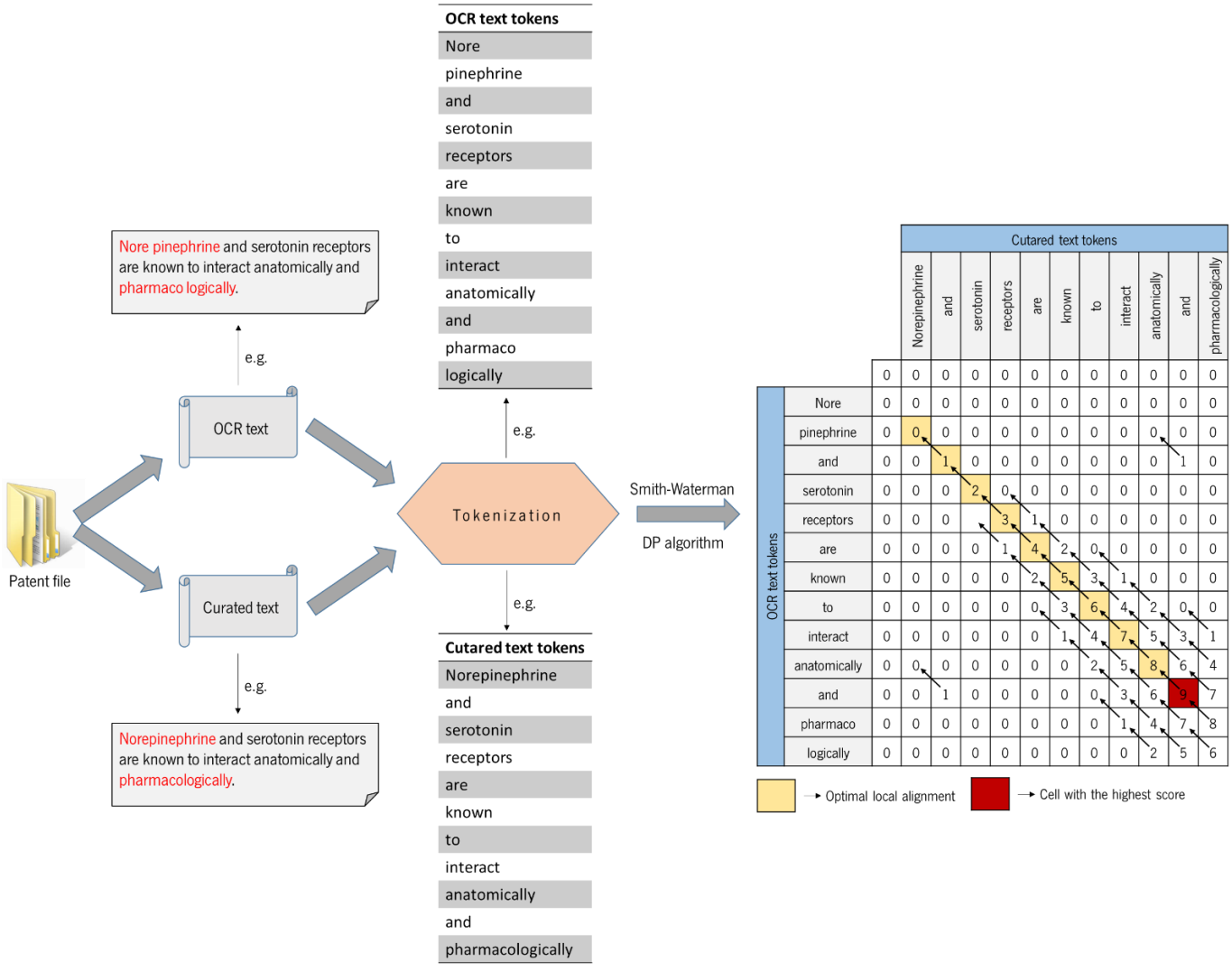


Figure 21 – Smith-Waterman algorithm applied to OCR texts and the correspondent curated text. The example was taken from US7417064 patent text.

As it can be seen in Figure 21, the OCR evaluation verifies the exact match between each token. In the given example, two fragmented words were introduced on the alignment which leads to an alignment with 9 as score representing nine matches of the eleven possible ones.

3.6. Java Implementation

The patent pipeline is divided into four different modules representing the developed IR Search process (represented by the *search sources module* and the *metainformation sources module*), the IR Crawling process (represented by the *retrieval sources module*) and PDF to text conversion, represented by the *PDF conversion module*.

The IR Search process and the IR Crawling process modules have specific interfaces that combined with abstract classes, turns mandatory the implementation of several methods for all the components (Figure 22). That approach allows the correct implementation of all the used components into their respective modules and using that architecture, new components are able to be added posteriorly into the designed system.

The components used in the *PDF conversion module* use specific configurations without any interface representing it. That approach was made since that module is viewed as a multi-option module that is able to convert PDF files with several encryption modes. So, although new conversion modules can be added, the designed architecture was not built to that as in the other modules.

At the implementation level, the *search sources module* encompasses a patent ID retrieval configurator (*IRPatentIDRecoverConfigurationImpl*) that allows the configuration of patent ID retrieval step through a keywords definition to the posterior search process; an abstract class (*AIRPatentIDRecoverSource*) that turns the patent ID retrieval configuration and the respective validation mandatory to the use of any component of this module; a class (*WrongIRPatentIDRecoverConfigurationException*) that handles all exceptions that can be returned from the *search sources module* simplifying eventual error cases resulting from input errors from the user; and, three packages representing the used components. Linked to the *search sources module* are the *Custom Search* API, the *Bing Search* API, and the *OPS web services* API packages. All components of this module must have a similar implementation with a configurator that extends to the patent ID retrieval configurator, allowing the introduction of the component-specific access

credentials, alongside with the input keywords and with a data access class extending the abstract class of the *search sources module* and taking a patent ID retrieval configuration (*IIRPatentIDRecoverConfiguration*) as input. That class has the methods responsible for searching and returning a set of the patent IDs that satisfy the introduced keywords.

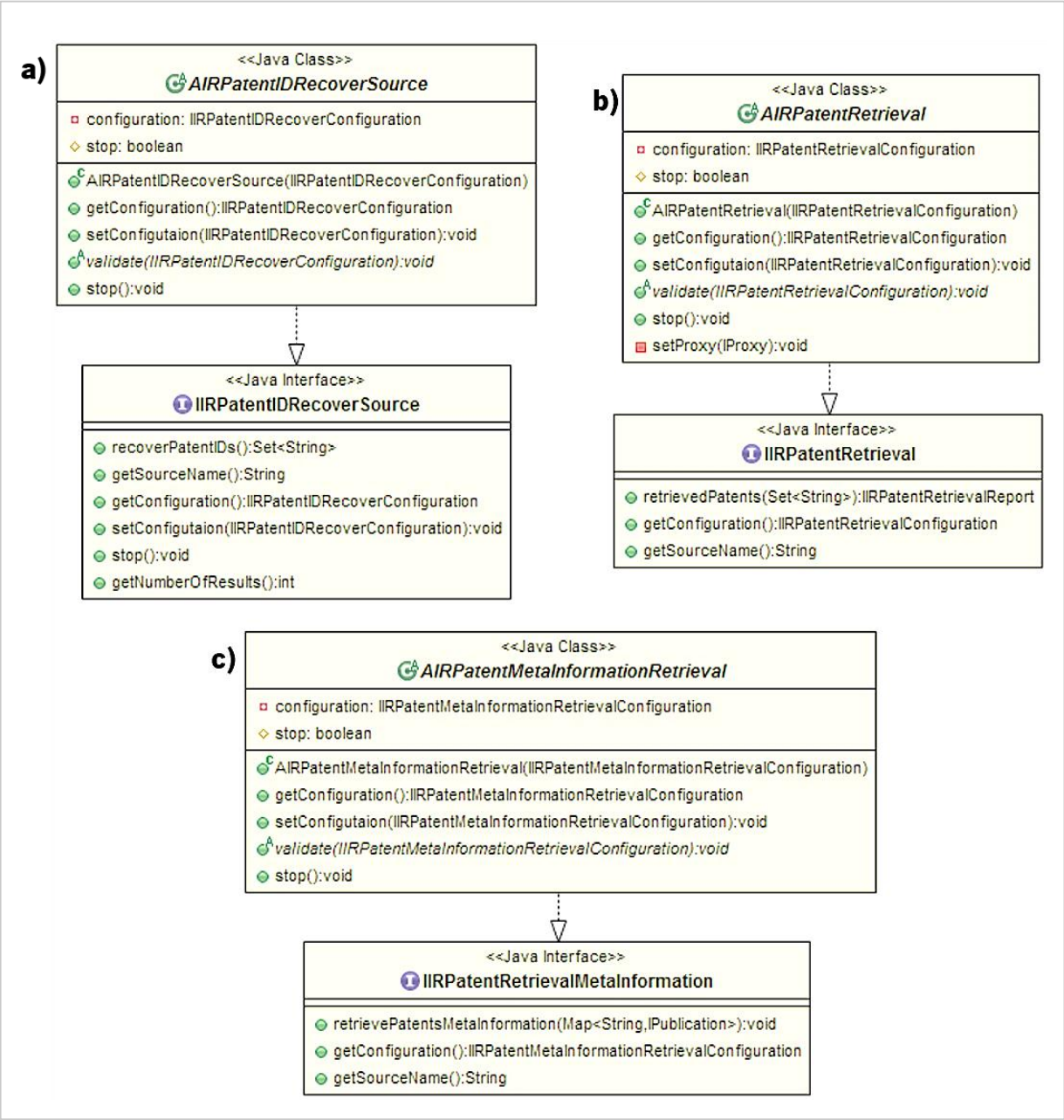


Figure 22 – The implemented interfaces and the respective abstract classes used to implement the components of each module. **a)** represents the architecture of the *search sources module*; **b)** represents the architecture of the *retrieval sources module*; and **c)** represents the architecture of the *metainformation sources module*.

The *Custom Search* API encompasses also a package (*googleEntities*) beyond the configurator (*IRPatentIDRecoverGoogleSearchConfigurationImpl*) and the data access class (*GoogleSearchPatentIDRecoverSource*). That package contains the API data structures, as the class (*GoogleResults*) responsible for saving all results information obtained from the server after the search process.

Similar to the *Custom Search* API, the *Bing Search* API encompasses a package (*bingEntities*) with API data structures. The other components are the expected ones for all *search sources module* components: a configurator (*IRPatentIDRecoverBingSearchConfigurationImpl*), and a data access class (*BingSearchPatentIDRecoverSource*).

The *OPS web services* API package alongside with a configurator class (*IRPatentIDRecoverEPOSearchConfigurationImpl*) and a data access class (*EPOSearchPatentIDRecoverSource*) encompasses a class (*OPSUtils*) with static methods that allow many different requests to EPO's server to get different information as patent IDs, and a package (*opshandler*) that allows the XML parsing for every possible result type.

The *metainformation sources module* encompasses several structures that define the module and build a pattern that must be followed by any component. A patent metadata retrieval configurator (*IRPatentMetaInformationRetrievalConfigurationImpl*) allows the proxy definition for each component which, when defined, allows the existence of an intermediary server for client requests seeking resources from other servers and makes easier the information exchanges. A patent metadata report implementer (*IRPatentMetaInformationRetrievalReportImpl*) allows the creation or the temporary storing of a hash map with the patent IDs and the respective scientific document data implementation (*IPublication*), a data structure from @note2 that is then used to store the obtained metadata. An abstract class (*AIRPatentMetaInformationRetrieval*), similar to the abstract class of *search sources module*, turns the use of a patent metadata retrieval configuration, and the respective validation, mandatory to any *metainformation sources module* component.

To handle any error resulting from the input parameters defined, a class (*WrongIRPatentMetaInformationRetrievalConfigurationException*) was also implemented.

Along with all described classes, there are also two different packages on *metainformation sources module,* representing the two components used: *PATENTSCOPE web services* API and *OPS web services* API.

All components of *metainformation sources module* must encompass a configurator class that extends the patent metadata retrieval configurator, allowing the input of component's access credentials, alongside with the proxy and a data access class that takes a patent meta data retrieval configuration (*IIRPatentMetaInformationRetrievalConfiguration*) as input. After patent meta data retrieval configuration validation, that data access class retrieves all available metadata for each patent ID. All retrieved data are inserted into a @note data structure (*IPublication*) of a given hash map earlier generated.

Similar to the *search sources module*, each component can have other implemented structures specially designed for that system. The *PATENTSCOPE web services* API package, besides the configurator class (*IRWIPOPatentMetaInformationRetrievalConfigurationImpl*) and the data access class ((*WIPOPatentMetaInformationRetrieval*), encompasses a package (*wipoEntities*) that contains the classes used by this API to client authentication and to request information to WIPO servers alongside with a handler (*WIPOXMLSAXPHandler*) created to parse the XML files obtained from server, extracting the available metadata.

The same happens with the *OPS web services* API package, which encompasses a configurator class (*IROPSPatentMetaInformationRetrievalConfigurationImpl*), a data access class (*OPSPatentMetaInformationRetrieval*), but also a class (*OPSUtils*) with static methods responsible for getting authorization from the server and returning metadata from the EPO server for a given patent ID, and a package (*opshandler*) that contains handlers for the parsing of XML files obtained from the EPO server.

The *retrieval sources module*, similarly to the *metainformation sources module,* is linked to the same two different components represented by two different packages inside the module: the *PATENTSCOPE web services* API and *OPS web services* API. Alongside with component packages, there are also: a patent retrieval process configurator (*IRPatentRetrievalConfigurationImpl*), that allows the proxy and the output directory definition; a patent retrieval process report (*IRPatentRetrievalReport*), that allows the existence of a report to give information about which patent

IDs were retrieved and those that were not, allowing a general assessment of the success of the retrieval process and preventing that an already downloaded patent can be processed two times; an abstract class (*AIRPatentRetrieval*) that turns the patent retrieval process configuration mandatory; and a class (*WrongIRPatentRetrievalConfigurationException*) that handles input errors evidenced on patent retrieval configuration.

All *retrieval sources module* instances must encompass a configurator class that extends the patent retrieval process configurator and, similarly to other module configurator classes, allows the input of components access credentials alongside with proxy and the output directory.

The *PATENTSCOPE web services* API package encompasses also the same package (*wipoEntities*) that were used to retrieve patents metadata on *metainformation sources module* to grant the access to WIPO server, retrieving data along with the configurator class (*IIRWIPOPatentRetrievalConfigurationImpl*), and the data access class (*WIPOPatentRetrieval*).

The *OPS web services* package encompasses a class (*OPSUtils*) with the static methods available to the authentication process and to deal with the data requested and returned from the server, and a package (opshandler) with XML parsers to deal with the returned data, allowing the downloading of the patent pages. As expected, the *OPS web services* package also contains the configurator class (*IROPSPatentRetrievalConfigurationImpl*) and the data access class (*OPSPatentRetrieval*).

The last step of the patent pipeline is relative to the PDF to text framework. The PDF to text package encompasses the main class (*PDFtoText*) with a public method that allows the PDF to text conversion, alongside with private methods that are used to apply the different conversion techniques separately or to validate the conversion helping to realize if the use of another conversion technique is required.

We also implemented a control class (*TesseractManager*) that turns the *Tesseract* instance into a singleton, avoiding time losses and memory overload opening the trained models for every PDF that is converted using OCR methodologies. The control class is also used to manage the *Tesseract* parameters that were previously described in section 3.5 allowing the results improvement if necessary. Alongside with these classes, a package (*tessdata*) is necessary, containing all

dictionaries and models previously trained, for the required languages. In this project, only English models were considered.

The class diagram of the patent pipeline, with all the implemented interfaces that were used to build the designed architecture, is represented in Figure 23.

To engage the entire pipeline, a class (*PatentPipeline*) was created, which allows adding different module components configurations that are then used to run the entire pipeline, using a single Java method. Some variations can be made using that class since each patent pipeline module can be run separately.

The patent pipeline has a great interconnection with the @note2 structure, sharing some data structures that allow an approach to other systems already implemented. Among these data structures is the final content of each pipeline sections: a query (*IQuery*) or a corpus (*ICorpus*). That interconnection allowed the insertion of the designed IR Crawling process and the IR Search process to PMM and a pre-processing step on CM, as described at the beginning of this chapter (section 3.1).

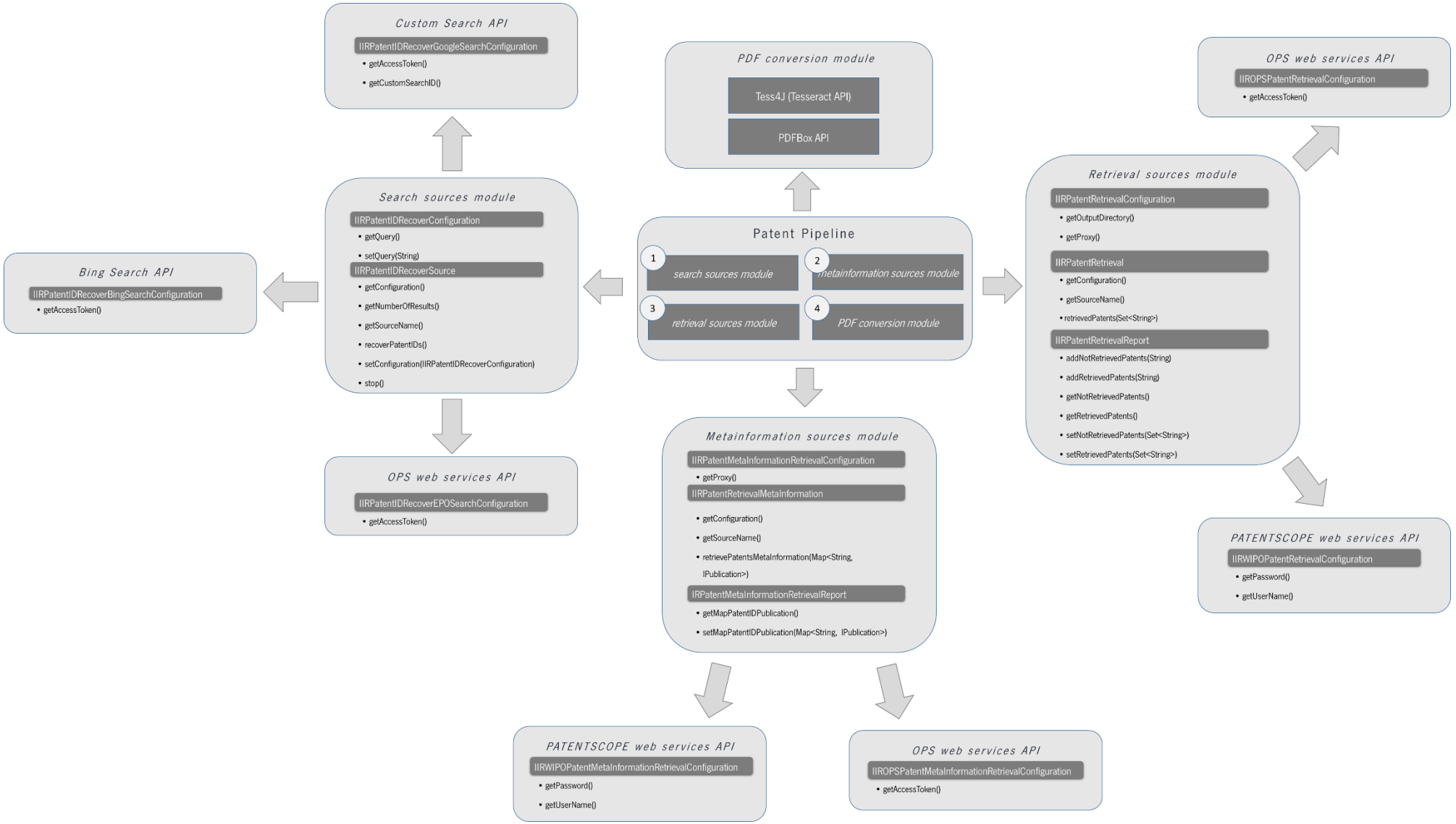


Figure 23 – Class Diagram for the designed patent pipeline. The numbers represent the patent pipeline flow.

4. CASE STUDY

4.1. BioCreative V CHEMDNER task

BioCreative is a text mining event/community oriented to the implementation of IE systems on texts from the biological domain. The BioCreative V event, in particular, was designed to the development of systems to solve several BioTM problems. Being composed of several tasks, the BioCreative V CHEMDNER was the one that was directed to the implementation of automatic IE processes in patents related to chemicals.

For the detection of mentions to chemical entities within patents (CEMP task), the BioCreative organizing committee provides three different corpora with manual annotations by domain experts: a training, a development and a test set of patent documents. The training and development sets are used to build the systems that are then validated using the test set. That allows the evaluation of every participating team by their system prediction capabilities.

All the used corpora are constituted by 7000 patent IDs, titles and respective abstracts of patent documents related to medicinal chemistry applications. However, to avoid manual curation, the test set is distributed with 40000 patents, from which 33000 were added as background noise.

All patents from the used datasets were published between 2005 and 2014, in English, obtained from several patent databases as WIPO, EPO or USPTO. Those patents have assigned at least one of the following IPC codes: A61P and one A61K31, being the A61P IPC code related to patents with the specific therapeutic activity of chemical compounds or medical preparations and the A61K31 IPC code to patents related to medicinal preparations containing organic active ingredients.

To get a comprehensive number of patents to be used as a study case, these three patent sets were grouped into a single set for which a loader was developed (54000 patents). This loader was implemented with a text length evaluator for the abstract section, allowing the creation of a map with the 1000 longest abstracts and the respective patent IDs. On Table 10, the 10 patents with the longest abstract section are represented for illustration purposes, alongside with the title and the total length, in single characters.

Table 10 – The patent Ids and respective title for the patents with the biggest abstract size of all BioCreative V CHEMDNER task datasets.

Patent ID	Title	Abstract length
CA2633585A1	Treatments for any and all mental/brain neuron/spinal cord illness	12390
US20110124626	Benzazepine derivatives and their use as histamine h3 antagonists	4072
US20110123468	Use of benzotropolone derivatives as UV absorbers and antioxidants and their use in sunscreens and/or cosmetic compositions	4298
US20050033051	Nucleosides preparation thereof and use as inhibitors of RNA viral polymerases	2837
US20050038069	Substituted azabicyclic compounds	2534
US20100233083	Microparticles comprising a crosslinked polymer	2508
US20100137405	RNA Interference Mediated Inhibition of Cyclic Nucleotide Type 4 Phosphodiesterase (PDE4B) Gene Expression Using Short Interfering Nucleic Acid (siNA)	2451
CA2662538A1	Azabicyclic compounds as inhibitors of monoamines reuptake	2450
US20080167245	Novel metastasis suppressor gene on human chromosome 8	2446
US20100124537	Medical applications of alpha-ketoglutarate	2431

4.2. Patent pipeline validation process

The map obtained as described in the previous section was used as input to the designed patent pipeline described in the previous chapter. The patent IDs are processed by the developed IR process, creating the query data structure and extracting all the available metadata for each patent. This procedure eliminates the patent ID retrieval process on the patent pipeline flow. Indeed, the validation of the patent ID search process is not viable, since it uses several search engines with results that change every day by the emergence of new publications. So, instead of using the patent ID search task, it was replaced for the case study patent IDs. This approach allowed the IR Search process validation, surpassing the keywords introduction, resorting to patent ID pre-selection. The

advantage of this approach is that all the components used to build the IR Search process (relative to the metadata retrieval process), the IR Crawling process or the PDF to text conversion system can be validated since a well-known number of patents is used, working as a gold standard to benchmark our system.

Using the two components that constitute the *metainformation sources module* on this project (the *PATENTSCOPE web services* API and the *OPS web services* API), complete patent meta-information was extracted for 917 patents from the 1000 that were previously given, corresponding to a success rate of 91,7%. That information is relative to all the publication fields that can be filled with patent data on the @note2 *IPublication* data structure: the invention title, the abstract section (that for some cases is merged with the description and claims section), authors, publication date and external link.

From the remaining 83 patents, 76 were filled with partial data that is relative to invention title, authors, publication, and external link. For some patents, these fields are all filled but to others, only some of this information is obtained. The others 7 patents do not return any information from any component server. This can be related to the use of free access credentials to the data obtaining from the servers since not all the requested information is available through the use of web services without paying; or, probably, they are not registered on the used databases or they are assigned to other patent ID.

The use of free credentials is a great disadvantage since the obtained results are dependent on the used components. To improve that result, decreasing the number of limitations on the amount of retrieved data, the best solution besides using paid credentials is the implementation of new components on *metainformation sources module.*

After all available metadata is retrieved, a @note2 query data structure is built using that information. A brief preview of the obtained query for our study case is represented in Figure 24.

After that metadata retrieval process, the patent PDF files were downloaded for the respective patent IDs. Using the *PATENTSCOPE web services* API and *the OPS web services* API, the two components of the *retrieval sources module*, 993 patents were downloaded, which corresponds to a success rate of 99,3%. The success rate was not 100% due to the inexistence of some patents

on the database but, as before, can be also related to restrictions imposed by the use of free credentials to access server information.

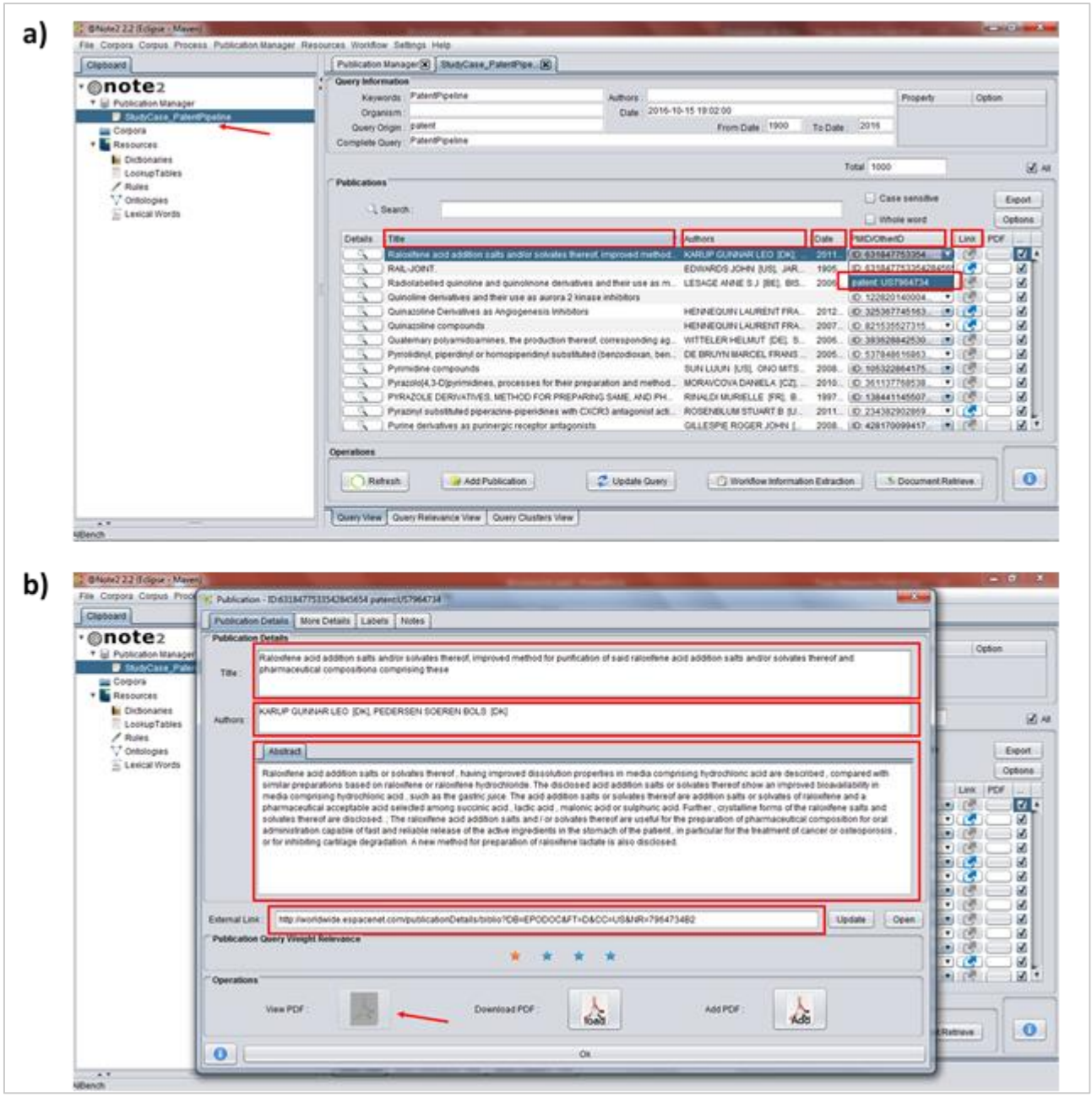


Figure 24 – The @note2 query data structure obtained from the patent pipeline study case using the implemented IR Search process. All metadata extracted is evidenced through red boxes. a) represents a general view of the query data structure; b) represents the detailed description of a single patent from the obtained query data structure.

The results for the developed IR Search process relative to the metadata retrieval system and the IR Crawling process are shown in Table 11.

Table 11 – Number of processed patents for the metadata and the patent retrieval process, using the IR Search and the IR Crawling modules, respectively.

		IR Search (metadata retrieval)	IR Crawling
Evaluation parameters	Number of processed patents	1000	1000
	Number of patents (Total)	917 (or 993, with partial data included)	993
	Success rate (%)	91,7%	99,3%

The query data structure, after the PDF retrieval process, is updated with the obtained PDF file for the corresponding patent, as previously seen in Figure 9 (section 3.1). The resulting @note2 query can be seen in Figure 25, with the PDF file section evidenced.

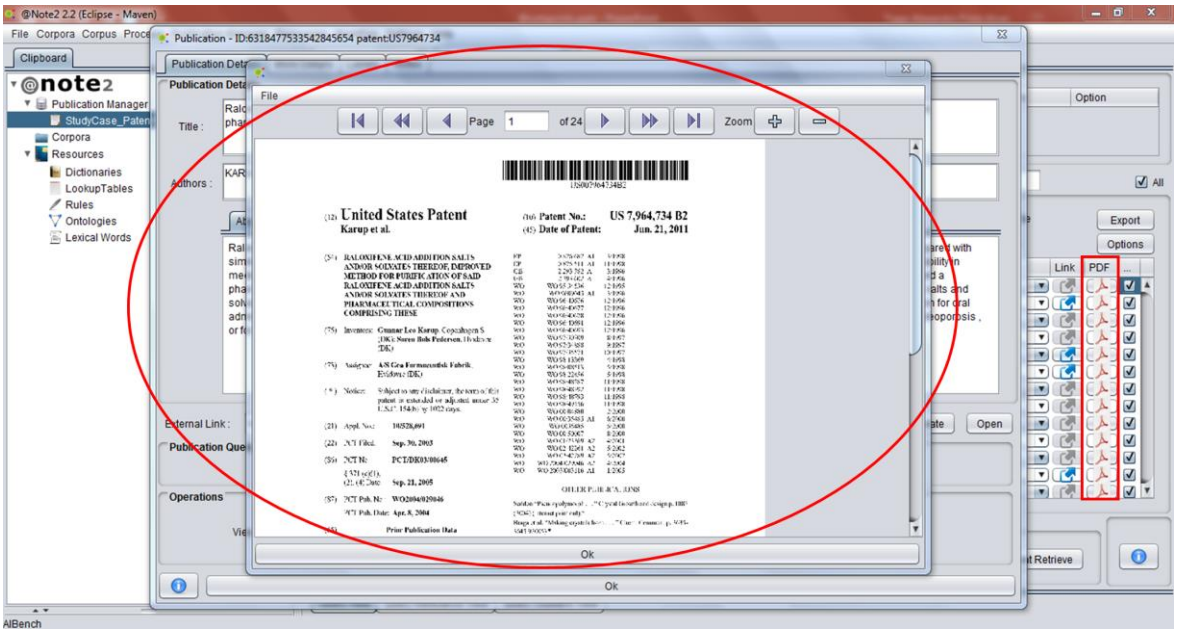


Figura 25 - The @note2 query data structure obtained from the patent pipeline study case implementing the IR Search process, including the patent pipeline PDF retrieval task.

Regarding the fourth patent pipeline task (PDF to text conversion process), the *PDF to text conversion module* is constituted by three different methods, and the entire process can be seen as multiple sequential sub-processes, where if one approach is not capable of returning good results, the next one is used.

So, since the entire patent PDF files are constituted by patent images aggregated into a single file, the *Tess4J* method is applied. Since all PDF files have the same structure, this approach is used to convert all the files and the *PDFBox* or the hybrid method are never used in this case study. Taking advantage of this feature, since *Tesseract* has several parameters that can change the algorithm behavior, several alternative values for these parameters have been tried to find the better PDF to text conversion (Table 12).

The 1st approach uses the default values for *Tesseract* algorithm. It was used to start the evaluation of the PDF to text conversion and to be the basis to compare against any modification of *Tesseract* parameters.

The 2nd approach is based on two different modifications: one on the *init* only and the other on the general parameters. The *init* only parameter changed refers to the given resolution to the patent PDF pages on the loading process (*image_default_resolution* parameter) which was changed to 400 DPI. The general parameter changed was the "*tessedit_pageseg_mode*". Previously defined as "0", this parameter represents the page segmentation mode. The default value refers to the assumption that the text is represented by a single uniform block of text. On patent PDF files, in many situations, the text is printed in two different columns. So, this value must be defined as "1", representing an automatic page segmentation process with the detection of text blocks, orientation, fonts and language scripts, ignoring other languages and characters besides the used on English (e.g. Chinese symbols).

The 3rd approach uses the same two modifications made on the 2nd approach alongside with two new modifications on *Tesseract* general parameters. The *heuristic_segcost_rating_base* parameter, related to the segmentation cost, when bigger, allows the exclusion of some unrecognized characters from results and the inclusion of entire words that were broken with a lower value. In this approach, it was defined as 1.5. The *textord_min_linesize* parameter was changed to 2.5,

representing the minimum line size. Since it was increased, the text from very tiny lines that is frequently converted into background noise on the conversion process was excluded.

Table 12 – The used values for some *Tesseract* parameters using three different approaches.

***Tesseract* parameters**	**Used values**		
	1st approach	2nd approach	3rd approach
image_default_resolution	300	400	400
editor_image_blob_bb_color	4	4	4
editor_image_xpos	590	590	590
tessedit_image_border	2	2	2
load_system_dawg	1	1	1
load_punc_dawg	1	1	1
tessedit_pageseg_mode	0	1	1
matcher_good_threshold	2	2	1,5
heuristic_segcost_rating_base	1,25	1,25	1,5
language_model_penalty_punc	0,2	0,2	0,2
textord_linespace_iqrlimit	0,2	0,2	0,2
textord_min_linesize	1,25	1,25	2,5
language_model_ngram_space_delimited_language	1	1	1

Applying the latter approach to PDF files with good classification metrics, obtained without those changes, can be an error, since some correctly extracted text will be ignored. To surpass that particularity, the 3rd approach was applied only to the documents where the 2nd approach results had a recall value under 80%. The recall was the chosen metric to decide the inclusion or the exclusion of the 2nd approach results, since most of the times, although the local alignment being smaller due to

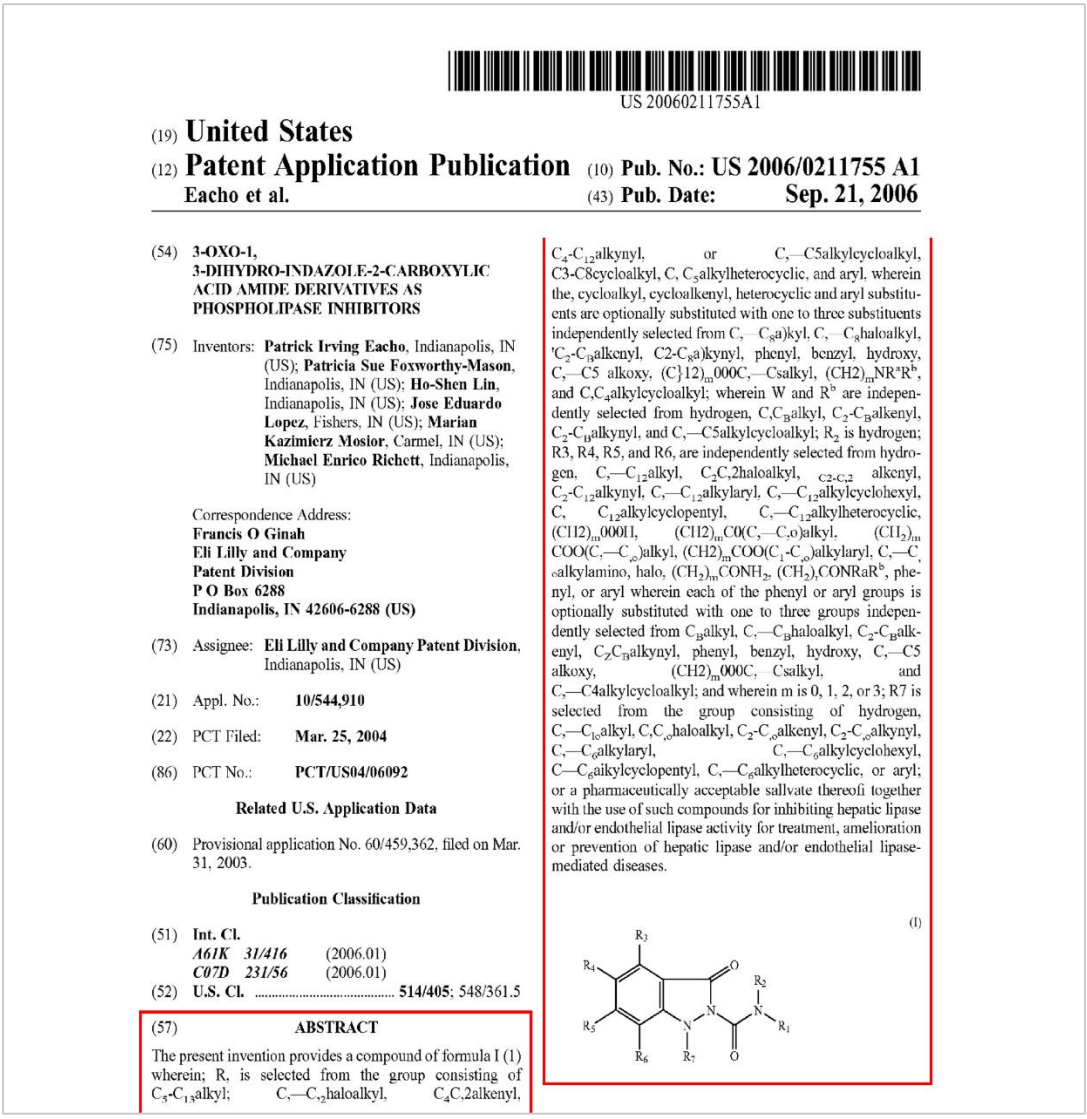
background noise removal, it represents high precision values and extremely low recall values. That is caused by the small fraction of the total abstract that is present on the alignment.

To get the evaluation metrics and validate the task, a Dynamic Programming algorithm was used for the comparison of the tokens in the obtained PDF text with the known abstract, available on the case study patent dataset, properly tokenized.

It is noteworthy that all full-texts and abstracts are pre-processed before being tokenized in a process where spaces are added to words that are linked with commas (“,”), semicolon (“;”), plus sign (“+”), asterisk (“*”), hyphen (“-“), parenthesis (“(“ or “)”) and other similar symbols. This allows the correct detection of chemical names/structures, special words or simply to separate words from punctuation symbols. The common word breaks, tabulations, excessive line breakers, as well as invalid characters are also removed on this pre-processing. That normalization process allows surpassing several PDF to text conversion errors, allowing a correct comparison between the two texts.

Taking into account that the abstract section covers only a short section of the entire patent full text, to apply the DP algorithm, the best approach that can be made is by a local alignment of the two sequences of tokens, similar to the example presented in section 3.5. It is assumed that errors in the abstract alignment can be extrapolated to the whole patent text.

Most of the times, the abstract section is printed into two different columns and uses several chemical formulas. That makes possible testing if the different columns were detected and if chemical formulas were correctly converted. As an example, in Figure 26 the first page of a patent (US20060211755) is represented, where the abstract section is printed in two different columns.



US 20060211755A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2006/0211755 A1**

Eacho et al. (43) **Pub. Date: Sep. 21, 2006**

(54) **3-OXO-1, 3-DIHYDRO-INDAZOLE-2-CARBOXYLIC ACID AMIDE DERIVATIVES AS PHOSPHOLIPASE INHIBITORS**

(75) Inventors: **Patrick Irving Eacho**, Indianapolis, IN (US); **Patricia Sue Foxworthy-Mason**, Indianapolis, IN (US); **Ho-Shen Lin**, Indianapolis, IN (US); **Jose Eduardo Lopez**, Fishers, IN (US); **Marian Kazimierz Mosior**, Carmel, IN (US); **Michael Enrico Richett**, Indianapolis, IN (US)

Correspondence Address:
Francis O Ginah
Eli Lilly and Company
Patent Division
P O Box 6288
Indianapolis, IN 42606-6288 (US)

(73) Assignee: **Eli Lilly and Company Patent Division**, Indianapolis, IN (US)

(21) Appl. No.: **10/544,910**

(22) PCT Filed: **Mar. 25, 2004**

(86) PCT No.: **PCT/US04/06092**

Related U.S. Application Data

(60) Provisional application No. 60/459,362, filed on Mar. 31, 2003.

Publication Classification

(51) **Int. Cl.**
A61K 31/416 (2006.01)
C07D 231/56 (2006.01)

(52) **U.S. Cl.** **514/405**; 548/361.5

(57) **ABSTRACT**

The present invention provides a compound of formula I (1) wherein; R, is selected from the group consisting of C_5-C_{13}alkyl; C,—C,,haloalkyl, C_4C,2alkenyl, C_4-C_{12}alkynyl, or C,—C5alkylcycloalkyl, C3-C8cycloalkyl, C, C_5alkylheterocyclic, and aryl, wherein the, cycloalkyl, cycloalkenyl, heterocyclic and aryl substituents are optionally substituted with one to three substituents independently selected from C, —C_8a)kyl, C, —C_8haloalkyl, 'C$_2$-C$_B$alkenyl, C2-C$_8$a)kynyl, phenyl, benzyl, hydroxy, C,—C5 alkoxy, (C}12)$_m$000C,—Csalkyl, (CH2)$_m$NR^{a}R^b, and C,C$_4$alkylcycloalkyl; wherein W and R^b are independently selected from hydrogen, C,C$_B$alkyl, C$_2$-C$_B$alkenyl, C$_2$-C$_B$alkynyl, and C,—C5alkylcycloalkyl; R$_2$ is hydrogen; R3, R4, R5, and R6, are independently selected from hydrogen, C,—C$_{12}$alkyl, C$_2$C,2haloalkyl, C2-C,2 alkenyl, C$_2$-C$_{12}$alkynyl, C,—C$_{12}$alkylaryl, C,—C$_{12}$alkylcyclohexyl, C, C$_{12}$alkylcyclopentyl, C,—C$_{12}$alkylheterocyclic, (CI12)$_m$000I1, (CI12)$_m$C0(C,—C,o)alkyl, (CI1$_2$)$_m$ COO(C,—C$_{,o}$)alkyl, (CH2)$_m$COO(C$_1$-C$_{,o}$)alkylaryl, C,—C$_{,o}$alkylamino, halo, (CH$_2$)$_m$CONH$_2$, (CH$_2$)$_s$CONRaR^b, phenyl, or aryl wherein each of the phenyl or aryl groups is optionally substituted with one to three groups independently selected from C$_B$alkyl, C.—C$_B$haloalkyl, C$_2$-C$_B$alkenyl, C$_Z$C$_B$alkynyl, phenyl, benzyl, hydroxy, C,—C5 alkoxy, (CH2)$_m$000C, Csalkyl, and C,—C4alkylcycloalkyl; and wherein m is 0, 1, 2, or 3; R7 is selected from the group consisting of hydrogen, C,—C$_{Io}$alkyl, C,C$_{,o}$haloalkyl, C$_2$-C$_{,o}$alkenyl, C$_2$-C$_{,o}$alkynyl, C,—C$_6$alkylaryl, C,—C$_6$alkylcyclohexyl, C—C$_6$aikylcyclopentyl, C,—C$_6$alkylheterocyclic, or aryl; or a pharmaceutically acceptable sallvate thereofi together with the use of such compounds for inhibiting hepatic lipase and/or endothelial lipase activity for treatment, amelioration or prevention of hepatic lipase and/or endothelial lipase-mediated diseases.

(I)

Figure 26 – First page of the patent US20060211755. The abstract section is evidenced by the red box.

So, analyzing the conversion of this full-text section, the obtained values for the precision, recall and F1 measure can be generalized to the entire patent. The used DP values for the match, mismatch and gap penalty were the same as used by Reiners (2008) on his work. The results for the three made approaches are presented in Table 13.

Table 13 – Precision, recall and F1 values for the three PDF to text conversion approaches applied to the 993 patents downloaded using the components of the *retrieval sources module*.

Approach	Precision	Recall	F1
1st	93,07%	21,09%	34,45%
2nd	94,72%	72,85%	82,35%
3rd	94,95%	78,42%	85,90%

The 1st approach, with 93,07% of precision, 21,09% of recall and 34,45% for the F1 score, is the worst approach, as can be seen in Table 13. It was expected since only the default *Tesseract* parameters were used, which allows the conversion of more simplistic PDF files in less time but not the conversion of complex patent PDF structure. Although the precision value being high, the recall value shows that the obtained alignments are very short comparing to the respective entire abstract sections. This can be related with the text columns identification being the text recognized line by line as if it were only a single block of text.

The best PDF to Text conversion was made using the 3rd approach with a precision of 94,95%, a recall of 78,42% and an F1 of 85,90%. Although being less discrepant than any approach with the 1st one, the differences between the evaluation metrics for the 3rd and the 2nd approaches are evident. The precision value is close for the two approaches (94,72% to the 2nd approach and 94,95% to de 3rd one) but the recall value increased from 72,93% to 78,53%. That increase is due to the fact that only the worst results from the 2nd approach were reconverted with the 3rd approach.

Even using the 3rd approach, the recall value was not bigger than 78,53% since there are several patents with chemical structures printed on the abstract that are converted with PDF to text methodologies to background noise but in the curated texts are omitted by the use of several spaces. Alongside that, there are many chemical formulas and chemical designations on the patent abstracts with special characters that sometimes lead to misunderstandings resulting on wrong character identification.

Even though the best approach being the 3rd one, it cannot be applied to all patents since for most of them, the recall value decreased a lot comparing with the use of the 2nd approach. So, applied to real scenarios where curated patent sections may not be available, the best *Tesseract*

configuration that can be applied is using the 2nd approach parameters. However, since the abstract section can be obtained with the metadata retrieval process to almost all the processed patents, the 3rd approach, combined with DP evaluation metrics, still is a possible solution to improve PDF to text conversion methodologies and achieve as well as possible, the better identification and consequent extraction of patent texts.

The @note2 corpus created after the PDF to text conversion using the 3rd approach can be seen in Figure 27.

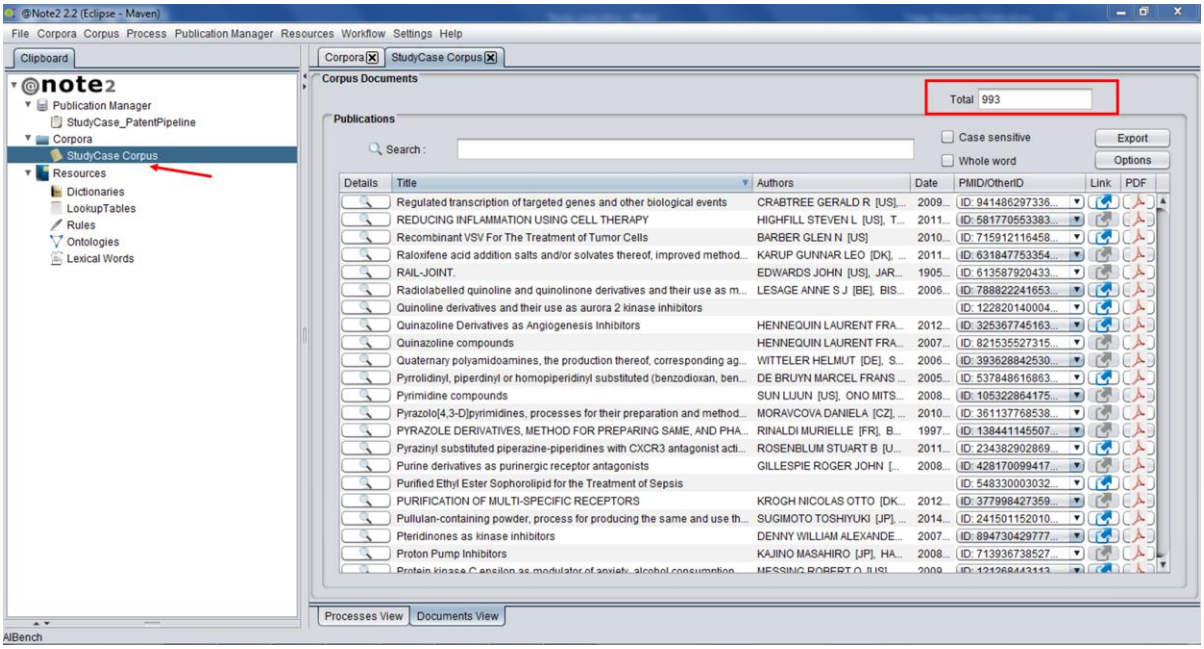


Figure 27 – The corpus obtained from the application of all the steps designed on the patent pipeline.

After the analysis of all approaches results and the corresponding parameter changes, it seems that changes in general *Tesseract* parameters have more influence on PDF to text conversion that the *init* only ones. That can be related to the algorithmic changes that these parameter changes leads to. However, conjugated with changes on the loading process, a more generalized and successful PDF to text conversion system can be achieved.

CONCLUSIONS AND FURTHER WORK

Biomedical text mining is a research field that has grown in the last years in biomedical research. Information retrieval is one of the BioTM fields with major importance since it is responsible for getting all the necessary information to be applied to the research field. Recently, patents have been the target of the scientific community to the application of BioTM techniques since they are a great information source for most knowledge areas.

Based on the @note2 structure, IR Search, and IR Crawling processes were designed and implemented in this thesis. Both processes allow the search and retrieval of patent information and respective documents based on keyword terms. Alongside those two processes, a pre-processing approach was made for the @note2 Corpora Module allowing the PDF to text conversion, obtaining machine-readable text.

These implemented processes were tested using a set of 1000 patents from the BioCreative V CHEMDNER task. This allowed testing the success rate for the PDF retrieval process, as well as the success rate for the patent metadata retrieval processes.

Since every patent PDF file uses the *Tess4J* API in the PDF to text conversion process, to test that process three approaches were made based on the configuration of the *Tesseract* application. We selected a configuration which allowed for over 85% of F-score, raising the recall to nearly 80%.

The main innovation of this work was the creation of new IR processes applied to patent texts surpassing several common problems related to searching and retrieving of that documents.

With this work, it is now possible getting, with great facility, several patent documents related to specific search terms in an extremely fast way and without the need to expend large amounts of time surfing on the several patent databases trying to find the desirable information. So, this framework opens several doors to the approximation of the scientific community to all the patent information with biological relevance using all sections of the published patent data, something that is not made until now.

Although the achieved system results were promising, the necessity of access keys to almost all the used components, all limitations imposed by using free versions of these credentials and the necessary processing time still are problems that can be appointed to this system.

However, some improvements can be done. All the designed tasks on this pipeline are composed of several components which follow an implementation pattern imposed by specific interfaces for each module. That modular architecture allows the integration of new module components, introducing other components on the patent pipeline improving the patent ID search, the metadata, and the PDF retrieval or the PDF to text conversion processes.

In the PDF to text conversion process, several post-processing modifications can be made to the returned text, removing all background noise that can influence the text integrity. Although some modifications are already implemented, there are some conversion errors that still remain on the converted text and new post-processing modifications could fix that.

The implemented system can be applied graphically into the @note2 platform by a GUI implementation. That will allow the quick input and configuration of each module without the necessity of Java programming or use a command line interface.

Technologically, the integration of these methods within @note2 allows developing an extensive set of possible text mining pipeline over patents, which were only accessible to scientific articles so far. The possibility of having quickly a large number of patents related to given keywords accomplished with several important information (including the respective full texts), allows the application of several BioTM approaches to patent texts. Among these approaches, several IE tools can use these texts to identify and annotate different biological entities, as well as relations between them, automating the search of structured information on patents and taking advantage of all the great informative capacity applied on these documents.

REFERENCES

Akhondi, S.A.; Klenner, A.G.; Tyrchan, C.; Manchala, A.K.; Boppana, K.; Lowe, D.; Zimmermann, M.; Jagarlapudi, S.A.; Sayle, R.; Kors, J.A.; Muresan, S. (2014) *Annotated chemical patent corpus: a gold standard for text mining*. PLoS One.9:e107477.

Akhondi, S.A.; Pons, E.; Afzal, Z.; Haagen, H.; Becker, B.; Hettne, K.M.; Mulligen, E.M.; Kors, J.A. (2015) *Patent mining: combining dictionary-based and machine-learning approaches*. Proceedings of the fifth BioCreative challenge evaluation workshop.

Álvarez, D.; Fernández, R.; Sánchez, L. (2015) *Stroke-based intelligent character recognition using a deterministic finite automaton*. Logic Journal of the IGPL.23:463-471.

Asif, A.M.A.M.; Hannan, S.A.; Perwej, Y.; Vithalrao, M.A. (2014) *An Overview and Applications of Optical Character Recognition*. International Journal of Advance Research In Science And Engineering.3.

Bach, N.; Badaskar, S. (2007) *A review of relation extraction*. Literature review for Language and Statistics II

Berners-Lee, T.; Fielding, R.; Masinter, L. (2005) *Uniform Resource Identifier (URI): Generic Syntax*. Acessed on: September 1, 2016. Available from: https://tools.ietf.org/html/rfc3986#section-1.1.1.

Breuel, T.M. (2008) *The OCRopus open source OCR system*. IS&T/SPIE 20th Annual Symposium.

Campos, D.; Matos, S.; Oliveira, J.L. (2013) *A modular framework for biomedical concept recognition*. BMC Bioinformatics.14:281.

Campos, D.; Matos, S.; Oliveira, J.L. (2015) *A document processing pipeline for annotating chemical entities in scientific documents*. J Cheminform.7:S7.

Chen, N.; Liu, Y.; Cheng, Y.; Liu, L.; Yan, Z.; Tao, L.; Guo, X.; Luo, Y.; Yan, A. (2015) *Technology Resource, Distribution, and Development Characteristics of Global Influenza Virus Vaccine: A Patent Bibliometric Analysis*. PLoS One.10:e0136953.

Clark, A.; Fox, C.; Lappin, S. (2010) *The Handbook of Computational Linguistics and Natural Language Processing.* Wiley-Blackwell.

Cohen, K.B.; Hunter, L. (2008) *Getting started in text mining*. PLoS Comput Biol.4:e20.

Cunningham, H.; Tablan, V.; Roberts, A.; Bontcheva, K. (2013) *Getting more out of biomedical documents with GATE's full lifecycle open source text analytics*. PLoS Comput Biol.9:e1002854.

Curbera, F.; Duftler, M.; Khalaf, R.; Nagy, W.; Mukhi, N.; Weerawarana, S. (2002) *Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI*. IEEE Internet Computing.6:86-93.

Cutter, M.; Manduchi, R. (2015) *Towards Mobile OCR: How To Take a Good Picture of a Document Without Sight*. Proc ACM Symp Doc Eng.2015:75-84.

ECMA. (2013) *Standard ECMA-404, The JSON Data Interchange Format*. 1st edition.

Edgar, R.C. (2004) *MUSCLE: a multiple sequence alignment method with reduced time and space complexity*. BMC Bioinformatics.5:113.

Eikvil, L. (1993) *Optical Character Recognition*.

Eisinger, D.; Tsatsaronis, G.; Bundschus, M.; Wieneke, U.; Schroeder, M. (2013) *Automated Patent Categorization and Guided Patent Search using IPC as Inspired by MeSH and PubMed*. J Biomed Semantics.4 Suppl 1:S3.

EPO. (2015) *Open Patent Services RESTful Web Services. Reference Guide.* European Patent Office.Version 1.2.14.

EPO. *Smart search - operators* (2016) [Accessed on: September, 11,2016]. Available from: https://worldwide.espacenet.com/help?locale=en_EP&method=handleHelpTopic&topic=operators.

Faro, A.; Giordano, D.; Spampinato, C. (2012) *Combining literature text mining with microarray data: advances for system biology modeling*. Brief Bioinform.13:61-82.

Fielding, R.T. (2000) *Architectural styles and the design of network-based software architectures*: University of California, Irvine.

Google. *JSON/Atom API Reference* (2013) [Accessed on: August, 20,2016]. Available from: https://developers.google.com/custom-search/json-api/v1/reference/cse/list.

Google. *Custom Search JSON/Atom API* (2015) [Accessed on: August 8,2016]. Available from: https://developers.google.com/custom-search/json-api/v1/using_rest.

Google. *Search Help* (2016) [Accessed on: September, 1,2016]. Available from: https://support.google.com/websearch/answer/2466433?hl=en&ref_topic=3081620.

Granlund, G.H. (1972) *Fourier Preprocessing for Hand Print Character Recognition*. Computers, IEEE Transactions on.C-21:195-201.

Heifets, A.; Jurisica, I. (2012) *SCRIPDB: a portal for easy access to syntheses, chemicals and reactions in patents*. Nucleic Acids Res.40:D428-433.

Hoffmann, R.; Valencia, A. (2004) *A gene network for navigating the literature*. Nat Genet.36:664.

Holley, R. (2009) *How Good Can It Get? Analysing and Improving OCR Accuracy in Large Scale Historic Newspaper Digitisation Programs* D-Lib Magazine.15.

Hossain, M.Z.; Amin, M.A.; Yan, H. (2012) *Rapid Feature Extraction for Optical Character Recognition*. CoRR.abs/1206.0238.

Hugunin, J. (1997) *Python and Java: The best of both worlds*. 6th international Python conference.

Hunter, L.; Cohen, K.B. (2006) *Biomedical language processing: what's beyond PubMed?* Mol Cell.21:589-594.

Jeong, C.; Kim, K. (2014) *Creating patents on the new technology using analogy-based patent mining*. Expert Systems with Application.41:3605–3614.

Jessop, D.M.; Adams, S.E.; Willighagen, E.L.; Hawizy, L.; Murray-Rust, P. (2011) *OSCAR4: a flexible architecture for chemical text-mining*. J Cheminform.3:41.

Kemp, N.; Lynch, M. (1998) *Extraction of Information from the Text of Chemical Patents. 1. Identification of Specific Chemical Names.* Journal of Chemical Information and Computer Sciences.38:544-551.

Kiss, M.; Nagy, Á.; Vincze, V.; Almási, A.; Alexin, Z.; Csirik, J. (2012) *A Manuall Annotated Corpus of Pharmaceutical Patents*. Springer Berlin Heidelberg\.135–142.

Klinger, R.; Kolarik, C.; Fluck, J.; Hofmann-Apitius, M.; Friedrich, C.M. (2008) *Detection of IUPAC and IUPAC-like chemical names*. Bioinformatics.24:i268-276.

Krallinger, M.; Leitner, F.; Rabal, O.; Vazquez, M.; Oyarzabal, J.; Valencia, A. (2013) *Overview of the chemical compound and drug name recognition (CHEMDNER) task*. Proceedings of the fourth BioCreative challenge evaluation workshop.2.

Krallinger, M.; Leitner, F.; Rabal, O.; Vazquez, M.; Oyarzabal, J.; Valencia, A. (2015) *CHEMDNER: The drugs and chemical names extraction challenge*. J Cheminform.7:S1.

Krallinger, M.; Valencia, A. (2005) *Text-mining and information-retrieval services for molecular biology*. Genome Biol.6:224.

Laffleur, F. (2016) *Mucoadhesive therapeutic compositions: a patent review (2011-2014)*. Expert Opin Ther Pat.

Latimer, M.T. (2005) *Patenting inventions arising from biological research*. Genome Biol.6:203.

Leaman, R.; Wei, C.H.; Lu, Z. (2015) *tmChem: a high performance approach for chemical named entity recognition and normalization*. J Cheminform.7:S3.

Liu, X.; Bordes, A.; Grandvalet, Y. (2015) *Extracting biomedical events from pairs of text entities*. BMC Bioinformatics.16 Suppl 10:S8.

Lourenço, A.; Carreira, R.; Carneiro, S.; Maia, P.; Glez-Peña, D.; Fdez-Riverola, F.; Ferreira, E.C.; Rocha, I.; Rocha, M. (2009) *@Note: A workbench for Biomedical Text Mining*. Journal of Biomedical Informatics.42:710-720.

Lu, Z. (2011) *PubMed and beyond: a survey of web tools for searching biomedical literature*. Database (Oxford).2011:baq036.

Mathiak, B.; Eckstein, S. (2004) *Five Steps to Text Mining in Biomedical Literature*. Proceedings of the Second European Workshop on Data Mining and Text Mining in Bioinformatics.

Microsoft. *Bing Search API* (2012) [Accessed on: September 1,2016]. Available from: https://datamarket.azure.com/dataset/bing/search#schema.

Miner, G.; Elder, J.; Hill, T.; Nisbet, R.; Delen, D.; A., F. (2012) *Practical text mining and statistical analysis for non-structured text data applications.* Academic Press.

Nottingham, M.; Sayre, R. (2005) *The Atom Syndication Format RFC 4287*. Acessed on: August 15, 2016. Available from: https://tools.ietf.org/html/rfc4287.

Oldham, P.; Hall, S.; Forero, O. (2013) *Biological diversity in the patent system*. PLoS One.8:e78737.

Papadatos, G.; Davies, M.; Dedman, N.; Chambers, J.; Gaulton, A.; Siddle, J.; Koks, R.; Irvine, S.A.; Pettersson, J.; Goncharoff, N.; Hersey, A.; Overington, J.P. (2016) *SureChEMBL: a large-scale, chemically annotated patent document database*. Nucleic Acids Res.44:D1220-1228.

Parberry, I.; Gasarch, W. (2002) *Problems on Algorithms, Second Edition*. Englewood Cliffs, N.J. Prentice Hall. 179 p. p.

Park, J.; Rosania, G.R.; Shedden, K.A.; Nguyen, M.; Lyu, N.; Saitou, K. (2009) *Automated extraction of chemical structure information from digital raster images*. Chem Cent J.3:4.

Pasche, E.; Gobeill, J.; Kreim, O.; Oezdemir-Zaech, F.; Vachon, T.; Lovis, C.; Ruch, P. (2014) *Development and tuning of an original search engine for patent libraries in medicinal chemistry*. BMC Bioinformatics.15 Suppl 1:S15.

Patel, C.; Patel, A.; Patel, D. (2012) *Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study*. International Journal of Computer Applications.55.

Reddy, C.S.; Ghai, R.; Rashmi; Kalia, V.C. (2003) *Polyhydroxyalkanoates: an overview*. Bioresour Technol.87:137-146.

Reiners, P.D. (2008) *Dynamic programming and sequence alignment*. Acessed on: September 12, 2016. Available from: http://www.ibm.com/developerworks/library/j-seqalign/.

Salgado, D.; Krallinger, M.; Depaule, M.; Drula, E.; Tendulkar, A.V.; Leitner, F.; Valencia, A.; Marcelle, C. (2012) *MyMiner: a web application for computer-assisted biocuration and text annotation*. Bioinformatics.28:2285-2287.

Sayle, R.; Xie, P.H.; Muresan, S. (2012) *Improved chemical text mining of patents with infinite dictionaries and automatic spelling correction*. J Chem Inf Model.52:51-62.

Settles, B. (2005) *ABNER: an open source tool for automatically tagging genes, proteins and other entity names in text*. Bioinformatics.21:3191-3192.

Shatkay, H.; Feldman, R. (2003) *Mining the biomedical literature in the genomic era: an overview*. J Comput Biol.10:821-855.

Singh, R.; Kumar, S. (2016) *An Overview of World Wide Web Protocol (Hypertext Transfer Protocol and Hypertext Transfer Protocol Secure*. International Journal of Advanced Research in Computer Science and Software Engineering.6:396-399.

Smith, R. (2007) *An Overview of the Tesseract OCR Engine*. IEEE Ninth International Conference.

Thessen, A.E.; Cui, H.; Mozzherin, D. (2012) *Applications of natural language processing in biodiversity science*. Adv Bioinformatics.2012:391574.

Trier, Ø.D.; Jain, A.K.; Taxt, T. (1996) *Feature extraction methods for character recognition-A survey*. Pattern Recognition.29:641-662.

USNLM. (2016) [Accessed on: September, 21,2016]. Available from: https://www.nlm.nih.gov/bsd/medline_cit_counts_yr_pub.html.

Waring, P. (2012) *PCT PATENTSCOPE Web-services for Offices*. Acessed on: August 18, 2016. Available from: http://www.wipo.int/edocs/mdocs/pct/en/wipo_pct_mow_12/wipo_pct_mow_12_ref_pctpatentscope.pdf.

WIPO. *WIPO IP Statistics* (2014) [Accessed on: July, 24,2016]. Available from: http://ipstats.wipo.int/ipstatv2/keysearch.htm?keyId=201.

WIPO. (2015a) *Guidelines for Preparing Patent Landscape Reports*.

WIPO. (2015b) *Internation Patent Classification*

WIPO. (2015c) *WIPO Guide to Using Patent Information*.

WIPO. (2015d) *World Intellectual Property Indicators, 2015 edition*. World Intellectual Property Organization - Economics and Statistics Division.

WIPO. *PATENTSCOPE Data Services* (2016) [Accessed on: August 19,2016]. Available from: http://www.wipo.int/patentscope/en/data/.

Wu, C.; Schwartz, J.M.; Brabant, G.; Peng, S.L.; Nenadic, G. (2015) *Constructing a molecular interaction network for thyroid cancer via large-scale text mining of gene and pathway events*. BMC Syst Biol.9 Suppl 6:S5.

Xu, S.; An, X.; Zhu, L.; Zhang, Y.; Zhang, H. (2015) *A CRF-based system for recognizing chemical entity mentions (CEMs) in biomedical literature*. J Cheminform.7:S11.

Zweigenbaum, P.; Demner-Fushman, D.; Yu, H.; Cohen, K.B. (2007) *Frontiers of biomedical text mining: current progress*. Brief Bioinform.8:358-375.

APPENDIX I – PERCENT-ENCODING PROCESS APPLIED TO THE SEARCH SOURCES MODULE COMPONENTS. THE RESERVED CHARACTERS REPRESENT THE URI SPECIFIC CHARACTERS WITH SPECIAL FUNCTIONS AND THE UNSECURE CHARACTERS ARE THOSE WHO CAN BE EASILY CONFUSED WITH OTHER CHARACTERS INSIDE THE URI.

Character type	Invalid character	Percent-encoding replacement	*search sources module* components		
			***Custom Search* API**	***Bing Search* API**	***OPS web services* API**
Reserved characters	?	%3F	•	•	•
	@	%40	•	•	•
	#	%23	•	•	•
	!	%21	•	•	
	‘	%27	•	•	
	(	%28	•	•	
	)	%29	•	•	
	;	%3B	•	•	•
	:	%3A	•	•	•
	&	%26	•	•	•
	=	%3D	•	•	•
	+	%2B	•	•	•
	$	%24	•	•	•
	,	%2C	•	•	•
	/	%2F	•	•	
	[	%5B	•	•	•
	]	%5D	•	•	•
Unsecure Characters	%	%25	•	•	•
	SPACE	%20			•
	\	%5C	•	•	
	“	%22	•	•	•
	>	%3E	•	•	•
	<	%3C	•	•	•
	{	%7B	•	•	•
	}	%7D	•	•	•
	^	%5E	•	•	•
	~	%7E	•	•	•

APPENDIX II – SOME *TESSERACT* PARAMETERS THAT CAN BE USED TO CHANGE THE ALGORITHM BEHAVIOR AND RESPECTIVE DESCRIPTION.

Type	*Tesseract* parameters	Description
Init only parameters	*image_default_resolution*	Allows rescaling the PDF images by changing the dots per inch (DPI). It must be defined to 300DPI or better.
	editor_image_blob_bb_color	Allows a random variation on image parameters as brightness or color to make the text easier to read.
	editor_image_xpos	Allows the page rotation getting the text lines presented horizontally.
	tessedit_image_border	Allows the definition of border limits from which blobs are ignored.
	load_system_dawg	Allows the dictionary correction enabling or disabling.
	user_words_suffix	Allows the introduction of user words on the used dictionary for a given language.
	load_punc_dawg	Allows the definition of several rules for punctuation.
General parameters	*tessedit_pageseg_mode*	Defines the page segmentation mode allowing the OCR application only in a few page sections.
	matcher_good_threshold	Defines a threshold for a good match between the founded words and the dictionary words previously loaded (lower the better).
	heuristic_segcost_rating_base	Defines the multiplying factor to calculate the segmentation cost, adding it to word rating.
	language_model_penalty_punc	Defines the penalty value for the founded inconsistent punctuation.
	textord_linespace_iqrlimit	Defines the maximum value for the median of line space.
	textord_min_linesize	Allows the minimum line size value from which lines are ignored.
	language_model_ngram_space_delimited_language	Defines if words are delimited by space or not.
Debug parameters	*textord_debug_bugs*	Allows the returning of the output related to founded bugs related with text lines.
	textord_tabfind_show_reject_blobs	Allows the printing of the blobs that were considered as noise.
	textord_debug_images	Allows the use of a greyed image background to debug the text lines identification process.
	dawg_debug_level	Allows controlling the quantity of the debug messages (general debug info, detailed debug info or all the debug messages).
	tessdata_manager_debug_level	Allows control the debugging process related with functions that manage the tesseract data (e.g. language dictionaries).