



**André Jerónimo
Martins dos Santos**

**Anotação automática e interativa de documentos
PDF**

**Automatic and interactive annotation of PDF
documents**



**André Jerónimo
Martins dos Santos**

**Anotação automática e interativa de documentos
PDF**

**Automatic and interactive annotation of PDF
documents**

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Sérgio Guilherme Aleixo de Matos, Investigador Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Prof. Doutor José Luís Oliveira

Professor Associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Sérgio Sobral Nunes

Professor Auxiliar do Departamento de Engenharia Informática da Faculdade de Engenharia da Universidade do Porto

Doutor Sérgio Guilherme Aleixo de Matos

Investigador Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro (orientador)

agradecimentos / acknowledgements

Gostava de agradecer, em primeiro lugar ao meu orientador, Sérgio Matos, pela oportunidade, orientação e apoio incondicional durante o mestrado. Os seus conselhos, recomendações e discussões foram fundamentais para a realização deste trabalho. Também quero agradecer a todos os elementos do grupo de Bioinformática por proporcionarem um ambiente divertido e de entreajuda no nosso local de trabalho.

Agradeço também ao David Campos pela ajuda e conselhos que contribuíram para a execução deste trabalho.

Quero também agradecer aos meus amigos, Ivo Silva, José Sequeira, Ricardo Martins, Tiago Henriques, Vasco Santos e Miguel Vicente pelas discussões e momentos de convívio durante o mestrado. Agradeço ainda aos meus amigos Catarina Jorge, Miguel Oliveira, Rui Lebre, André Marques, Nuno Marques, Daniel Siva e Tiago Magalhães pelo apoio e amizade incondicional.

Aproveito ainda para agradecer a todos os que me apoiaram ao longo do meu percurso académico, em particular aos meus colegas do curso de Mestrado Integrado em Engenharia de Computares e Telemática, e a todos aqueles com que tive o privilégio de trabalhar na Associação BEST Aveiro da Universidade de Aveiro.

Finalmente, um agradecimento especial aos meus pais, Carlos Santos e Ângela Santos, e à minha irmã, Lara Santos, por toda a força, motivação, apoio, educação e amor que me deram desde sempre. À minha família, em especial, dedico esta dissertação.

Palavras Chave

Bioinformática, extração de informação, reconhecimento de conceitos, mineração de texto, anotação de documentos biomédicos, portable document format.

Resumo

O aumento acelerado da literatura biomédica levou ao desenvolvimento de vários esforços para extrair e armazenar, de forma estruturada, a informação relativa aos conceitos e relações presentes nesses textos, oferecendo aos investigadores e clínicos um acesso rápido e fácil à informação. No entanto, este processo de "curadoria de conhecimento" é uma tarefa extremamente exaustiva, sendo cada vez mais comum o uso de ferramentas de anotação automática, fazendo uso de técnicas de mineração de texto. Apesar de já existirem sistemas de anotação bastante completos e que apresentam um alto desempenho, estes não são largamente usados pela comunidade biomédica, principalmente por serem complexos e apresentarem limitações ao nível de usabilidade. Por outro lado, o PDF tornou-se nos últimos anos num dos formatos mais populares para publicar e partilhar documentos visto poder ser apresentado exatamente da mesma maneira independentemente do sistema ou plataforma em que é acedido. A maioria das ferramentas de anotação foram principalmente desenhadas para extrair informação de texto livre, contudo hoje em dia uma grande parte da literatura biomédica é publicada e distribuída em PDF, e portanto a extração de informação de documentos PDF deve ser um ponto de foco para a comunidade de mineração de texto biomédico.

O objetivo do trabalho descrito nesta dissertação foi a extensão da *framework* Neji, permitindo o processamento de documentos em formato PDF, e a integração dessas funcionalidades na plataforma Egas, permitindo que um utilizador possa visualizar e anotar, simultaneamente, o artigo original no formato PDF e o texto extraído deste.

Os sistemas desenvolvidos apresentam bons resultados de desempenho, tanto em termos de velocidade de processamento como de representação da informação, o que também contribui para uma melhor experiência de utilizador. Além disso, apresentam várias vantagens para a comunidade de mineração de texto e curadores, permitindo a anotação direta de artigos no formato PDF e simplificando o uso e configuração destes sistemas de anotação por parte de investigadores.

Keywords

Bioinformatics, information extraction, concept recognition, text mining, relation mining, biomedical curation, portable document format.

Abstract

The accelerated increase of the biomedical literature has led to various efforts to extract and store, in a structured way, the information related with the concepts and relations presented in those texts, providing to investigators and researchers a fast and easy access to knowledge. However, this process of “knowledge curation” is an extremely exhaustive task, being more and more common demanding the application of automatic annotation tools, that make use of text mining techniques. Even though complete annotation systems already exist and produce high performance results, they are not widely used by the biomedical community, mainly because of their complexity and also due to some limitations in usability. On the other hand, the PDF has become in the last years one of the most popular formats for publishing and sharing documents because of it can be displayed exactly in the same way independently of the system or platform where it is accessed. The majority of annotation tools were mainly designed to extract information from raw text, although a big part of the biomedical literature is published and distributed in PDF, and thus the information extraction from PDF documents should be a focus point for the biomedical text mining community.

The objective of the work described in this document is the extension of Neji framework, allowing the processing of documents in PDF format, and the integration of these features in Egas platform, allowing a user to simultaneously visualize the original article in PDF format and its extracted text.

The improved and developed systems present good performing results, both in terms of processing speed and representation of the information, contributing also for a better user experience. Besides that, they present several advantages for the biomedical community, allowing the direct annotation of PDF articles and simplifying the use and configuration of these annotation systems by researchers.

CONTENTS

CONTENTS	i
LIST OF FIGURES	iii
LIST OF TABLES	v
ACRONYMS	vii
1 INTRODUCTION	1
1.1 Biomedical information extraction	1
1.2 The PDF format	2
1.3 Motivation and objectives	2
1.4 Results	3
1.5 Thesis outline	4
2 BACKGROUND	5
2.1 Biomedical information extraction	5
2.1.1 Preliminaries	6
2.1.2 Concept recognition	11
2.1.3 Biomedical concept recognition systems	14
2.2 Interactive biomedical curation	19
2.2.1 Web-based interactive curating platforms	19
2.3 PDF format	25
2.3.1 Text extraction from PDF	25
2.3.2 Information extraction challenges	29
2.3.3 Tools	30
3 ARCHITECTURE AND IMPLEMENTATION	39
3.1 General architecture	39
3.2 Neji	40
3.2.1 Pipeline and modules	40
3.2.2 PDF information extraction	43
3.2.3 Neji web services	49
3.3 Egas	56
3.3.1 Architecture	56
3.3.2 PDF handling and visualization	57

4	APPLICATION AND RESULTS	63
4.1	Neji	63
4.1.1	Neji web services	66
4.2	Egas	77
4.3	Results	82
4.3.1	Corpora	83
4.3.2	Resources	83
4.3.3	PDF text extraction evaluation	83
4.3.4	PDF processing speed evaluation	84
5	CONCLUSION	87
	BIBLIOGRAPHY	89

LIST OF FIGURES

2.1	Medical Literature Analysis and Retrieval System Online (MEDLINE) growth over the last ten years. (adapted from [3])	6
2.2	Illustration of Natural Language Processing (NLP) linguistic processing tasks. (from [20])	10
2.3	Illustration of the biomedical concept recognition task.	11
2.4	General processing pipeline of Named entity recognition (NER) solutions.	12
2.5	Neji processing pipeline. (from [27])	14
2.6	National Center for Biomedical Ontology (NCBO) Annotator workflow. (from [30])	16
2.7	NCBO Annotator annotation example.	16
2.8	Whatizit annotation example using its graphical interface.	18
2.9	Egas entity annotation example. (from [34])	20
2.10	Brat Rapid Annotation Tool (BRAT) entity annotation example. (from [35]) . .	21
2.11	MyMiner entity tagging example.	22
2.12	PubTator annotation example.	23
2.13	tagtog annotation example.	24
2.14	Portable Document Format (PDF) file basic structure.	25
2.15	PDF file structure.	27
2.16	PDF document structure.	28
2.17	PDFBox extraction example.	31
2.18	Tika extraction example.	32
2.19	PDFxStream architecture. (from [47])	32
2.20	PDFTextStream relative performance. (from [47])	33
2.21	PDFTextStream document model. (from [47])	33
2.22	PDFxStream extraction example.	34
2.23	LA-PDFText block identification algorithm. Left side shows a PDF page; right side shows the corresponding blocks identification made by LA-PDFText for that page. (from [48])	35
2.24	LA-PDFText extraction example.	36
2.25	pdftotext extraction example.	37
3.1	General architecture diagram.	40
3.2	Neji processing pipeline and modular architecture. (from [27])	41
3.3	Neji dictionary example.	42
3.4	Left side shows a PDF page; right side shows the corresponding text blocks identification for that page.	44

3.5	Position mapping example for the sentence “The interaction of olfactory (or odorant) receptors with their odorant ligands is the first step in signal transduction pathway that results in the perception of smell.”	46
3.6	Neji web services architecture diagram.	49
3.7	Neji web services data structure.	50
3.8	Egas architecture diagram. (from [34])	56
4.1	Neji command line interface.	64
4.2	Example of annotation using Neji with a simple PDF file.	65
4.3	Neji web services home page.	66
4.4	Neji web services dictionaries page.	67
4.5	Neji web services add dictionary form.	67
4.6	Neji web services models page.	68
4.7	Neji web services add model form.	68
4.8	Neji web services annotation services page.	69
4.9	Neji web services add service form.	69
4.10	Neji web services annotation input page.	70
4.11	Neji web services pubmed article pop-up.	71
4.12	Neji web services annotation output page.	71
4.13	Neji web services concept popover example.	72
4.14	Neji web services exporting results example.	72
4.15	Example of using the annotate plain text web service.	73
4.16	Example of using the export plain text web service.	74
4.17	Example of using the extract PDF document text web service.	75
4.18	Example of using the annotate PDF document web service.	75
4.19	Example of using the export web service for PDF document.	76
4.20	Example of importing a PDF article.	78
4.21	Visualization of a PDF article in Egas.	78
4.22	Visualization of a text article.	79
4.23	Synchronization between annotation area and PDF visualizer.	79
4.24	Synchronization between PDF visualizer and annotation area.	80
4.25	Adding a new annotation service to a project.	80
4.26	Selecting a new annotation service.	81
4.29	Enable and disable of PDF viewer features.	81
4.27	Annotation result using an annotation service from Neji web services.	82
4.28	Example of a PDF visualization with annotations.	82

LIST OF TABLES

2.1	Biomedical concept recognition systems comparison.	18
2.2	Performance extraction tools comparison.	24
2.3	PDF extraction tools comparison.	38
2.4	Performance extraction tools comparison.	38
4.1	Parameters of annotate text web service.	72
4.2	Response of annotate text web service.	73
4.3	Parameters of export web service.	73
4.4	Parameters of extract PDF document text web service.	74
4.5	Parameters of annotate PDF document web service.	74
4.6	Parameters of export web service for PDF documents.	76
4.7	PDF text extraction evaluation using the Neji framework.	84
4.8	PDF processing speed evaluation using the Neji framework.	84
4.9	PDF processing speed evaluation using the Neji web services.	85

ACRONYMS

API	Application Programming Interface
BRAT	Brat Rapid Annotation Tool
CLI	Command Line Interface
CRF	Conditional Random Field
CSS	Cascading Style Sheets
DFA	Deterministic Finite Automaton
FIFO	First In First Out
TP	False Negative
FP	False Positive
GO	Gene Ontology
GSC	Gold Standard Corpora
HTML	Hypertext Markup Language
HTTPS	Hypertext Transfer Protocol Secure
IE	information extraction
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
MEDLINE	Medical Literature Analysis and Retrieval System Online
ML	Machine Learning
NCBI	National Center for Biotechnology Information
NCBO	National Center for Biomedical Ontology
NCIBI	National Center for Biotechnology Information
NER	Named entity recognition
NLM	United States National Library of Medicine
NLP	Natural Language Processing
PDF	Portable Document Format
PMCID	PubMed Central reference number
PMID	PubMed Unique Identifier
POS	Part-of-Speech

REST	Representational State Transfer
SQL	Structured Query Language
SSC	Silver Standard Corpora
TM	Text Mining
TN	True Negative
TP	True Positive
TSV	Tab-separated values
UIMA	Unstructured Information Management Architecture
UMLS	Unified Medical Language System
UniProt	Universal Protein Resource
URI	Uniform Resource Identifier
XFA	XML Forms Architecture
XHTML	eXtensible Hypertext Markup Language
XML	eXtensible Markup Language
XMP	Extensible Metadata Platform

CHAPTER 1

INTRODUCTION

The need to store and share information started being manifested many years ago. From the beginning of the human race the more common objects started being used for storing information, such as stones, silicon and petroglyphs, and it ended evolving to the current efficient and portable devices that surround our society. A recent study, presented by Hilbert and López [1] shows that the World capacity to store and exchange information is growing at a growth rate of at least 23% per year in the last 20 years. In 1986, the world had the capacity to store almost 540 MB per person, and 20 years later, in 2007, this amount grew to approximately 43 GB per person.

1.1 BIOMEDICAL INFORMATION EXTRACTION

Information extraction is the process of examining text in order to find relevant information, that in biomedical domain would be collecting information about biomedical entities, relations and events, such as, proteins, genes and chemicals [2].

Nowadays a large amount of biomedicine information is continuously being produced, verifying also a fast growth in the number of published biomedicine documents, such as articles, books and journals. By the end of 2015, Medical Literature Analysis and Retrieval System Online (MEDLINE), which is a bibliographic database of life sciences and biomedical information compiled by the United States National Library of Medicine (NLM), contained over 22 million references to journal articles in life sciences [3]. At this rate it is impossible to manage all the data, making it difficult for researchers to stay updated with the current information and have access to the relevant publications in their study fields, limiting their potential knowledge.

In order to extract relevant information from the documents, maintaining the existing knowledge updated, several biomedical resources started by manually curate scientific articles. However, with the rapid growth of data, this task became unfeasible because it is expensive, since it requires expert curators to do it, and it takes a lot of time and effort to be done correctly and accurately. Thus, automatic text annotation became an important focus, allowing researchers to extract and identify the relevant information mentioned in scientific articles. This need for automatic and fast information extraction led

to the emergence and development of various computational solutions to extract biomedical information from scientific publications, using different techniques and algorithms.

Several biomedical information extraction tools and applications have already been developed and currently produce accurate results in various scenarios, however they are often hard to use and configure by researchers. Thus several web platforms were also developed in order to facilitate the use of automatic solutions to curate documents by presenting intuitive and interactive interfaces to the users. However, there is still a large margin to develop tools that can be widely used by the biomedical community.

1.2 THE PDF FORMAT

Portable Document Format (PDF) is today one of the most popular electronic file formats for publishing documents on the web. As its name suggests, portability is clearly one of its strongest points. It was the first format solving a very important problem, that is to be shown exactly the same everywhere, independently of the operating system, hardware or software applications used to access it.

The idea of a portable format was first conceived by John Warnock, Adobe Systems' co-founder, who in 1991 outlined a system he called The Camelot Project [4], whose goal was to develop a new file format capable of performing in the same way, digitally and faster, independently of the system and platform. In the following year, this project evolved to PDF.

PDF has become one of the most popular file formats for publishing and sharing documents because of the several advantages it offers. It is easy to create, read and use by everyone. It offers options for protecting the file contents by defining different levels of access, such as passwords, digital signatures or watermarks. It can be compacted with no loss of quality, facilitating the exchange of documents. Besides that, PDF reading software was free from almost the beginning of PDF. This fact led to a big and fast adoption of PDF, making it today one of the most popular electronic file formats, and an open standard for electronic document exchange.

But this format has some disadvantages too. By its nature PDF documents are not editable, because it is simply an image of a document, so using and editing its contents can be a hard task. On the other hand, there are various types of PDFs, requiring different ways to work with them, for example for extracting and searching information.

The PDF format provides a lot of advantages and it is clearly the preferred method for publishing and distributing documents on the web, but it still presents important challenges if we want to work with its information.

1.3 MOTIVATION AND OBJECTIVES

As stated in this chapter, with the increasing amount of biomedical literature being produced every day, several efforts have been made in order to extract information related with the concepts and relations presented in those texts, offering to researchers and investigators a fast and easy access to knowledge about genes, proteins, drugs and interactions between these and other biomedical concepts.

However, this process of “knowledge curation” is an extremely exhaustive task, being more and more common the use of automatic annotation tools, that make use of Text Mining (TM) techniques.

These tools were mainly designed to extract information from raw text, but today a considerable part of the biomedical literature is published and distributed in PDF because of the already addressed advantages of that format. In the absence of tools that allow the extraction of biomedical information from PDF documents, it should clearly be a focus point since it facilitates this process, avoiding the need to extract the text from the documents before using it in the biomedical information extraction tools.

The main goal of the work described in this thesis was therefore, to develop a way to extract biomedical information from PDF documents, such as concepts and relations, using and extending the current state of the art methods and tools in text mining. And allowing its usage by researchers and curators through a simple and intuitive curating platform.

This thesis objectives can then be defined as:

- Allow the extraction of text contained in PDF files;
- Extend Neji framework to integrate the PDF text extraction and perform the biomedical concept annotation to the extracted text;
- Integration of PDF annotation in EGAS platform;
- Possibly the simultaneous visualization of the original PDF file and the extracted text in Egas, allowing the “navigation” between both zones, synchronizing the text annotation area and the PDF visualization area.

1.4 RESULTS

The work developed in this thesis generated improvements and implementation of new features and systems:

- Extension of Neji framework to allow the processing of PDF documents;
- Development of Neji web services, a web-services platform that facilitates the use and integration of Neji in other text mining pipelines, providing a public Representational State Transfer (REST)ful Application Programming Interface (API) for biomedical concept recognition, and an interactive management and annotation interface;
- Integration of PDF information extraction in Egas platform, allowing a user to simultaneously visualize the original article and its extracted text, and to navigate between those zones synchronously.

Three publications have also resulted from this work:

- Sérgio Matos, André Santos, David Campos, José Luís Oliveira. Neji: a BioC compatible framework for biomedical concept recognition. Proceedings of the Fifth BioCreative Challenge Evaluation Workshop, Sevilla, Spain, p. 17-21, September 2015;

- Kim, Sun et al. BioCreative V BioC Track Overview: Collaborative Biocurator Assistant Task for BioGRID, Database (under review);
- André Santos, Sérgio Matos, David Campos, José Luís Oliveira. A curation pipeline and web-services for PDF documents, 7th International Symposium on Semantic Mining in Biomedicine, Potsdam, Germany, August 2016.

1.5 THESIS OUTLINE

The next chapters of this thesis are organized in the following sequence:

- Chapter 2 presents a detailed analysis of the state of the art in biomedical information extraction (IE), interactive curation platforms and PDF text extraction. Firstly some techniques and methods used in information extraction area. Then are described some of the most used and reliable tools and applications;
- Chapter 3 presents an detailed description of the architecture and implementation of the developed solutions in the scope of this thesis, namely Neji, Neji web services and Egas. To support this some related technologies are also addressed and described in this chapter;
- Chapter 4 presents the final application and results of the developed solutions. It describes the new and improved features of each system, explain their usage and presents some screenshots of the applications interface. Besides that are also presented the results obtained by evaluating Neji in terms of PDF text extraction and processing speed;
- Chapter 5 presents the conclusions of this thesis.

BACKGROUND

In this chapter is presented a detailed analysis of the state of the art in the biomedical IE, interactive curation and PDF text extraction. For each domain is firstly described the techniques and methods used nowadays, and then is presented some of the most used and accurate tools.

2.1 BIOMEDICAL INFORMATION EXTRACTION

Nowadays, a large amount of biomedical information is continuously being produced, contributing to a fast growth in the number of published articles, journals, books and biomedical reports. It is then important to find efficient ways to analyse and extract knowledge from these data and store it in structured databases. Due to this fact, there has been increasing interest in the last years in information extraction for biomedical area.

In the last 10 years, MEDLINE database has been growing at rate of approximately 4% per year [3]. While in 2005 there were approximately 15 million publications in MEDLINE, in 2015 it had more than 22 million publications, resultant from the addition of 2000 to 4000 new entries every day (see figure 2.1).

To maintain their knowledge updated, many biomedical resources such as MEDLINE started to manually curate their articles. However, with this rapid growth of information this task proved to be difficult and expensive to do, since it needs to be done by experts and requires a lot of time. This led to an increasing interest in the development of automatic solutions for biomedical information extraction from scientific articles, in order to maintain knowledge databases updated [5].

The biomedical domain, like many other scientific domains, is complex and presents a lot of challenges in the application of text mining (TM). In first place it is divided in many fields and sub-fields, being hard to link concepts between them. Additionally, more information is constantly being produced. On the other hand, in some cases a term can be ambiguous, i.e, the same term can represent different concepts; and a concept may be represented by different names. Due to all these challenges, the development of TM solutions that perform well in biomedical area are seen has a good prove of their application and it is assumed that they will perform equally well in different and simpler domains [6].

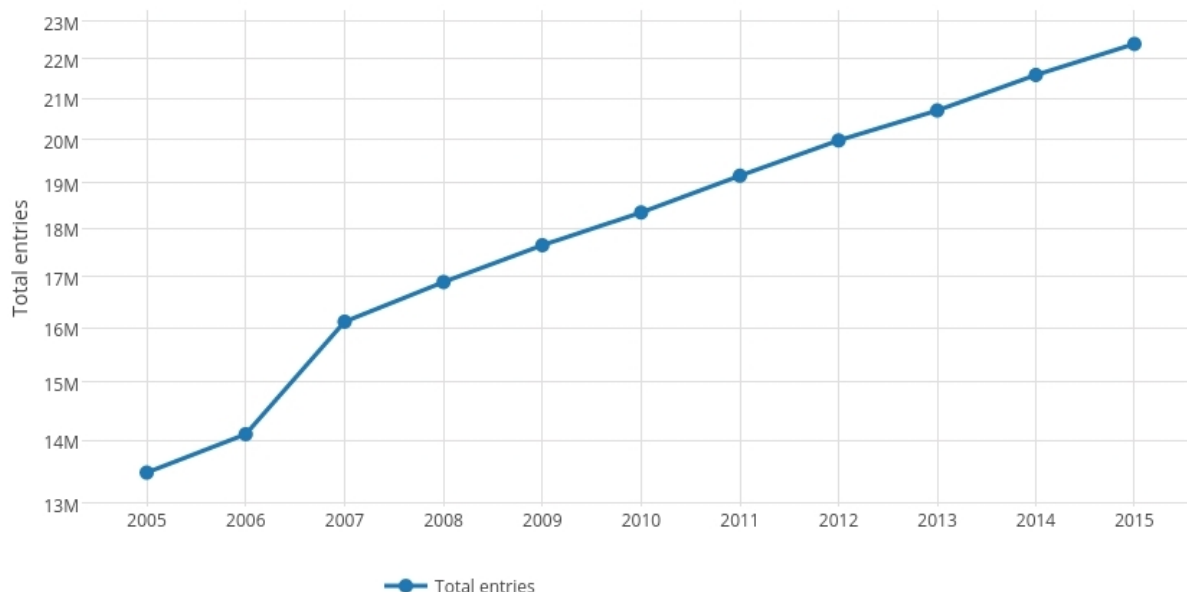


Figure 2.1: MEDLINE growth over the last ten years. (adapted from [3])

The curation process requires always a manual part, so automatic solutions present significant benefits since they accelerate this process, and thus it is less expensive. Although, they are not so precise as expert curators, since it is not easy to collect all curator’s domain knowledge into a unique structured representation. Even different curators have different interpretations of the same data and terms [7]. These facts can affect the quality and precision of the extraction results and so it is important to carefully collect biomedical information from scientific articles, helping to maintain the knowledge databases updated and improving the discovery of new knowledge.

The applicability of TM solutions in real life problems had already been proved to be useful, mainly in pharmacovigilance, drug discovery and drug repurposing [8]. The beneficial effects of fish oil to patients with Raynaud’s disease, and the potential of magnesium to treat migraines were the first hypothesis discovered by TM solutions. These results were both presented by Swanson, in 1990, and were validated in clinical trials [9][10], becoming well established techniques in nowadays.

One of the essential tasks that biomedical information extraction focuses on is concept recognition. In the following pages a detailed description and analysis of this task is presented, covering their goals, challenges, techniques and existing solutions.

2.1.1 PRELIMINARIES

Several IE tasks have common processing tasks and use the same resources, for that reason we will start by describing them. This preliminary tasks include the collecting of resources to support the development and evaluation, definition of evaluation metrics to analyse and compare IE solutions, and the use of pre-processing methods to allow the automatic application of IE approaches.

RESOURCES

Resources are used to support the development of biomedical IE solutions by providing fundamental data that allows the development of these automatic solutions. On the other hand, these data can be used to test, validate and compare the results of different IE solutions.

Knowledge bases

Knowledge bases are one of the most important resources for biomedical IE domain, with special attention to databases and ontologies. While ontologies provide a good way to represent the reality, databases are better to store and perform searches when facing big amounts of data [11]. In general, knowledge bases focus on collecting detailed information regarding a specific type of concepts, such as genes and proteins, enzymes, chemicals, disorders and species. Universal Protein Resource (UniProt) [12] is a well known database that provides a centralized repository of protein sequence and function information. On the other hand, Gene Ontology (GO) [13] provides a set of structured vocabularies for specific biological domains, which are commonly used to represent gene and gene product attributes across all species.

Corpora

Another essential resource for biomedical IE are corpora. A corpus is a large set of text documents that commonly contains annotations on a specific biomedical domain, used to develop and evaluate the performance and accuracy of IE solutions. Through this evaluation it is possible to compare different solutions between each other.

There are two types of corpus depending of the source of the annotations:

- Gold Standard Corpora (GSC): annotations are created manually by expert curators following specific annotation guidelines;
- Silver Standard Corpora (SSC): annotations are created automatically by computerized solutions.

Typically, manually annotated corpora provides more valuable and high-quality information since it is done by expert curators. Due to the expensive cost and time needed to build such corpora, it usually produces a small amount of annotated documents. Automatically generated corpora, on the other hand, provides big amounts of information and large sets of annotated documents, and thus it is highly useful. However, some researchers defend that this type of corpora should not be trusted and used as primary targets for development or evaluation because it still presents a significant number of mistakes. Corpora also vary in terms of granularity, it can be targeted to document abstracts, full text documents or just some selected sentences. Full text documents corpora potentially contains more information than just their abstracts, but also requires more time and resources to be processed [14].

EVALUATION

In order to validate the automatic generated annotations, it is important to define a way to measure the accuracy of the developed solutions. This can be performed by annotating a corpus and then

comparing the automatic generated annotations with the ones provided by expert curators for that corpus (GSC). Each annotation can be classified as:

- True Positive (TP): the annotation exists in the GSC;
- True Negative (TN): the non-identification of an annotation is correct since it is not in the GSC;
- False Positive (FP): the annotation does not exist in the GSC;
- False Negative (FN): an annotation in the GSC is not identified by the automatic tool.

Different types of matching can be used to obtain performance results and understand the behaviour of biomedical IE solutions: exact and approximate matching. Approximate matching provides the performance results when minor and non-informative mistakes are discarded. Performance results are calculated using the fundamental measures: precision, recall and F-measure. These three measures take values between zero and one.

Precision measures the ability of a solution to identify only the relevant annotations.

$$Precision = \frac{\text{Relevant items retrieved}}{\text{Total items retrieved}} = \frac{TP}{TP + FP} \quad (2.1)$$

Recall measures the ability of a solution to identify all relevant items.

$$Recall = \frac{\text{Relevant items retrieved}}{\text{Total items in collection}} = \frac{TP}{TP + FN} \quad (2.2)$$

F-measure is the harmonic between precision and recall.

$$F - \text{measure} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (2.3)$$

Besides these three measures, there are also other relevant measures to evaluate binary classification problems, such as accuracy and sensitivity.

Accuracy measures the ability of a solution to provide correct predictions, both positive and negative.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.4)$$

Specificity measures the ability of a solution to identify negative results.

$$Specificity = \frac{TN}{TN + FP} \quad (2.5)$$

PRE-PROCESSING

Various pre-processing methods, such as Natural Language Processing (NLP) and stopwords removal, can be previously applied to input data in order to simplify the biomedical IE tasks.

Natural language processing

NLP solutions appeared with the goal of studying problems related with the automatic generation and understanding of natural language. Currently, NLP tasks can be effectively performed by computerized systems, although in IE domain it is first necessary to split the documents into meaningful units, such as sentences. NLP solutions will then split each sentence in tokens, which are the basic data processing units. NLP is commonly composed by several linguistic processing tasks, such as tokenization, Part-of-Speech (POS) tagging, lemmatization, chunking and dependency parsing.

Sentence splitting: Splits the text of a document into sentences. There are various solutions to perform this task, such as Lingpipe¹, GENIA SS [15], and OpenNLP².

Tokenization: Splits a sentence into tokens, which are considered the basic data processing units. This is clearly one of the most important tasks of IE, because all other processes will be based on the tokens resulting from this task. Some well known tools to perform this task are GENIA Tagger [16], and SPECIALIST NLP³.

Part-of-Speech: Associates each token with a particular grammatical category based on its context and definition. Thus, this process helps understanding the linguistic role of each token in a sentence. As result of this process a token can, for example, be tagged as a noun (NN), adjective (JJ) or adverb (RB). GENIA Tagger, Lingpipe and OpenNLP also support POS tagging.

Lemmatization: Finds the root form of each word. Applying this process all inflected forms of a word can be group together and processed as a single item. GENIA Tagger and BioLemmatizer [17] are examples of solutions that perform this process.

Chunking: Splits a sentence into groups of tokens that constitute a grammatical unit, such as noun phrase (NP), verb phrase (VP) or preposition phrase (PP). It helps understanding the structure of the sentences. There are various solutions that perform this process, like GENIA Tagger, Lingpipe

¹<http://alias-i.com/lingpipe/>

²<https://opennlp.apache.org/>

³<https://lsg3.nlm.nih.gov/Specialist/Home/index.html>

and OpenNLP.

Dependency parsing: Identifies the relations between each chunk phrases and categorizes them according to their grammatical roles, such as noun modifier (NMOD), verb modifier (VMOD) and preposition modifier (PMOD). While other tasks provide a local analysis of the sentence, dependency parsing allows to understand with more detail how tokens and chunk phrases are related in a sentence, providing a syntactic analysis of the sentence. GDep [18] and Enju [19] are examples of solutions that support biomedical dependency parsing.

Figure 2.2 illustrates the NLP linguistic processing tasks, presenting the output of each task considering the sentence “Down-regulation of interferon regulatory factor 4 gene expression in leukemic cells.”.

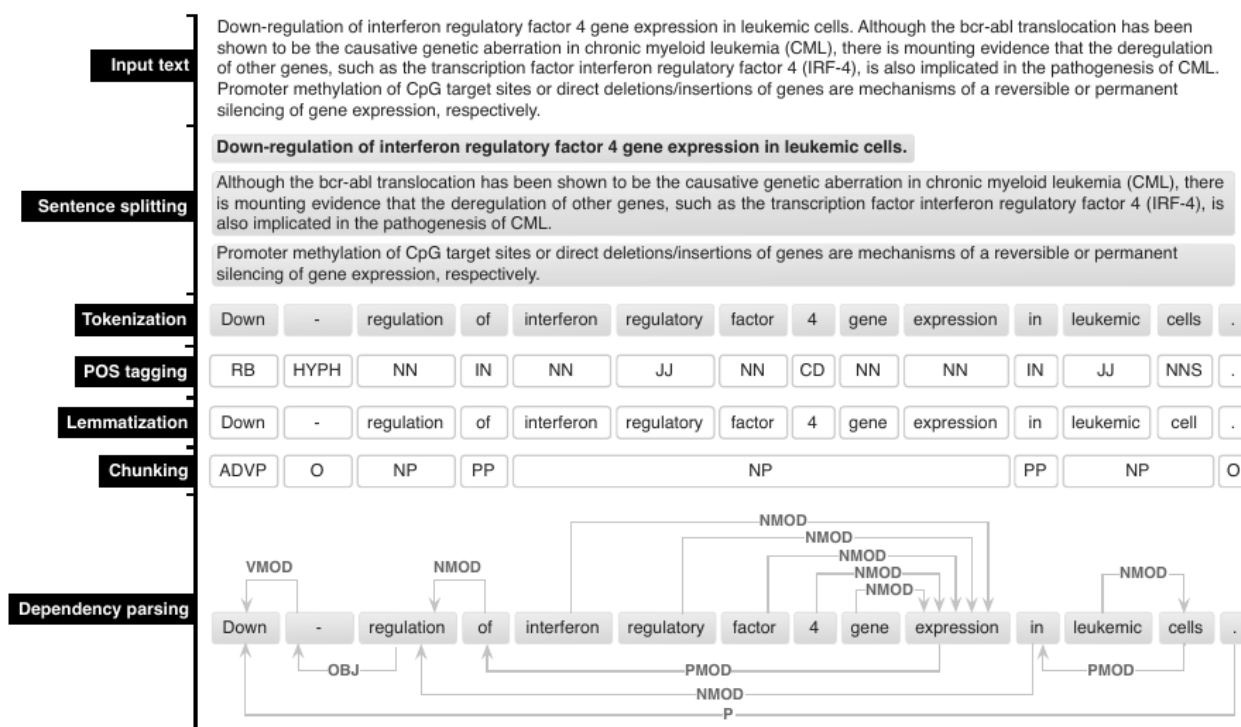


Figure 2.2: Illustration of NLP linguistic processing tasks. (from [20])

Stopword removal

Stopword removal is a common technique in text analysis, and it consists in discarding a set of words that are already known to be non-informative, such as “be” and “which”, consequently leading to the production of a large amount of mistakes. This process contributes to the improvement of performance results and reduces the amount of data to be processed. For example, Pubmed provides a list of specific stopwords for the biomedical area [21].

2.1.2 CONCEPT RECOGNITION

A concept is a biomedical entity that can be found on a curated resource, and it is used to represent and map the current knowledge. As seen in section 2.1.1, biomedical resources are generally databases and ontologies, that contain information about a specific knowledge sub-field, where each concept is identified by a unique identifier. For example, “CD34” is a protein that exists in UniProt database as a concept with the associated unique identifier “P28906” (see figure 2.3). Concept recognition is the process that allows the extraction of concept names and relates them with unique identifiers from biomedical curated resources.

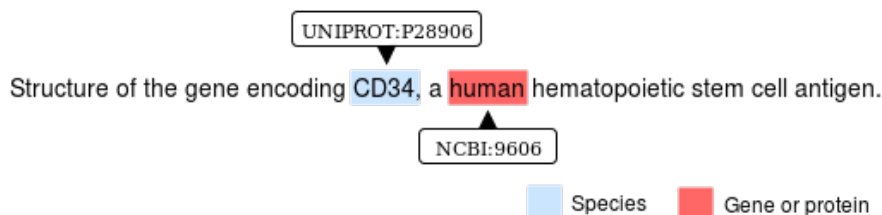


Figure 2.3: Illustration of the biomedical concept recognition task.

Concept recognition is an important step in IE, because all following steps rely on its results to be performed successfully. However, biomedical literature presents a lot of complex challenges, making it difficult to apply such techniques. The main challenges are associated with the terminology, because the terms used in biomedical concepts and processes are very complex [22]. Some of these challenges are: non-standardized naming convention, ambiguous names, abbreviations, descriptive naming convention and nested names.

The biomedical domain contains a large spectrum of knowledge, it is thus important to automatically extract information of the whole spectrum from scientific articles, in order to build rich and reliable information profiles. Due to their implications and inherent interactions and relations, the following can be considered the most important biomedical concepts: species or organisms, genes or proteins, enzymes, mutations, drugs, chemicals, anatomical entities, disorders, pathways, biological processes and molecular functions.

Biomedical concept recognition can be divided in two steps: Named entity recognition (NER), the process of recognizing the concept mentions in text, and normalization, the process responsible for associating the previously recognized names with unique identifiers from knowledge databases. Frequently, disambiguation of concept mentions is necessary in order to achieve correct normalization, since biomedical concepts may share names. A common example is the name of a gene and of an associated disease.

NAMED ENTITY RECOGNITION

The objective of NER is to identify chunks of text and associate them with their specific concept types. Different approaches can be used to perform NER, namely rule-based, dictionary matching and Machine Learning (ML) solutions. Figure 2.4 illustrates the general processing pipeline of NER solutions, which can be decomposed in the following steps and resources:

- Corpus: set of related documents;

- Pre-processing: processes to simplify and allow NER process, for instance NLP;
- NER: automatic recognition of concept names;
- Post-processing: processes to refine the previously recognized concept names, for example abbreviation resolution;
- Annotated corpus: output documents counting the recognized concept.

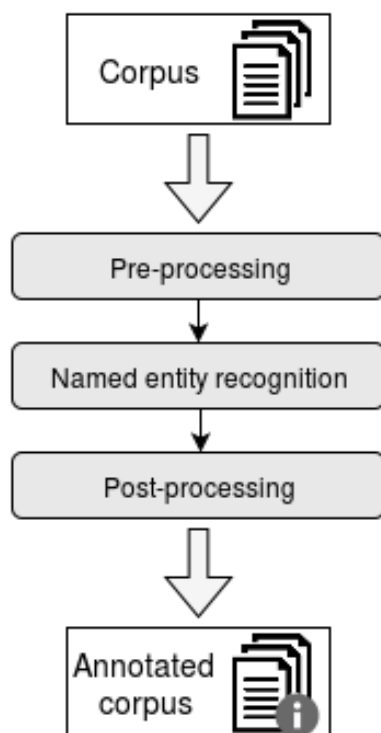


Figure 2.4: General processing pipeline of NER solutions.

Even though different approaches follow a similar processing pipeline, each one of them fulfill different requirements, and thus their performance vary depending on the concept types being identified. Therefore, we should use different approaches taking into account the requirements of each concept type:

- Rule-based: concept names with a strong orthographic and morphological structure. For example, genes;
- Dictionary matching: closely defined vocabulary of names. For example, species;
- Machine Learning based: large variability and dynamic vocabulary of names. For example, chemicals.

Rule-based

Rule-based solutions make use of a set of rules defined by experts, combining orthographic characteristics with word syntactic and semantic properties. The definition of those rules requires a

considerable time and resources, and beyond that the generated rules are too specific, and thus they perform well recognizing concept names on specific corpora, but when applied to others the general performance falls. Therefore, rule-based approaches are only recommended for the recognition of well defined and standardized concepts. PROPER [23] and PASTA [24] are examples of rule-based systems.

Dictionary matching

A dictionary is a large collection of names about a specific concept. This approach tries to recognize the concept names of a specific context by matching the dictionary entries with the document text. The successful matches using this approach are highly related with the unique identifiers from knowledge databases. Although, dictionary based solution presents some limitations, like the production of a large number of first positives due to concepts with short names. This can be minimized by removing short concepts names from the dictionaries, but with the consequence that these names will never be identified as concepts. Another limitation is the nonexistence of all positive spelling variations of a name in the dictionaries.

Machine Learning based

Machine Learning (ML) based solutions use standard techniques to learn how to recognize concept names. These learning processes use annotations from Gold Standard Corpora as knowledge base. By recognizing new spelling variations of concept names, the ML approaches solve one of the limitations of dictionary based solutions. In this approach it is firstly necessary to train a computational model to induce the characteristics of concept names, and then the model is ready to predict what chunks of text may be identified as concepts. One disadvantage of ML based solutions is that they do not provide unique identifiers from knowledge resources to normalize the recognized concept names. However, this problem can be solved by using a set of dictionaries in a following step, linking the recognized names with the entries of those dictionaries.

NORMALIZATION AND DISAMBIGUATION

The objective of normalization and disambiguation task is to associate the recognized concept names with unique identifiers from knowledge databases.

The methods used in this task are similar to the ones applied on dictionary-based approaches. In this case the matches are made between the chunks of text previously recognized as concept names and the dictionaries' entries [25]. The normalization process starts by verifying if a recognized name matches any name on biomedical resources. If no match is found, then the concept may be discarded. On the other hand, if there is only one identifier associated with the concept mention, then that identifier is assigned to that mention. Finally, if there is more than one identifier associated with a mention, then the concept is considered ambiguous. The second step of this task is disambiguation, which goal is to resolve ambiguous names to the correct concepts. If this step is performed with success, then it increases the number of concept names normalized correctly. However, advanced disambiguation techniques are not trivial due to the complexity of the biomedical domain, and requires a large amount of curated data in order to achieve good results.

2.1.3 BIOMEDICAL CONCEPT RECOGNITION SYSTEMS

Various biomedical concept recognition systems exist [26], in the following pages are introduced some of the most popular ones.

NEJI

Neji [27] is an open source framework for biomedical concept recognition that was built around four key characteristics: modularity, scalability, speed and usability. Regarding modularity, each processing task of Neji is performed by an independent module, so each module has its own input and output specifications. Concerning scalability, Neji is able to support various dictionaries and ML models for concept recognition, and process large datasets at the same time.

The core component of Neji is the processing pipeline, which is illustrated the figure 2.5.

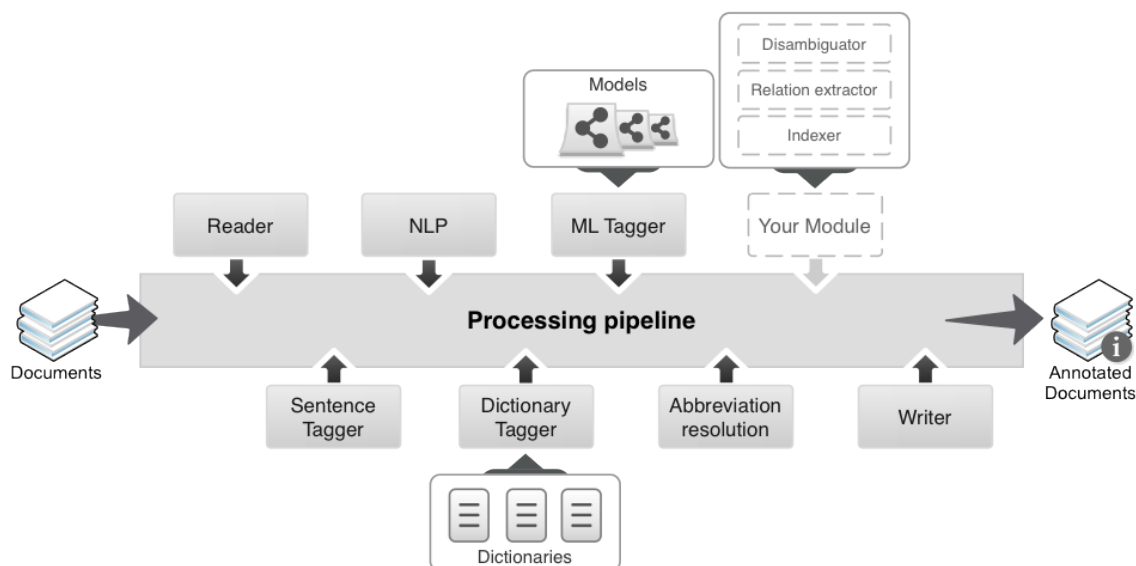


Figure 2.5: Neji processing pipeline. (from [27])

The pipeline allows users to submit various modules that will be executed following a First In First Out (FIFO) strategy. Common modules in a processing pipeline are:

Reader: Neji supports two input formats, raw text and eXtensible Markup Language (XML). XML format allows the user to specify which tags should be processed, while in raw format all text is processed.

NLP: For NLP Neji uses GDep or OpenNLP, dependency parsers, which perform tokenization, lemmatization, POS tagging, chunking and dependency parsing.

Dictionary matching: Performs case-insensitive exact matching in order to recognize biomedical concepts. Note that common English terms from a specified list of stopwords, and terms with less than three characters are ignored during the matching process.

Machine Learning: Neji also supports ML based solutions to recognize biomedical entity types [28]. A simple and general normalization algorithm based on prioritized dictionaries is used to establish a relation between the recognized entity mentions and the unique identifiers of the used dictionaries;

Abbreviation resolution: It integrates abbreviation resolution, identifying the abbreviations and their full forms. If either the large or short form is recognized as a concept, then the other form is also added as a concept, with the same identifier.

Writer: Neji supports several formats used in biomedical domain, such as IeXML, A1 (brat standoff format), CoNLL and JavaScript Object Notation (JSON).

Neji achieved high performance results on name entity recognition and entity normalization when evaluated against three gold standard corpora that contain heterogeneous biomedical concepts [27]. Furthermore, it provides fast and multi-threaded processing.

METAMAP

MetaMap [29] is a highly configurable program developed to map biomedical text to the Unified Medical Language System (UMLS) Metathesaurus.

Using approaches based on NLP and computational linguistic techniques it provides annotation of heterogeneous concepts, using the UMLS Methathesaurus and a set of rules to score text chunks as candidates for concept names.

This tool does not allow the use of dictionary matching or machine-learning based solutions, which are techniques that produce significantly better results than rule-based approaches.

NCBO ANNOTATOR

The NCBO Annotator [30] is a web service that annotates text with ontology terms from public datasets (almost two hundred ontologies from BioPortal [31] and UMLS⁴).

The annotation workflow is composed by two steps (see figure 2.6):

1. **Concept recognition:** The text is parsed by a concept recognition tool, Mgrep⁵ (developed by National Center for Biotechnology Information (NCBI)), using dictionaries of 191 ontologies from BioPortal and UMLS, producing the annotations;
2. **Semantic expansion:** The previous produced annotations are then expanded by expansion algorithms, such as mapping between ontologies, semantic similarity algorithms and `is_a` relations.

⁴https://www.nlm.nih.gov/research/umls/knowledge_sources/metathesaurus/

⁵<https://sourceforge.net/projects/multiline-grep/>

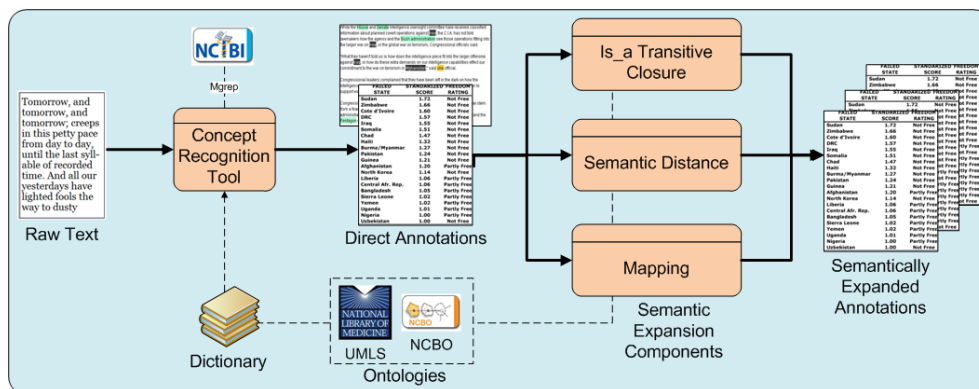


Figure 2.6: NCBO Annotator workflow. (from [30])

Even though it provides a fast and accurate annotation, and allows the use of dictionaries, it does not support ML based solutions, which can improve even more the results. Figure 2.7⁶ illustrates an example of annotating with NCBO Annotator.

Annotator
Get annotations for biomedical text with concepts from the ontologies ?

insert sample text

Melanoma is a malignant tumor of melanocytes which are found predominantly in skin but also in the bowel and the eye.

Select Ontologies
CCONT x PR x clear selection select from list

Select UMLS Semantic Types
Type here to select UMLS semantic types

☐ Match Longest Only ☒ Include Mappings
☒ Exclude Numbers ☐ Match Partial Words
☒ Exclude Synonyms

Include Ancestors Up To Level: None

Get Annotations

Annotations

CLASS	filter	ONTOLOGY	filter	TYPE	filter	CONTEXT
melanoma		Cell Culture Ontology		direct		Melanoma is a malignant ...
skin		Cell Culture Ontology		direct		... predominantly in <u>skin</u> but also in ...
eye		Cell Culture Ontology		direct		... and the <u>eye</u> .

Figure 2.7: NCBO Annotator annotation example.

CONCEPTMAPPER

ConceptMapper [32] is a highly configurable dictionary lookup tool, implemented as an Unstructured Information Management Architecture (UIMA)⁷ component. It maps the input text into dictionary entries using various matching algorithms, producing annotations.

⁶<http://biportal.bioontology.org/annotator/>

⁷<https://uima.apache.org/>

As mentioned ConceptMapper is highly configurable, and therefore all its functionality aspects can be configured:

- Dictionary processing;
- Input documents processing;
- The lookup strategy.

Dictionaries: The ConceptMapper dictionary is represented in XML and presents a very flexible structure. Each dictionary entry is specified by a <token> tag, and it contains one or more synonyms/variants (each one specified by a <variant> tag). Each entry can have as many attributes (features) as needed, their variants inherit all their features and can override them or also have other features.

For dictionary processing, the complete dictionary is loaded into memory, providing very fast lookups.

Input documents: Processing of input documents is token-based and its application is limited to a specific context, generally within a sentence or noun phrase. The input token processing can be configured in terms of case sensitive or insensitive matching, the use of stop words, stemming, abbreviation expansions and spelling variants, use a feature of a token instead of its text, and it is also possible to skip tokens based on its features.

Lookup strategies: The dictionary lookup algorithm is controlled by three parameters: token-order independent lookup, find only the longest match or all possible matches and search strategy. There are three search strategies: the first and default strategy just considers contiguous text tokens, the second can ignore non-matching tokens (proximity searches) and the third is similar to the second strategy, but the next searches starts one token after the beginning of the last match, rather of after the last match, allowing overlapped matches.

WHATIZIT

Whatizit [33] is a text processing web-based system that allows you to perform concept recognition tasks through a set of public web services. It identifies concepts and links them to the corresponding entries in public available bioinformatics databases.

It can also be seen as a suite of modules because it allows users to choose from a set of different pipelines or modules according to the type of entities of interest: chemicals, diseases, proteins, genes, species and others.

The annotation is based on pattern matching, and because of that in some cases some matches are identified, but are not linked to any database entry because they can have more than one meaning, i.e., they are ambiguous.

Figure 2.8⁸ presents an example of annotating with Whatizit graphical interface.

Whatizit

1 Place your text here:

In Duchenne muscular dystrophy (DMD), the infiltration of skeletal muscle by immune cells aggravates disease, yet the precise mechanisms behind these inflammatory responses remain poorly understood. Chemotactic cytokines, or chemokines, are considered essential recruiters of inflammatory cells to the tissues.

2 The text is: **3** Select a pipeline:

Plain Text whatizitUkPmcAll Tag text

Resulting tagged text

In **Duchenne muscular dystrophy (DMD)**, the infiltration of skeletal muscle by **immune cells** aggravates disease, yet the precise mechanisms behind these **inflammatory responses** remain poorly understood . Chemotactic cytokines, or chemokines, are considered essential recruiters of **inflammatory cells** to the tissues .

Figure 2.8: Whatizit annotation example using its graphical interface.

SYSTEMS COMPARISON

Table 2.1 presents a comparison between the features of the addressed biomedical concept recognition systems.

Table 2.1: Biomedical concept recognition systems comparison.

Tool	NLP	Dictionary matching	ML	Normalization
Neji	•	•	•	•
MetaMap	•			•
NCBO Annotator	•	•		•
ConceptMapper	•	•		•
Whatizit	•	•		•

All these systems apply NLP and linguistic techniques to the input text in order to divide it in sentences and tokens, and to normalize the identified concepts.

To perform concept recognition almost all tools use dictionary matching. MetaMap provides annotation based on rules a approach. Some of the systems, Neji and ConceptMapper, also allow a user to add his own dictionaries. Only Neji provides annotation using ML models.

⁸<http://www.ebi.ac.uk/webservices/whatizit/info.jsf>

2.2 INTERACTIVE BIOMEDICAL CURATION

As seen in the previous section a lot of efforts have been made in order to develop tools to perform automatic annotation of concepts. However, sometimes these tools and libraries are complex and their integration and installation is not easy for scientists and researchers, and because of that they are not extensively used by biomedical research communities. Moreover, although these tools can achieve a high performance results in some tasks, they still produce many mistakes and thus the results can still be significantly improved by manual curation. Taking in account these problems have been developed various web-based interactive systems to assist biocurators in the curation tasks. These systems do not require any installation and simplify the use of annotation tools through interactive and intuitive interfaces. Most of them provide both automatic and manual annotation in order to assist biocurators and achieve better results.

2.2.1 WEB-BASED INTERACTIVE CURATING PLATFORMS

EGAS

Egas [34] is a web-based platform for biomedical text mining and collaborative curation. It supports both automatic and manual annotation of concepts and relations between these concepts.

The automatic annotation can be performed by calling a document annotation service, that produces the annotations, or they can be previously added to the documents using one of the supported input formats.

Egas allows users to correct or remove automatic TM results, manually add new annotations, and export the results to well-known and standard formats.

The interface of Egas was developed focusing on simplicity and usability, providing a very intuitive use of all provided tools, facilitating the interaction and making it as simple as possible. Figure 2.9 presents the interface of Egas.

The typical usage pipeline of Egas allows us to understand better all the tools and features it provides. The pipeline is the following:

- **Import documents:** The users can import their own documents in the supported formats (with raw or previously annotated text), or they can use remote resources to import documents, providing a list of document identifiers or by performing a search on these remote resources;
- **Annotation services:** The documents can be automatically annotated using the provided annotation services;
- **Project administration:** The project administrators can specify concept and relation types according to the task, associate each type to a knowledge base for normalization, upload documents that specify the annotation guidelines, which must be followed by the users, and specify the users that belong to the project;
- **Interactive annotation:** Curators can annotate the available documents of the project, by adding, editing and removing concept and relation annotations. This is made using real-time

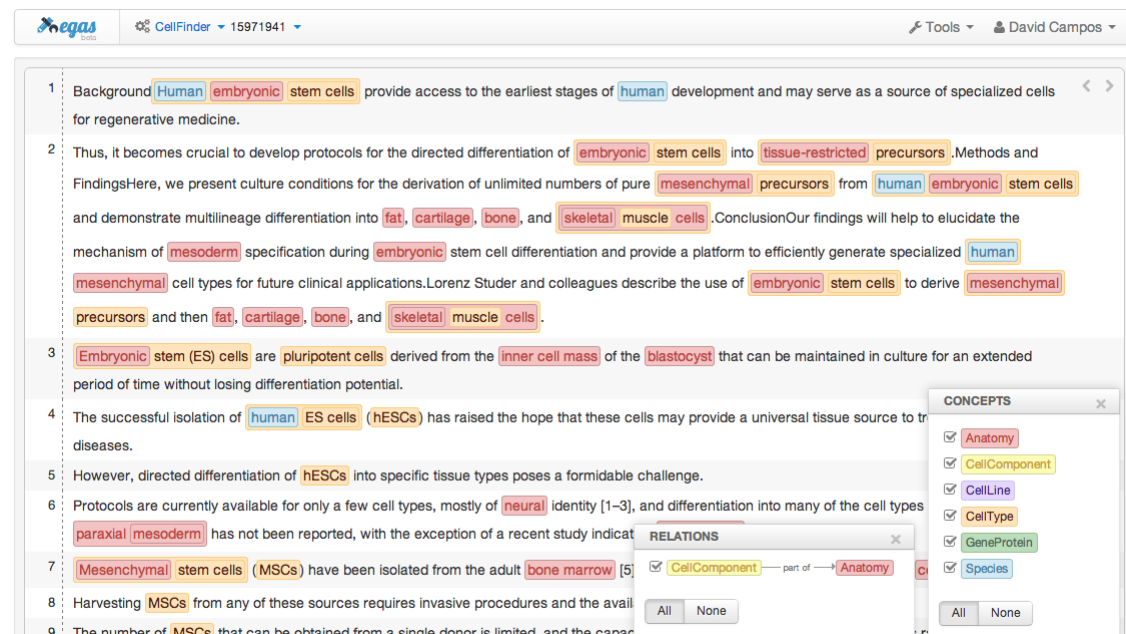


Figure 2.9: Egas entity annotation example. (from [34])

collaboration features that allows a faster and simpler communication between the curators;

- **Export documents:** Finally, the users of the project can export the annotated documents to the supported formats.

To conclude Egas was developed using standard web technologies, such as Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) and JavaScript in order to provide fast documents and annotation processing, and it is supported in the most popular web browsers. The automatic concepts and relations annotation services are provided through RESTful web services.

BRAT

BRAT [35] is a web-based tool for text annotation that allows both manual and automatic concept annotation, and collaborative curation.

The BRAT interface is very simple, user friendly and intuitive, making it a very easy tool to use for inexperienced users.

This tool provides a lot of useful features, such as: inline annotation of documents, concept normalization features, automatic annotation services, dependency parsing, annotation of texts in any language, search capabilities, documents comparison and real-time collaboration. Figure 2.10 presents an example of BRAT platform.

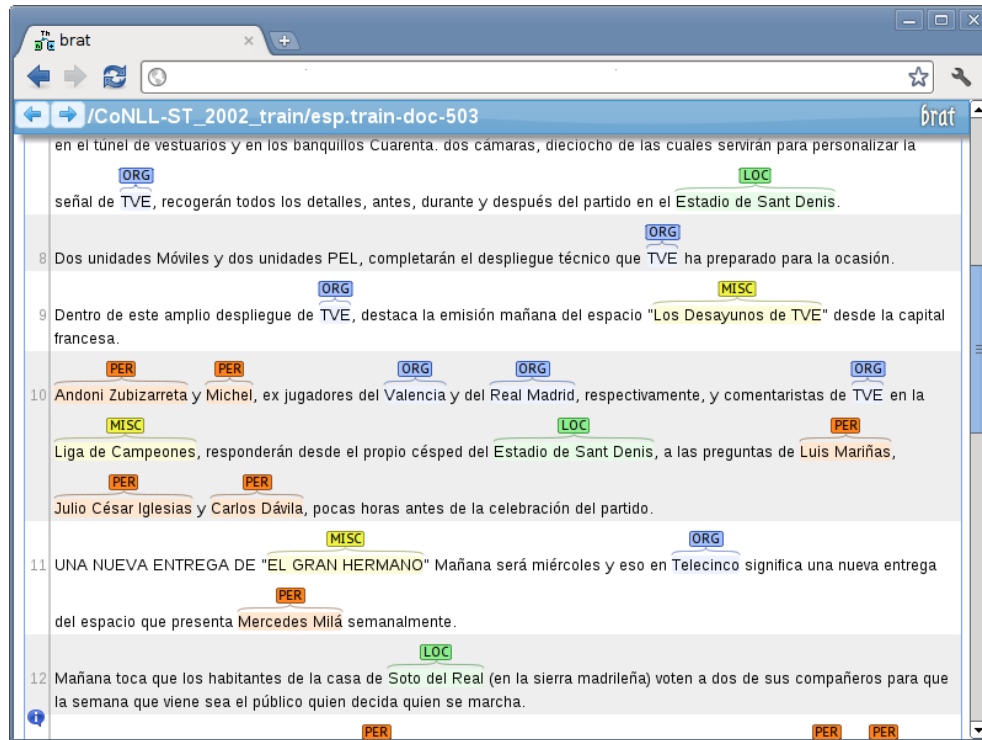


Figure 2.10: BRAT entity annotation example. (from [35])

BRAT also allows annotation task configuration, by defining the target concepts and relations, normalization resources and the automatic annotation services. Although, it seems to be difficult and isn't accessible for normal users.

The evaluations of BRAT show that this tool can decrease significantly the annotation time relatively to another popular tools [35].

MYMINER

MyMiner [36] is an interactive web application for computer-assisted biocuration and text annotation. Like the previous tools it also allows manual and automatic annotation of concepts.

The core of the MyMiner system are four application modules that can be independently used or combined into a processing pipeline:

- **File labelling module:** interface for manual text classification, allowing the classification of documents and the use of labels specified by the user;
- **Entity tagging module:** module for entity mention recognition. Allows the manual detection of entities within a document, which can then be used to populate a knowledge database. It also allows the automatic detection of entities;
- **Entity linking module:** can be used to manually link entities to database identifiers;
- **Compare files module:** allows the direct comparison and evaluation of collections annotated by different approaches, softwares or users.

Figure 2.11⁹ presents an example of entity tagging using MyMiner.

Text to mark

Browse... No file selected. Upload

or
try with this set of articles : Example here

Specify some terms and their associated tags

Protein	DNA	RNA	Cell line	Cell type	Organism	COMP	PROC	FUNC
---------	-----	-----	-----------	-----------	----------	------	------	------

Tag all occurrences Yes ☐ No ☒

Automatically detect entities ☐

Skeletal muscle **myofibrillogenesis** as revealed with a monoclonal **antibody** to **titin** in combination with detection of the alpha- and gamma-isoforms of **actin**.

The distribution of **titin** during **myofibrillogenesis** was examined using **rat** skeletal muscle myogenic cultures and fluorescent-**antibody** staining. Efforts were made to compare the distribution and temporal sequence of incorporation of **titin** relative to that of the alpha- and gamma-isoforms of **actin**. The present observations suggested the following sequence of **titin** assembly: (1) newly synthesized **titin molecules** are distributed in a diffuse pattern throughout the sarcoplasm, (2) the **titin molecules** gradually associate with **alpha- and gamma-actin-positive stress fiber**-like structures (SFLS), (3) groups of **titin molecules** begin to segregate on the SFLS, and (4) **titin molecules** align in a mature doublet configuration in the **sarcomeres** of nascent myofibrils. **titin** assembly on the SFLS often appeared prior to the onset of either alpha- or gamma-actin periodicity on nascent myofibrils; the latter result suggested a role for **titin** in sarcomeric organization. **Actin** distribution on SFLS and its periodicity on nascent myofibrils was usually identical between the alpha- and gamma-isoforms. This suggested that **gamma-actin** participated in **myofibrillogenesis** in a manner indistinguishable from that of **alpha-actin**. The transition seen from continuous actin staining of SFLS to the I-band staining pattern of mature myofibrils is discussed in relation to the corresponding reorganization of **actin** filaments and the molecular associations that this would

Add new label PROC FUNC + Record

See untagged article + See all article +

Please don't forget to record your labelling before changing to another article or exporting your work.

Export all articles Export tagged articles Export untagged articles

Figure 2.11: MyMiner entity tagging example.

The previous presented modules provide a lot features: concept tagging and normalization, automatic concept recognition, document triage and document comparison. However, it doesn't apply inline representation of annotations and because of that the understanding of the inherent information may not be intuitive and clear. MyMiner doesn't provide also collaborative curation.

PUBTATOR

PubTator [37] is a web-based tool that provides assistance to curators, accelerating the manual annotation of concepts and relations using TM techniques. It provides both manual and automatic annotation.

To add documents to PubTator a user can enter a list of PubMed articles or instead search for a term and select the relevant articles to annotate, this articles are retrieved from National Center for Biotechnology Information (NCBI)'s Entrez Programming Utilities Web service. It is also possible to add your own texts, with or without annotations, to annotate.

For the automatic annotation of entities PubTator uses a set of tools for entity recognition and normalization that have presenting good performance results in several text-mining competitions:

⁹http://myminer.armi.monash.edu.au/entity_modified.php

GeneTUKit [38], GenNorm [39], SR4GN [40], Dnorm [41] and tmVar [42]. After the automatic annotation, a user can manually annotate the text by adding, editing or removing any annotation. This tool also allows users to create their own entity and relationship types. After the annotation task a user can export all the annotations. Figure 2.12¹⁰ illustrates an example of PubTator.

PubTator

PMID:26766368 **Trichostatin A enhances estrogen receptor-alpha repression in MCF-7 breast cancer cells under hypoxia.**
 Publication: Biochemical and biophysical research communications; 2016 Jan 6 [Full text links]

TITLE:
 Trichostatin A enhances estrogen receptor-alpha repression in MCF-7 breast cancer cells under hypoxia.

ABSTRACT:
 UNASSIGNED: Estrogen receptor (ER) is a crucial determinant of resistance to endocrine therapy, which may change during the progression of breast cancer. We previously showed that hypoxia induces ESRI gene repression and ERα protein degradation via proteasome-mediated pathway in breast cancer cells. HDAC plays important roles in the regulation of histone and non-histone protein post-translational modification. HDAC inhibitors can induce epigenetic changes and have therapeutic potential for targeting various cancers. Trichostatin A exerts potent antitumor activities against breast cancer cells in vitro and in vivo. In this report, we show that TSA augments ESRI gene repression at the transcriptional level and downregulates ERα protein expression under hypoxic conditions through a proteasome-mediated pathway. TSA-induced estrogen response element-driven reporter activity in the absence of estrogen was synergistically enhanced under hypoxia; however, TSA inhibited cell proliferation under both normoxia and hypoxia. Our data show that the hypoxia-induced repression of ESRI and degradation of ERα are enhanced by concomitant treatment with TSA. These findings expand our understanding of hormone responsiveness in the tumor microenvironment; however, additional in-depth studies are required to elucidate the detailed mechanisms of TSA-induced ERα regulation under hypoxia.

Concept View **Mention View** [Add bio-relation annotation to the table below.](#)

Entity type	Entity mention	Concept ID	Nomenclature	gene_gene	Delete
Disease	breast cancer	D001943	MEDIC	<input type="checkbox"/>	Delete
Disease	cancers tumor	D009389	MEDIC	<input type="checkbox"/>	Delete
Gene	ERα ESRI estrogen receptor-alpha	2099	NCBI Gene	<input type="checkbox"/>	Delete
Chemical	estrogen Estrogen	D004967	MESH	<input type="checkbox"/>	Delete
Gene	HDAC	9734	NCBI Gene	<input type="checkbox"/>	Delete
Disease	hypoxia hypoxic	D000890	MEDIC	<input type="checkbox"/>	Delete
Chemical	Trichostatin A	D012585	MESH	<input type="checkbox"/>	Delete
Gene	TSA	7001	NCBI Gene	<input type="checkbox"/>	Delete

Relation name Relation type Bio-entities Delete

Save Annotation Results Save & Export Annotation Results

Figure 2.12: PubTator annotation example.

The interface is easy to use, intuitive and allows the user to select only the entity types that he wants to see highlighted, although this tool does not allow collaborative curation.

TAGTOG

tagtog [43] is a web-based framework for the annotation of named entities. It provides both manually and automatic annotation to biocurators.

To add documents a user can directly insert the text, upload a file or retrieve an article by its PubMed Unique Identifier (PMID) or PubMed Central reference number (PMCID).

The automatic annotation is based on a ML solution, a named entity recognizer implemented with Conditional Random Field (CRF), trained using a single backward model. In the beginning the model is trained with a small set of annotated documents.

¹⁰<http://www.ncbi.nlm.nih.gov/CBBresearch/Lu/Demo/PubTator/>

After the automatic annotation the user can perform manually annotation of entities and relations by adding, editing or removing any of them. With the continuous user feedback the ML model will be retrained, leading to an improvement in automatic prediction.

The entity normalization can be made linking the annotated concepts to unique identifiers from public standard biomedical databases or the user can also upload its own dictionaries. Figure 2.13¹¹ presents an example of tagtog.

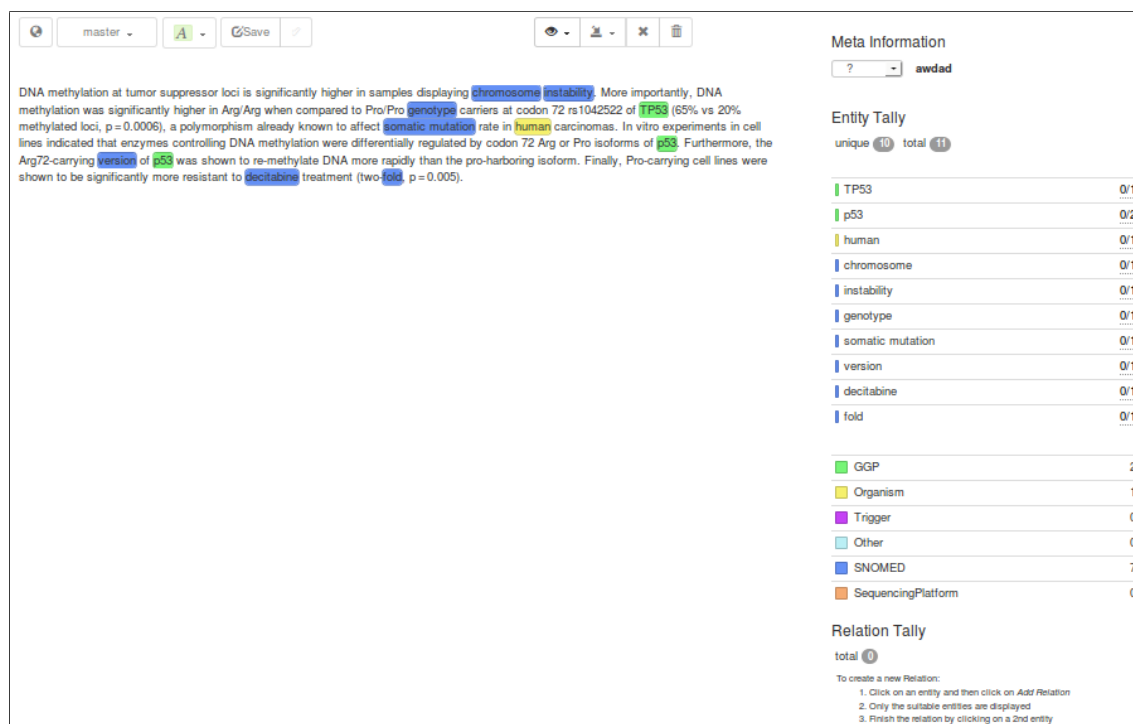


Figure 2.13: tagtog annotation example.

tagtog presents a very simple and intuitive interface, where is possible to select the entity types that user wants to annotate.

PLATFORMS COMPARISON

Table 2.2 presents a comparison between the addressed curating platforms.

Table 2.2: Performance extraction tools comparison.

Tool	Automatic	Normalization	Inline annotation	Collaborative
Egas	•	•	•	•
BRAT	•	•	•	•
MyMiner	•	•		
PubTator	•	•		
tagtog	•	•		•

¹¹<https://www.tagtog.net/>

All these platforms have a common objective, assist biocurators in the curation task, so they try to present a simple, easy and intuitive interface, and provide automatic annotation and normalization of concepts. As seen, only Egas and BRAT provide automatic annotation of relations between concepts. Inline annotation is an important feature that facilitates the representation and identification of relations between concepts, and it is only provided by Egas and BRAT too. Regarding collaborative curation between different users, it is only provided by Egas, BRAT and tagtog platforms.

2.3 PDF FORMAT

The PDF is one of the most popular electronic file formats for publishing documents on the web. It was the first format solving a very important problem, that is to be shown exactly the same everywhere, independently of the operating system, hardware or software applications used to access it. Besides that, PDF reading software was free from almost the beginning of PDF. These facts lead to a big and fast adoption of PDF, making it today one of the most popular electronic file formats, and an open standard for electronic document exchange. Due to its popularity, a lot of applications have been developed to view, create, and manipulate PDF documents. This section presents some general background on information extraction, presents the PDF format, and text extraction from PDF.

2.3.1 TEXT EXTRACTION FROM PDF

PDF is a file format used to present documents that contain text, images, multimedia elements, Internet links, and other elements. It encapsulates the description of a fixed-layout flat document, which includes the various elements that it needs to display, such as texts, fonts and images. Besides, not being a document format based just in text, it also includes several features, such as password access protection, digital signatures and code execution.

FILE STRUCTURE

The basic structure of a PDF file is presented in the figure 2.14 [44].

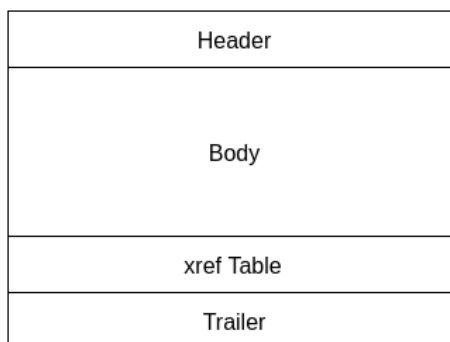


Figure 2.14: PDF file basic structure.

- Header: It is the first line of a PDF file and indicates the version of PDF that the document uses;
- Body: This section contains all the elements of the document that will be shown, like texts, images and multimedia elements;
- xref Table: Cross-reference table, which contains the references for all the elements of the document. It allows random access to the elements of the document, and thus it is not needed to read the whole PDF file to find a particular element. In the xref table each entry represents an element;
- Trailer: It contains a dictionary, an offset to the start of the xref table and the end-of-file marker. The dictionary contains information related to some special elements. PDF reader applications should start reading a PDF file from the end of the file, i.e, this section. The trailer section can specify several keys:
 - Size: number of entries in the cross-reference table (xref table);
 - Prev: offset from the beginning of the file to the previous cross-reference section. It is used only if there are multiple cross-reference sections;
 - Root: reference to the document catalog object, which is the root object of the document tree structure;
 - Encrypt: encryption dictionary of the document;
 - Info: reference to the document's information dictionary;
 - ID: file identifier.

The initial structure of a PDF file can be changed in order to update the information of a PDF document. The additional elements, from the update, are appended to the end of the file. So PDF allows incremental updates, i.e, new elements can be added to the end of a PDF file without the need of rewriting the whole PDF again, and thus the updates can be saved faster. The structure of a PDF file taking in account the incremental updates is presented in the figure 2.15.

Now, besides the sections that were previously addressed, there are also other body, cross-reference and trailer sections that were appended to the PDF file. The new cross-reference sections will just have entries for the new elements, or elements that have been changed or deleted. The deleted objects remain in the file, but are marked with a deletion flag. Finally, each trailer section needs to end with an end-of-file tag, and its Prev entry should point to the previous cross-reference section.

The PDF contains several types of objects:

- Boolean: two possible values, representing true or false;
- Number: represents integer and real values;
- String: series of bytes;
- Name: a slash (/) must be used to introduce a name. It is not part of the name, but indicates that the data after it represents a name;
- Array: ordered collections of PDF objects that can have different types;

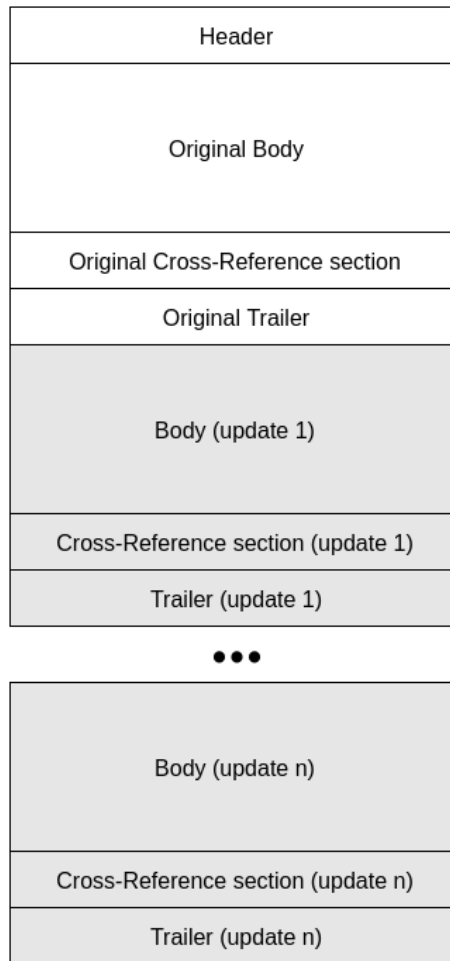


Figure 2.15: PDF file structure.

- Dictionary: table of key/value pairs. A key is a Name object, and the value can be any PDF object;
- Stream: contain large amounts of data, and can be used to represent big data blocks, such as images. It can be compressed and binary;
- Null object: represents the null object.

Additionally, objects can be labelled allowing them to be referenced by other objects, and in this case, those objects are also called indirect objects.

DOCUMENT STRUCTURE

As mentioned previously, a PDF document consists in a set of objects, which are included in the body section of the PDF file. In a PDF, even each page is a dictionary object, that contains the references to the objects that make part of it. All the objects belonging to a page are connected to each other and define a page tree, which is referenced using an indirect object in the document catalog.

The figure 2.16 illustrates the complete structure of a PDF document. The document catalog is the

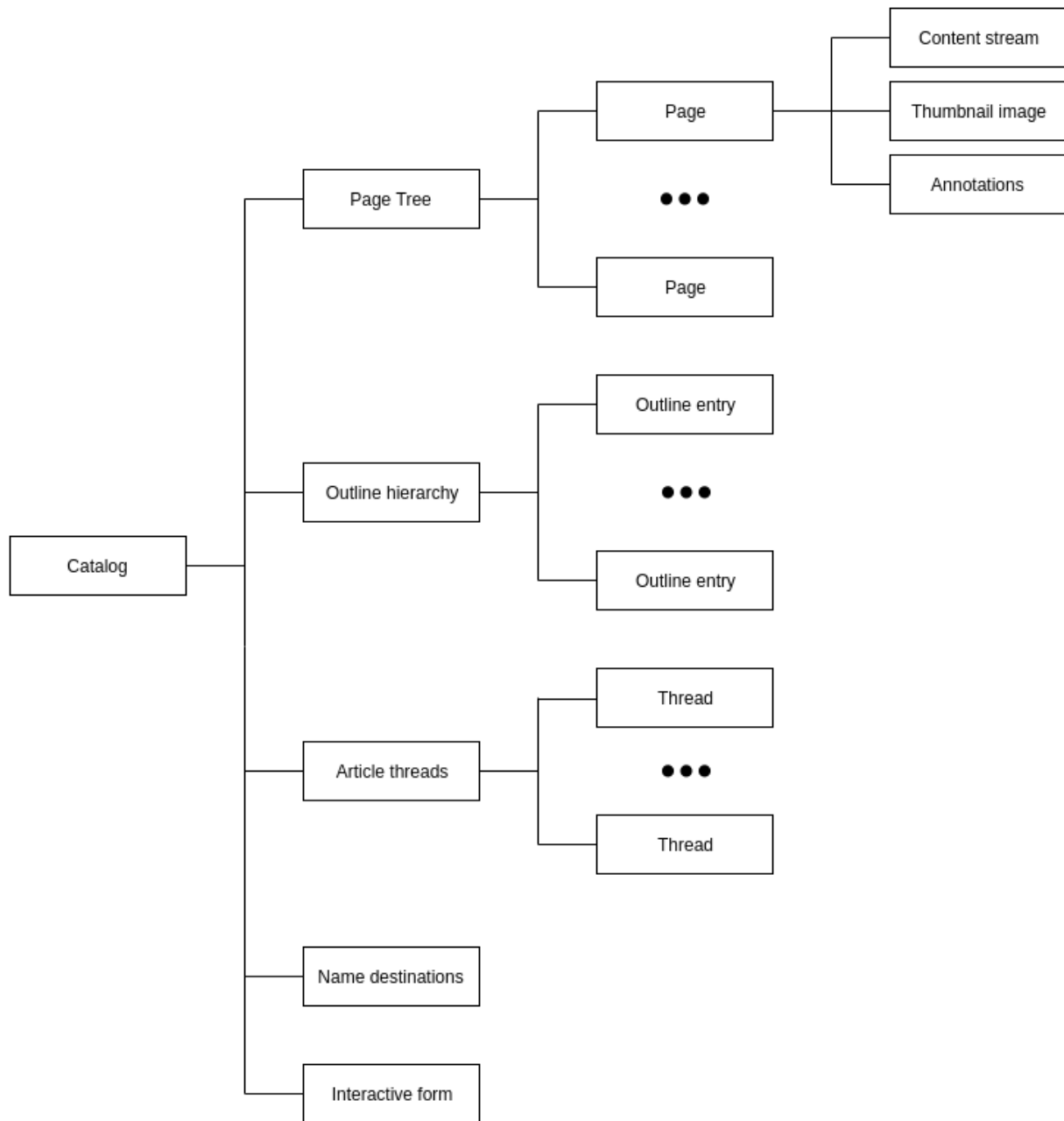


Figure 2.16: PDF document structure.

root of the tree, i.e, the root of the objects belonging to the PDF document, and includes information that declares how the document should be displayed, and references to other objects that specifies the document's catalog: page tree, outline hierarchy, article threads, name destinations and interactive form.

The most important element in PDF document structure is the page tree, which contains the indirect references to all the PDF page objects, and for each page the references for all the objects that make part of it. The nodes of the tree can be of two types: intermediate nodes, which are the page nodes; or leaf nodes, which are the pages objects. Each node in a page tree has the following entries:

- Type: the type of this object;

- Parent: indicates the parent object. This entry is not present in root node;
- Kids: indicates all child objects of the node. This entry is not present in leaf nodes;
- Count: indicates the number of leaf nodes of the current node.

VERSIONS

The first version of PDF is PDF 1.0, which was released in 1993, and since then till today, it has evolved a lot with the improvement and addition of new features. Above is presented a list that addresses the major releases of PDF and some of its features:

- PDF 1.0: The first version, released in 1993;
- PDF 1.1: Released in 1994, this version introduced external links, article threads, protection by password, notes and device independent color (PDF 1.0 only supported RGB);
- PDF 1.2: Released in 1996, this version introduced forms, unicode, multimedia features and improvement of the color support (addition of CMKY color space);
- PDF 1.3: Released in 1999, this version introduced more color spaces, smooth shading, annotations, digital signatures, JavaScript actions and RC4 encryption;
- PDF 1.4: Released in 2001, this version introduced transparency, improved support for JavaScript, JBIG2 compression, 128-bit RC4 encryption and support for Tagged PDF. A Tagged PDF contains structural information about the data that is being displayed, i.e, the metadata of objects can be included in a PDF document;
- PDF 1.5: Released in 2003, this version introduced support for layers, XML Forms Architecture (XFA), improved support for Tagged PDF and compression techniques.
- PDF 1.6: Release in 2005, this version introduced AES encryption, improvement of annotation and tagging, support for embedding 3D data and files into a PDF document, and XML forms;
- PDF 1.7: Released in 2006, this is the current version and introduced the support for comments, improvement of security and allow the embedding of information about printer settings, like paper selection and number of copies.

2.3.2 INFORMATION EXTRACTION CHALLENGES

Once a document is created in PDF format, it can be seen as a read-only document, since each PDF page is perceived as a simply image, which presents a big challenge if someone needs to be able to use, edit or manipulate the data of a PDF document. Besides that, PDF files do not include structure information like paragraphs, columns, layout and tables, but only the positions of the texts and graphics, which difficulties the ordered extraction of information from PDF documents.

Another challenge is the fact of PDF has evolved over the time, so there are several versions and types of PDFs, and thus do not exist a defined standard or fixed structure for PDF files, which requires

different ways to work with them. There are two types of PDFs: native and scanned. A native PDF is created from a document that was digitally processed, while a scanned PDF is created by scanning a physical document using an appropriate device.

This facts represents big challenges for the extraction of information from PDF documents, and therefore a lot of tools have been developed in order to correctly extract information from PDF files, taking in account the layout of the pages and the order in which the information is displayed.

2.3.3 TOOLS

In the following pages are presented some tools that allow the text extraction from PDF documents.

APACHE PDFBOX

Apache PDFBox library [45] is an open-source tool written in Java to work with PDF files. It allows the creation, manipulation and extraction of text and metadata from PDF files. It was started in 2002 in SourceForge, with the objective of enabling the extraction of text from PDF files for Lucene. In 2008 it becomes an Apache Incubator project, and in the following year, 2009, an Apache top level project.

The structure of PDFBox is divided in 3 components:

- FontBox: handle the information associated with the text fonts;
- JempBox: handle the Extensible Metadata Platform (XMP) metadata;
- PDFBox: the main component, handles the text extraction.

The three components are typically used together and are all included in the available PDFBox library.

It is an easy to use tool and provides a very complete documentation and a cookbook with several examples that explain how to use the supported features.

The text extraction can be made in a formatted way; to do that PDFBox tries to identify contiguous text blocks and separate them by break lines in the extracted text. However, it is apparent that in a lot of cases the blocks are not correctly identified, being divided in 2 or more blocks of text. For example, sometimes when the parser finds and hyphen (-), which indicates that the remaining part of a word continues in the next line, it tends to assume that this is the end of a text block (it occurs also with other special characters).

It allows the extraction of text from documents with 2 columns and mixed layout. The header and footer in this cases are correctly extracted being placed before and after the two columns respectively.

This tool has some interesting features: metadata extraction, allows extraction per page, legends extraction, and also allows a user/developer to implement his own parser.

Introduction

The innate immune response—the synthesis of antimicrobial peptides and mobilisation of dedicated immune cells—confers broad protection against pathogens to all multicellular organism. *Drosophila* has become a model for studying the role of hematopoietic (blood) cells and the evolution of cellular immunity (reviews by [1,2]). Similar to vertebrates, *Drosophila* hematopoiesis occurs in two waves during development [3]. A first population of hemocytes is specified in the embryo and gives rise to plasmatocytes involved in



Introduction

The innate immune response—the synthesis of antimicrobial peptides and mobilisation of dedicated immune cells—confers a broad protection against pathogens to all multicellular organisms. *Drosophila* has become a model for studying the role of hematopoietic (blood) cells and the evolution of cellular immunity (reviews by [1,2]). Similar to vertebrates, *Drosophila* hematopoiesis occurs in two waves during development [3]. A first population of hemocytes is specified in the embryo and gives rise to plasmatocytes involved in

Figure 2.17: PDFBox extraction example.

APACHE TIKA

Apache Tika [46] is a set of tools that allows the extraction of text and metadata from several file formats, such as PDF, PPT, XLS and many others. One of the main aspects of Tika is that all types of files can be parsed through a single interface, making Tika a very useful tool for search engine indexing and content analysis.

Currently, Tika is an Apache Software Foundation project, and previously it was an Apache Lucene sub-project.

As mentioned above Tika supports several document formats. To parse PDF documents this tool uses the PDFBox library, that has already been described in the previous section.

Tika allows you to perform the extraction in plain text or eXtensible Hypertext Markup Language (XHTML) format. Using the extraction in plain text the resultant extraction is similar to the obtained with PDFBox, while in XHTML format we get the text with a simple structure, organized in pages and paragraphs, identified by XHTML tags.

Tika is an easy to use tool and provides a detailed documentation and several examples on the document text extraction in both plain text and XHTML format.

In many cases the paragraphs division is not correctly made. For instance, frequently, the first line of a paragraph is identified as making part of the previous paragraph, while the remaining lines are identified as another paragraph.

The extraction for 2 columns PDF documents is made correctly and in order.

Despite allowing structured extraction, this tool does not permit to identify the rhetorical categories, like introduction, abstract and results, of its blocks/paragraphs, which could be useful to filter some of the extracted information.

Tika also allows metadata and legends extraction. It does not permit a per page extraction, but it is possible to implement a personalized handler, that could easily treat the pages identification since they are identified in the XHTML format extraction.

Introduction

The innate immune response—the synthesis of antimicrobial peptides and mobilisation of dedicated immune cells—confers broad protection against pathogens to all multicellular organism *Drosophila* has become a model for studying the role of hematopoiet (blood) cells and the evolution of cellular immunity (reviews by [1,2]). Similar to vertebrates, *Drosophila* hematopoiesis occurs in two waves during development [3]. A first population of hemocytes is specified in the embryo and gives rise to plasmatocytes involved in



```
<p>Introduction
</p>
<p>The innate immune response—the synthesis of antimicrobial
</p>
<p>peptides and mobilisation of dedicated immune cells—confers a
</p>
<p>broad protection against pathogens to all multicellular organisms.
</p>
<p>Drosophila has become a model for studying the role of hematopoietic
</p>
<p>(blood) cells and the evolution of cellular immunity (reviews by
</p>
<p>[1,2]). Similar to vertebrates, Drosophila hematopoiesis occurs in two
</p>
<p>waves during development [3]. A first population of hemocytes is
</p>
<p>specified in the embryo and gives rise to plasmatocytes involved in
</p>
```

Figure 2.18: Tika extraction example.

PDFXSTREAM

PDFxStream¹² is a library available in Java to extract data from PDF documents accurately and fast. It doesn't have dependencies so the only requirement is a compliant Java 1.5 or higher Java Virtual Machine (JVM).

It is currently used by several software development organizations to extract information from PDF documents.

PDFxStream is composed by 4 components (see figure 2.19), available through one unified API:

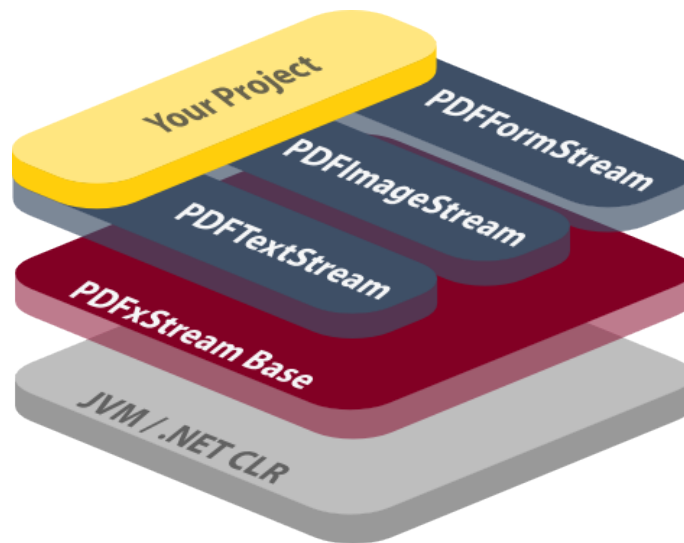


Figure 2.19: PDFxStream architecture. (from [47])

- **PDFxStream Base:** is the base where all the other PDFxStream features are built. This implements the basic capabilities and provides access to the data in PDF documents;
- **PDFTextStream:** provides the PDF text extraction capabilities;
- **PDFImageStream:** provides the PDF image extraction capabilities;
- **PDFFormStream:** provides easy extraction and filling of PDF forms.

¹²<https://www.snowtide.com/>

PDFTextStream claims to be the fastest component available for extracting text from PDF documents, and presents some results obtained through comparison with other extraction tools (see figure 2.20). These results were obtained using a collection of 1000 documents that contains different PDF specifications, languages and character sets. As seen in figure 2.20 PDFTextStream component seems to be the fastest from the three evaluated libraries.

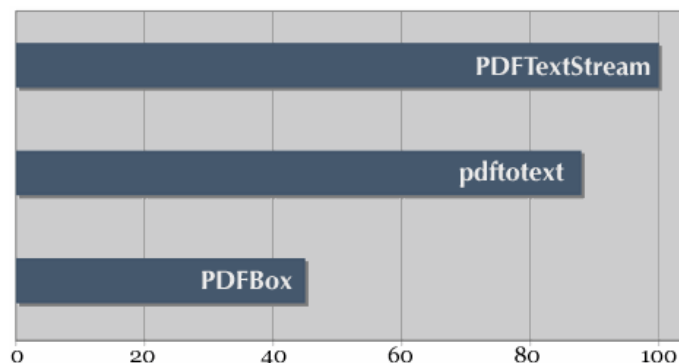


Figure 2.20: PDFTextStream relative performance. (from [47])

The PDFxStream library is very easy to use and offers a complete documentation. The information and examples provided have a good flow to understand what this library can do and how to use it, going from a simple extraction that just provides text, to a more formatted and structured extraction with more useful information.

This tool allows the extraction of text in a structured way. The PDFTextStream document model takes in account pages, blocks, lines and text units (see figure 2.21). Since PDF documents do not indicate the physical structure of the document, PDFTextStream tries to derive the structure applying advanced processes.

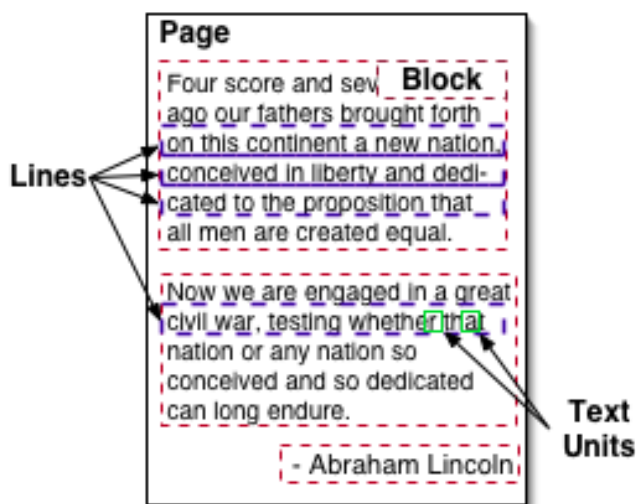


Figure 2.21: PDFTextStream document model. (from [47])

A page contains blocks, blocks may contain blocks or lines (but not both into the same block) and lines contain text units. We can say that a text unit roughly represents a single character. Blocks, lines and text units provide information about their bounds allowing us to retrieve their

positioning on the page (x-coordinate, y-coordinate, height, width, etc.). This is very useful to correctly identify blocks of information.

Using PDFTextStream it is possible to extract only a page or a region of the PDF document (a region can be just a piece of a page).


It allows the extraction of metadata, the common “DocumentInfo” name/value mappings and also Adobe XMP metadata. It is also possible to extract annotations, which can be text notes, styled text attachments, links to the PDF, local or external resources, files and drawings. Annotations contain a lot of information such as the page number where it occurs, their bounds and Uniform Resource Identifier (URI).

The extraction of two column PDF documents is very good, being considerably better than the two previous tools. However, the footer is not correctly extracted, and is usually placed placed between the text of two columns. Using the XML output format we get a structure with the elements page, block and text, that provides some useful information like the bounds of the blocks and the pieces of text in bold and italic. It is also possible to get the position and font-size of each character of the PDF document, and this tool also allows the extraction of images.

One disadvantage is that the free and open-source version can only be used in single-threaded applications.

Introduction

The innate immune response—the synthesis of antimicrobial peptides and mobilisation of dedicated immune cells—confers broad protection against pathogens to all multicellular organism *Drosophila* has become a model for studying the role of hematopoietic (blood) cells and the evolution of cellular immunity (reviews by [1,2]). Similar to vertebrates, *Drosophila* hematopoiesis occurs in two waves during development [3]. A first population of hemocytes is specified in the embryo and gives rise to plasmatocytes involved in



```
<block height="7.1332245" width="60.11633" xpos="58.0535" ypos="153.4677">
  <text>Introduction</text>
</block>
<text></text>
<block height="80.06817" width="239.1967" xpos="58.05367" ypos="59.58239">
  <text> The innate immune response—the synthesis of antimicrobial
  peptides and mobilisation of dedicated immune cells—confers a
  broad protection against pathogens to all multicellular organisms.
  Drosophila has become a model for studying the role of hematopoietic
  (blood) cells and the evolution of cellular immunity (reviews by
  [1,2]). Similar to vertebrates, Drosophila hematopoiesis occurs in two
  waves during development [3]. A first population of hemocytes is
  specified in the embryo and gives rise to plasmatocytes involved in</text>
</block>
```

Figure 2.22: PDFxStream extraction example.

LA-PDFTEXT

LA-PDFText [48] is an open-source tool that extracts text from PDF documents. It was developed specifically for extracting text blocks from full-text research articles, which are then classified into specific regions, like introduction, results, and others, using a set of rules.

This tool has been developed by members of the Biomedical Knowledge Engineering group at the Information Sciences Institute of University of Southern California.

The tool works in three processing steps:

- Step 1 - Detect contiguous text blocks;
- Step 2 - Classify text blocks into rhetorical categories;
- Step 3 - Stitch classified text blocks together in the correct order.

Step 1 - Detect contiguous text blocks: To identify contiguous text blocks, LA-PDFText starts by detecting “work-blocks”, bounding boxes of words, using JPedal¹³, an open-source Java

¹³<https://www.idrsolutions.com/jpedal/>

library to obtain bounding boxes of each word in PDF files. After identifying the “word-blocks”, it tries to aggregate them into “chunk-blocks” taking into account the words proximity and font characteristics. In figure 2.23 is an example of one page from a PDF article and the corresponding blocks identification made by LA-PDFText.



Figure 2.23: LA-PDFText block identification algorithm. Left side shows a PDF page; right side shows the corresponding blocks identification made by LA-PDFText for that page. (from [48])

Step 2 - Classify text blocks into rhetorical categories: In this phase LA-PDFText tries to classify the blocks identified in the last step into rhetorical categories/sections, such as paper title, authors, abstract, introduction, results, methods, discussion, figure legend, references and supporting information, using a rule-based method based on Drools¹⁴, a rule management system.

Step 3 - Stitch classified text blocks together in the correct order: In the last step LA-PDFText iterates over all the classified blocks and stitches them together to produce contiguous sections in the correct order.

The evaluation of LA-PDFText demonstrates that in the first step the block detection algorithm used is fairly accurate and produces very good results identifying the contiguous blocks of text, although the algorithm depends on the accuracy of JPedal identifying word blocks.

In the second step the classification of text blocks presents very good results in general (see [48]). However for the section “Supporting information” the recall is low because most supporting information sections contain figure legends, and the system correctly classifies these blocks as figure legends but not also as supporting information. The references section got a precision and recall lower than the other sections because the font used in tables is the same used in references in many articles, and the

¹⁴<http://www.drools.org/>

base rule-set of LA-PDFText doesn't contain a rule to identify tables so they are being identified as references. Note that the rule-set can be customized by developers in order to improve results for their dataset.

The evaluation made to the third and last step shows that LA-PDFText has the ability to extract text with fewer flow interruptions than the text extracted with the PDF2Text [49] library. It also handles correctly the headers and footers when documents have 2 columns, extracting the text of one page correctly, first the header, then the first column, second column and finally the footer, not introducing any part of the footer between the two columns in the extracted text.

The XML output of LA-PDFText is a structure that contains pages, chunks and words. A document has several pages, a page has chunks (text blocks) and a chunk contains words. For each word we can get the boundary information in the page, font and font-size. The chunks also provide their boundary information and classification.

Introduction

The innate immune response-the synthesis of antimicrobial peptides and mobilisation of dedicated immune cells-confers broad protection against pathogens to all multicellular organism *Drosophila* has become a model for studying the role of hematopoiet (blood) cells and the evolution of cellular immunity (reviews 1 [1,2]). Similar to vertebrates, *Drosophila* hematopoiesis occurs in two waves during development [3]. A first population of hemocytes specified in the embryo and gives rise to plasmatocytes involved in



```
<Chunk x1="58" y1="628" x2="118" y2="638" type="introduction.heading">
  <Word x1="58" y1="628" x2="118" y2="638" font="AdvP41461E" style="font-size:10pt">Introduction</Word>
</Chunk>
<Chunk x1="58" y1="649" x2="297" y2="732" type="introduction.body">
  <Word x1="67" y1="649" x2="79" y2="658" font="AdvP49811" style="font-size:9pt">The</Word>
  <Word x1="87" y1="649" x2="106" y2="658" font="AdvP49811" style="font-size:9pt">innate</Word>
  <Word x1="114" y1="649" x2="141" y2="658" font="AdvP49811" style="font-size:9pt">immune</Word>
  <Word x1="149" y1="649" x2="193" y2="658" font="AdvP49811" style="font-size:9pt">response-the</Word>
  <Word x1="201" y1="649" x2="238" y2="658" font="AdvP49811" style="font-size:9pt">synthesis</Word>
  <Word x1="237" y1="649" x2="242" y2="658" font="AdvP49811" style="font-size:9pt">of</Word>
  <Word x1="249" y1="649" x2="297" y2="658" font="AdvP49811" style="font-size:9pt">antimicrobial</Word>
  <Word x1="58" y1="659" x2="85" y2="668" font="AdvP49811" style="font-size:9pt">peptides</Word>
  <Word x1="92" y1="659" x2="102" y2="668" font="AdvP49811" style="font-size:9pt">and</Word>
  <Word x1="110" y1="659" x2="151" y2="668" font="AdvP49811" style="font-size:9pt">mobilisation</Word>
  <Word x1="158" y1="659" x2="164" y2="668" font="AdvP49811" style="font-size:9pt">of</Word>
  <Word x1="170" y1="659" x2="202" y2="668" font="AdvP49811" style="font-size:9pt">dedicated</Word>
  <Word x1="209" y1="659" x2="237" y2="668" font="AdvP49811" style="font-size:9pt">immune</Word>
</Chunk>
```

Figure 2.24: LA-PDFText extraction example.

PDFTOTEXT

The pdftotext¹⁵ is an open source command-line utility included in many Linux distributions for extracting text from PDF files to a plain text file.

This is a very easy-to-use tool and offers many options to configure the extraction, such as specifying the first and last page to extract, metadata extraction, and many others. Another option (-raw) provides the extraction of the text in content stream order, useful for PDF files with two columns or mixed format, but the use of this mode is no longer recommended.

The base extraction for 2 column format PDF files extracts text correctly, but does not order it correctly in several cases. Although the -raw option is not recommended, when using it in a 2 column PDF it is possible to extract the text in the correct order.

¹⁵<http://linux.die.net/man/1/pdftotext>

Introduction

The innate immune response—the synthesis of antimicrobial peptides and mobilisation of dedicated immune cells—confers broad protection against pathogens to all multicellular organisms. *Drosophila* has become a model for studying the role of hematopoietic (blood) cells and the evolution of cellular immunity (reviews by [1,2]). Similar to vertebrates, *Drosophila* hematopoiesis occurs in two waves during development [3]. A first population of hemocytes is specified in the embryo and gives rise to plasmatocytes involved in



Introduction

The innate immune response—the synthesis of antimicrobial peptides and mobilisation of dedicated immune cells—confers a broad protection against pathogens to all multicellular organisms. *Drosophila* has become a model for studying the role of hematopoietic (blood) cells and the evolution of cellular immunity (reviews by [1,2]). Similar to vertebrates, *Drosophila* hematopoiesis occurs in two waves during development [3]. A first population of hemocytes is specified in the embryo and gives rise to plasmatocytes involved in

Figure 2.25: pdftotext extraction example.

TOOLS COMPARISON

Table 2.3 presents the features comparison of the covered PDF extraction tools.

Table 2.3: PDF extraction tools comparison.

Tool	Layout		Extraction					
	2 col.	Mixed	Plain	Struct.	Class.	Legends	Metadata	Bounds
PDFBox	•	•	•			•	•	
Tika	•	•	•	•		•	•	
PDFxStream	•	•	•	•		•	•	•
LA-PDFText	•	•	•	•	•	•		•
pdftotext	•	•	•			•	•	

All the evaluated tools performed a good extraction for both 2 columns and mixed PDF document layouts, extracting an acceptable quantity of the text content and placing it in the correct order accordingly with the PDF layout. They also performed well in the extraction figure legends.

However, only three of the five tools, Tika, PDFxStream and LA-PDFText allow the extraction of text in a structured format based in XML or XHTML. Tika identifies pages and paragraphs. PDFxStream identifies pages and blocks of text, also indicating the bounds of each text block within a page. LA-PDFText identifies pages, chunks (blocks of text) and words, also providing the bounds information for each chunk and word within a page.

Only LA-PDFText provided text blocks classification into rhetorical categories, such as abstract, introduction, results, discussion, figure legend and others.

Table 2.4 presents a performance comparison of the covered PDF extraction tools, in terms of processing time. This comparison was made using a collection of 1000 PDF documents with different PDF specifications, and several languages and character sets, which was provided by PDFxStream team.

Table 2.4: Performance extraction tools comparison.

Tool	All collection	Average	Processed documents
PDFBox	1.71 min.	102.71 ms	1000
Tika	1.26 min.	60.49 ms	898
PDFxStream	1.96 min.	117.62 ms	998
LA-PDFText	7.77 min.	435.98 ms	739
pdftotext	0.31 min.	18.75 ms	1000

Analysing the results from table 2.4, we can conclude that the fastest tool is pdftotext, but as we previously saw it only extracts text, does not provide information about bounds neither performs classification of blocks. PDFBox, Tika and PDFxStream systems have similar processing times. LA-PDFText tool has a significantly higher processing time than the other tools, but it is the only one that does block classification based on a set of rules, which takes more processing time. The tools Tika, PDFxStream and LA-PDFText have failed when processing some documents of the collection because of varied reasons, such as file encryption and unsupported specifications.

ARCHITECTURE AND IMPLEMENTATION

In this chapter the overall architecture and implementation of the systems that were developed in order to provide the PDF text extraction and the annotation of concepts and relations are presented. Since the processing speed and usability were always strong factors taken into account, the technologies were carefully chosen and are also identified and described in this chapter.

3.1 GENERAL ARCHITECTURE

Neji is a framework for biomedical concept recognition that was extended to allow the processing of PDF documents.

Egas is a web-based platform for biomedical TM and collaborative curation, that was also extended to allow the processing of PDF articles.

To extend Neji a web services platform was developed, allowing its easy integration in Egas and other text mining pipelines. This platform also facilitates the creation and management of annotation services, through an interactive interface.

In the following sections the architecture and implementation of these systems will be explained in more detail.

Figure 3.1 illustrates the final system architecture for processing and annotating PDF documents. It is composed of two main components, that were extended and improved in order to attain the initial objectives.

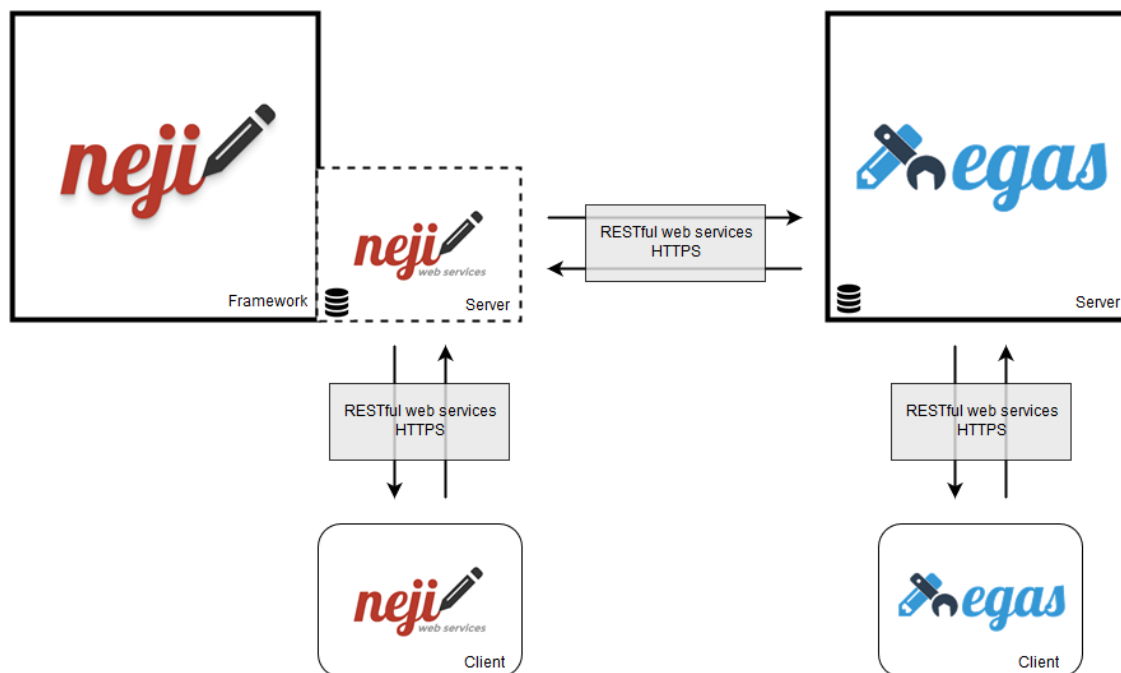


Figure 3.1: General architecture diagram.

3.2 NEJI

Neji [27] is an open source framework for biomedical concept recognition built around four crucial characteristics: modularity, scalability, speed and usability. It follows several state-of-the-art methods for biomedical natural language processing, such as sentence splitting, tokenization, lemmatization, POS, chunking and dependency parsing. The concept recognition tasks are performed using dictionary matching and machine learning techniques with normalization. This framework implements a very flexible and efficient concept tree, where the recognized concepts are stored, which supports nested and intersected concepts with one or more identifiers. It supports several input and output formats including the most popular ones in biomedical text mining, namely IeXML, Pubmed XML, A1, CONLL and BioC. The architecture of Neji allows users to configure the processing of documents according to their specific objectives and goals, providing a very rich and complete concepts information.

It is implemented in Java and uses concurrent processing, reaching very fast processing times, even when processing large datasets (with thousands of documents). Additionally, the libraries and techniques used in the different steps were carefully chosen and tested taking into account the processing flexibility and speed.

3.2.1 PIPELINE AND MODULES

The main component of Neji is the processing pipeline, which can be composed by several modules that will be executed following a FIFO strategy. Therefore, a pipeline is nothing more than a list of independent modules, each of them responsible for a specific processing task, that are executed sequentially. On a pipeline execution, the input documents are the input of the first module, and its output is the input of the second module, and so on. The last module provides the results in a format

specified by the user. Figure 3.2 illustrates the Neji’s processing pipeline and modular architecture:

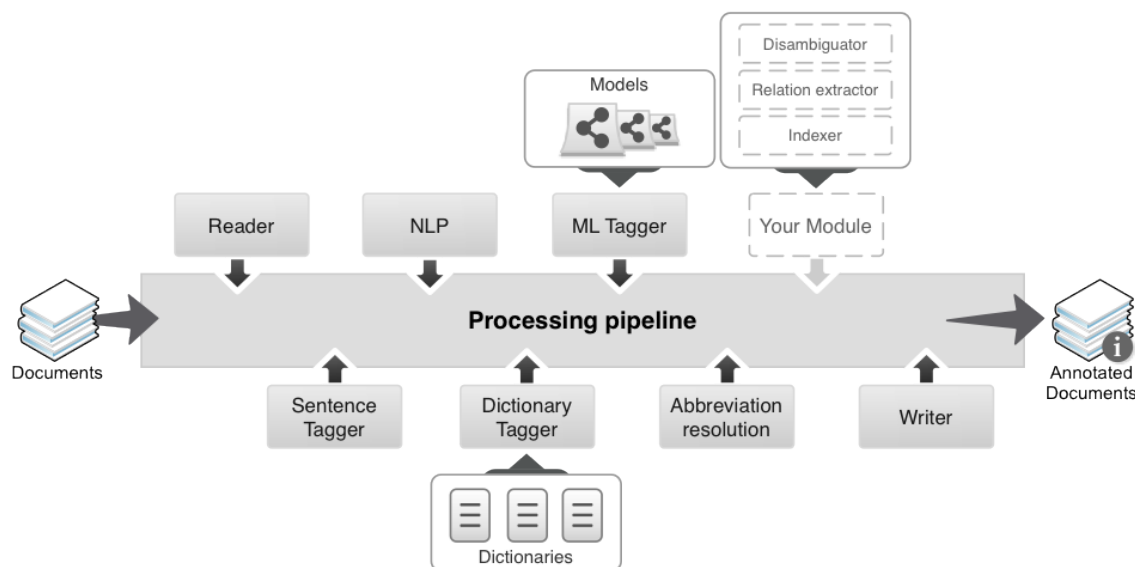


Figure 3.2: Neji processing pipeline and modular architecture. (from [27])

Reader: A reader module is responsible for extracting the relevant data from the input documents, converting it to a format that is readable by the following pipeline modules. Neji provides some different reader modules, one to process raw text, other for XML and another for BioC [50]. While the raw and BioC readers considers all data as relevant, the XML reader allows a user to specify only the tags that contains appropriate data.

NLP: This is usually the second module in the pipeline, and receives the relevant data from the reader module. It starts by performing sentence splitting, which is done using the Lingpipe library, since it contains a sentence splitting model trained using a biomedical corpora, and reached very good results in terms of performance. After sentence splitting, this module performs NLP tasks: tokenization, POS, lemmatization, chunking and dependency parsing. The NLP tasks are implemented using GDep, a dependency parser for biomedical text built on top of the GENIA Tagger. Some adaptations have been made to GDep in order to make NLP tasks more flexible and improve their speed [27].

Concept recognition: Neji uses two different approaches in order to attain a more accurate biomedical concept recognition:

1. **Dictionary tagger:** It compares the entries of one or more dictionaries with the input data trying to find matches. To improve the results of this approach common English words and terms with less than three characters are ignored during the matching process. Dictionary matching is done using a modified version of dk.brics.automaton library¹ which offers an efficient regular expression matching using Deterministic Finite Automatons (DFAs). The processed dictionaries are provided in tab-separated values (TSV) format, which are composed by two fields: an identifier with the format “source:identifier:type:group”, and a list of words that

¹<http://www.brics.dk/automaton/>

matches the identifier separated by pipes (“|”). A list of orthographic variants of the words can also be added to the list of words;(see figure 3.3).

```
UMLS:C0000744:T047:DISO abetalipoproteinemia|acanthocytosis|apolipoprotein b deficiency
UMLS:C0000774:T047:DISO abnormality of secretion of gastrin
UMLS:C0000786:T047:DISO spontaneous abortions|abortion, spontaneous|miscarriages|sponta
UMLS:C0000809:T047:DISO habitual abortion|recurrent abortions|recurrent abortion|recurr
UMLS:C0000814:T047:DISO missed abortions|miss abortion|missed abortion|abortion miss|ab
UMLS:C0000821:T047:DISO threatened abortion|abortions, threatened|abortion threatened|t
UMLS:C0000823:T047:DISO veterinary abortions|veterinary abortion|abortion, veterinary|a
UMLS:C0000833:T047:DISO abscess|abscess nos|abscesses
UMLS:C0000880:T047:DISO acanthamoeba keratitis|acanthamoeba keratitides|keratitides, ac
UMLS:C0000889:T047:DISO keratosis nigricans|acanthosis nigricans
```

Figure 3.3: Neji dictionary example.

2. **Machine Learning tagger:** Neji also supports ML based solutions to recognize biomedical entity types [28]. This uses Conditional Random Fields (CRFs) implementation from MALLET², a machine learning for language toolkit, which presents good results in popular corpora. A simple and general normalization algorithm based on prioritized dictionaries is used to establish a relation between the recognized entity mentions and the unique identifiers of the used dictionaries.

Post-processing: Neji offers the possibility to include post-processing modules with the objective of enhancing the extracted information and increase the recognized concepts. By default, one post-processing module is added:

- **Abbreviation resolution:** This module performs abbreviation resolution. It tries to identify abbreviations and their full forms with the assistance of a abbreviation definition recognizer based on a set of rules [51], and if one of them was recognized as a concept, then the other form is also added as a concept.

Writer: A writer module outputs the biomedical recognized concepts to files or databases. Neji contains a big set of output formats, including the most popular ones in biomedical text mining: IeXML, CoNLL, A1 and JSON.

To support the pipeline sequential infrastructure and execution, Neji uses the hierarchical text processing features of Monq.jfa³, a library for fast and flexible text filtering with regular expressions.

Finally, but not less important, Neji was designed to be easily used and configured by anyone, from developers to researchers. Thus, the pre-defined pipeline (figure 3.2) can be used, or users may implement their own modules, in order to fulfill their requirements, and then build a custom pipeline with those modules.

²<http://mallet.cs.umass.edu/>

³<http://www.pifpafpuf.de/Monq.jfa/>

3.2.2 PDF INFORMATION EXTRACTION

As we have seen, Neji is a powerful framework for biomedical information extraction that offers the possibility to automatically extract dozens of biomedical concepts by applying the most popular and optimized techniques. However, this only allows the extraction of concepts from text based documents, which does not apply to PDF documents.

To allow Neji to receive PDF documents as input arguments, a new reader module was developed specifically for handling this format of files. To create the reader various PDF text extraction tools were carefully tested (see section 2.3.3) to understand which of them would be more appropriate for our needs. From this evaluation was concluded that LA-PDFText library was the most appropriate tool to be used because of its features. It is an open source project, provides a good extraction of text for various page layouts, namely one column, two columns and mixed layout. Beyond the text, it also extracts other data related with the text blocks, such as its positioning on the page, text size and used font. Finally, it is the only tool that offers the possibility to identify the rhetorical category of text blocks, like introduction, abstracts, results, discussion and many others.

The developed reader module processes the PDF files in four steps:

1. Text extraction;
2. Block classification;
3. Block sorting;
4. NLP and mapping between the extracted text and PDF data.

Text extraction

First is extracted the whole text, as text blocks, from the PDF file using a modified version of the LA-PDFText library (see listing 1). This modified version were developed by us and improves the range of PDF types that LA-PDFText can handle, making it an even more general tool. It starts by identifying each word individually in a page, and then aggregates them into blocks of contiguous text, based on the words proximity and font characteristics [48]. Figure 3.4 presents an example of this processing step. Additionally, the text is extracted is encoded in UTF-8 to support the extraction of text from PDF documents in several languages.

```

1  private LapdfDocument readPDFFile(File pdfFile, File rulesFile) {
2
3      LapdfDocument doc;
4
5      try {
6          doc = pdfEngine.blockifyPdfFile(pdfFile);
7
8          if (rulesFile != null) {
9              pdfEngine.classifyDocument(doc, rulesFile);
10         } else {
11             pdfEngine.classifyDocumentWithBaselineRules(doc);
12         }
13     } catch (Exception ex) {
14         throw new RuntimeException("There was a problem parsing the PDF "
15             + "file. Document: "
16             + getPipeline().getCorpus().getIdentifier(), ex);
17     }
18
19     return doc;
20 }

```

Listing 1: PDF text extraction snippet.

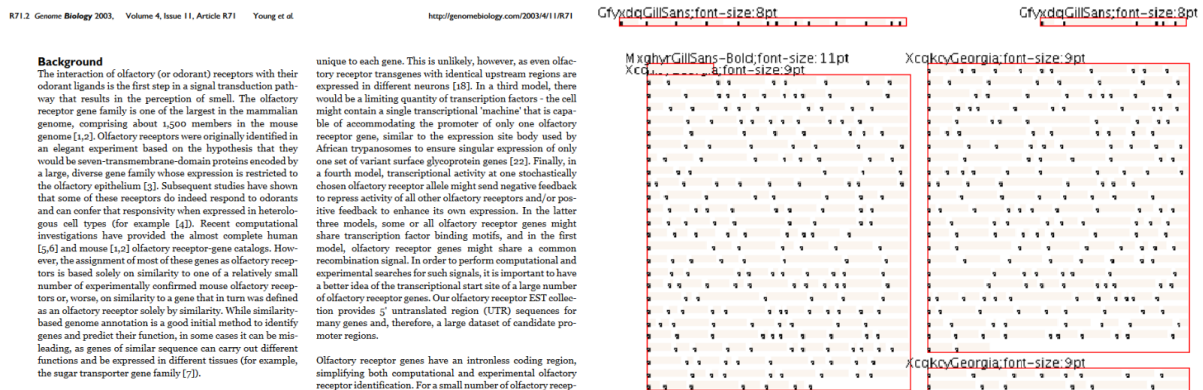


Figure 3.4: Left side shows a PDF page; right side shows the corresponding text blocks identification for that page.

Block classification

To classify the blocks into rhetorical categories, LA-PDFText uses a set of rules based on the properties of the text blocks (placement on the page, page number, font characteristics and others). Neji provides to LA-PDFText a default set of classifying rules, but since different PDF articles have different layouts and structures, different sets of rules will perform better or worst depending on the document where they are applied. Thus a user can define their own set of rules and provide it to Neji to be used instead of the default one.

Block sorting

Different PDF documents present different layouts. The more common ones are single column, two column and mixed layout (both single and two column layouts in the same page). The sorting process is done for each page individually, and the text blocks of each page are sorted taking into account that the page layout can be any of the three already addressed (see listing 2).

```
1    ...
2
3    // Iterate over the PDF file pages
4    for (int i=1 ; i<=doc.getTotalNumberOfPages() ; i++) {
5
6        PageBlock page = doc.getPage(i);
7
8        // Iterate over the page chunks
9        for (ChunkBlock chunk : page.getAllChunkBlocks(
10            SpatialOrdering.PAGE_COLUMN_AWARE_MIXED_MODE)) {
11
12            // Get chunk text and iterator
13            String text = chunkText + "\n";
14            Iterator<SpatialEntity> chunkWordsIt = page
15                .containsByType(chunk, SpatialOrdering.MIXED_MODE,
16                    WordBlock.class).iterator();
17
18            // Build corpus text
19            corpusText.append(text);
20
21    ...
```

Listing 2: Text blocks sorting snippet.

NLP and mapping between the extracted text and PDF data

Usually the NLP tasks are performed in an individual pipeline module, however in this case would be useful to associate some data of the processed PDF documents with the sentences and tokens provided by its tasks. If the NLP tasks were done in other module, all the processing steps addressed bellow would have to be made again in that module causing a big impact in the overall processing time of the pipeline.

Besides the text, LA-PDFText also provides other important data that should be preserved, namely the exact word positions within a page, and the rhetorical category of each text block. These data will be useful later to allow the implementation of some navigation and synchronization features in Egas platform.

The NLP processing tasks are applied to each text block, and for each obtained sentence is associated the page number where it occurs, the rhetorical category of its text block, and its position within the page. The sentence position is composed by the positions of the bounding boxes of the first and last token, and the most left and right positions from all tokens in the sentence. Combining

these data is possible to identify the area of a sentence in a PDF page (see figure 3.5). Instead of preserving this positions for each sentence, could have been saved the positions of all words of the sentence, although each word position is defined by a bounding box, composed by four positions with two coordinates each, what would generate a much bigger amount of data for each sentence, making it unfeasible to be sent through a web service.

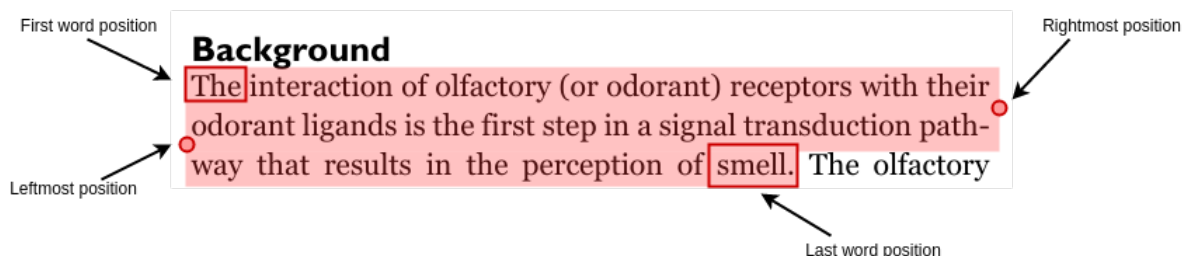


Figure 3.5: Position mapping example for the sentence “The interaction of olfactory (or odorant) receptors with their odorant ligands is the first step in signal transduction pathway that results in the perception of smell.”

After the reader module, and since it already performs the NLP tasks, the sentences are passed to the concept recognition modules, i.e, dictionary tagger and ML tagger modules. In order to identify were the annotations are placed in a PDF page, a process similar to the used with the sentences is applied to the annotations.

To export the obtained results was developed a new writer module because besides the text, sentences and annotations, it is also needed to export the extra information associated with the sentences and annotations. This new writer module is JSON based and is composed by the full document text and an ordered array of the text sentences. Each sentence has the following attributes:

- page: PDF page where it occurs;
- type: rhetorical category. If not identified, it is “unclassified”;
- startX1: upper left corner x position of sentence first word;
- startY1: upper left corner y position of sentence first word;
- startX2: lower right corner x position of sentence first word;
- startY2: lower right corner y position of sentence first word;
- endX1: upper left corner x position of sentence last word;
- endY1: upper left corner y position of sentence last word;
- endX2: lower right corner x position of sentence last word;
- endY2: lower right corner y position of sentence last word;
- leftX: leftmost x position from all sentence words;
- rightX: rightmost x position from all sentence words;
- id: index of the sentence;

- start: start position in the full document text;
- end: end position in the full document text;
- text: sentence text;
- terms: sentence annotations.

Listing 3 presents an example of the output of this writer module.

Note that any of the other writer modules can also be used when processing PDF documents, but they do not include the extra information extracted from the PDF documents.

```

1  {
2      "text": "BMC Genetics\nBioMed Central\nBMC Genetics 12 2001, 2
3          Research article\nIntraocular pressure in genetically
4          distinct mice an update and strain survey\n ...",
5      "sentences": [
6          ...
7          {
8              "page": 1,
9              "type": "title",
10             "startX1": 55,
11             "startY1": 104,
12             "startX2": 139,
13             "startY2": 121,
14             "endX1": 102,
15             "endY1": 122,
16             "endX2": 150,
17             "endY2": 139,
18             "leftX": 55,
19             "rightX": 524,
20             "id": 3,
21             "start": 69,
22             "end": 146,
23             "text": "Intraocular pressure in genetically distinct mice an
24                 update and strain survey",
25             "terms": [
26                 {
27                     "x1": 55,
28                     "y1": 104,
29                     "x2": 207,
30                     "y2": 121,
31                     "ids": "UMLS:C0021888:T042:PROC_FUNC",
32                     "score": 1,
33                     "id": 1,
34                     "start": 69,
35                     "end": 89,
36                     "text": "Intraocular pressure",
37                     "terms": []
38                 },
39                 ...
40             ]
41         },
42         ...
43     ]
44 }

```

Listing 3: Example of the output from the new developed writer.

3.2.3 NEJI WEB SERVICES

To extend Neji was created Neji web services, a web-services platform that intends to facilitate the integration of Neji with Egas and other tools. The provided RESTful API allows developers to send their documents and export the annotation results in various well-known formats in biomedical text mining. It also facilitates the creation and management of annotation services and resources by providing an easy and intuitive web interface. This way a user do not need to download, install or configure Neji in order to use it, he can access it using only his preferred web browser.

The main objective of Neji web services was to develop a way to easily manage concurrent annotation services, allowing the configuration of the properties and resources (dictionaries and ML models) of each of them independently.

ARCHITECTURE

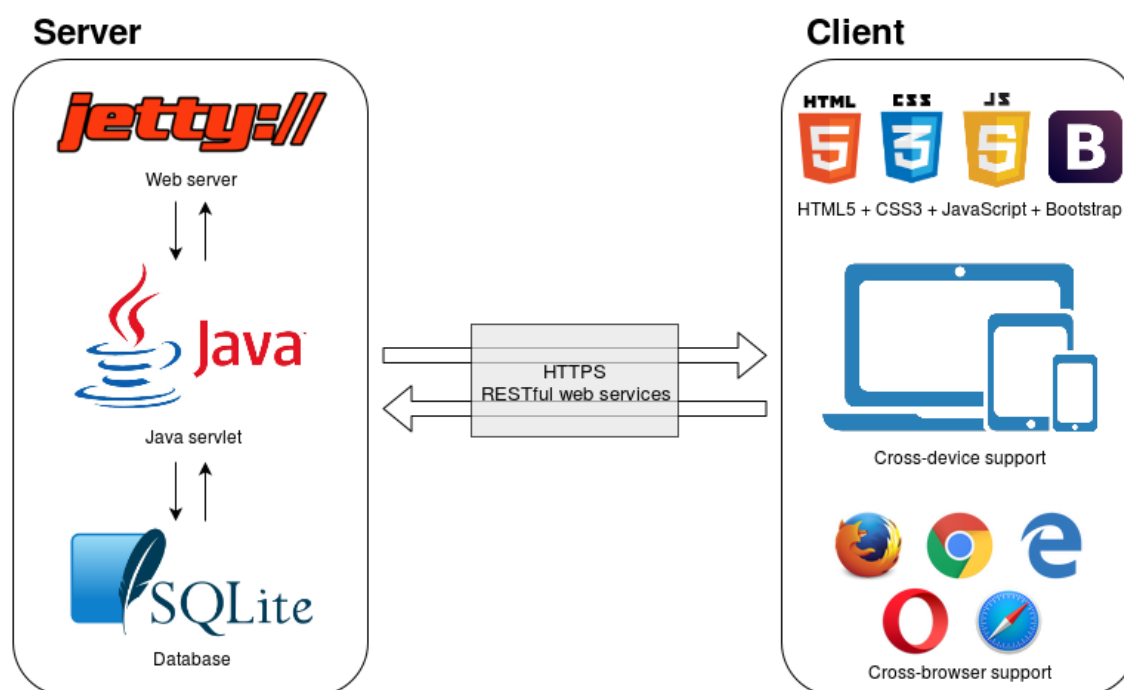


Figure 3.6: Neji web services architecture diagram.

Figure 3.6 illustrates the architecture of Neji web services, which is divided in two main parts: server and client. While the server-side is responsible for processing the data and storing it, the client-side is responsible for the interaction with the users web browsers.

SERVER

The server-side is responsible for storing all data and resources, as well as providing the mechanisms to access and interact with them. It was implemented in Java and uses as base Jetty⁴, which is an open source Java web server and servlet container. Thanks to Jetty features the server can be easily

⁴<http://www.eclipse.org/jetty/>

installed as a standalone application server, facilitating its deployment and distribution. The resources, dictionaries and ML models, are stored in the server file system, while their metadata, services information and configurations are stored in a SQLite⁵ relational database. SQLite is an embedded Structured Query Language (SQL) database engine that instead of have a separate processing server process, like other SQL databases, stores the data directly to a single disk file, which contributes towards making Neji web services a standalone application server. All operations are provided through a RESTful API, allowing an easy access to all features, and enabling their integration in other applications. It was developed using Jersey⁶, an open source framework for developing RESTful web services in Java, and deployed and made publicly available using the Jetty web server. In order to guarantee that the exchanged data is fully protected, the communication between the server and client is done through a secured and encrypted channel using Hypertext Transfer Protocol Secure (HTTPS).

Data structure

As mentioned before all data is centralized in a unique resource, that stores and provides the access to services, dictionaries, models and normalization information. Figure 3.7 presents Neji web services data structure.

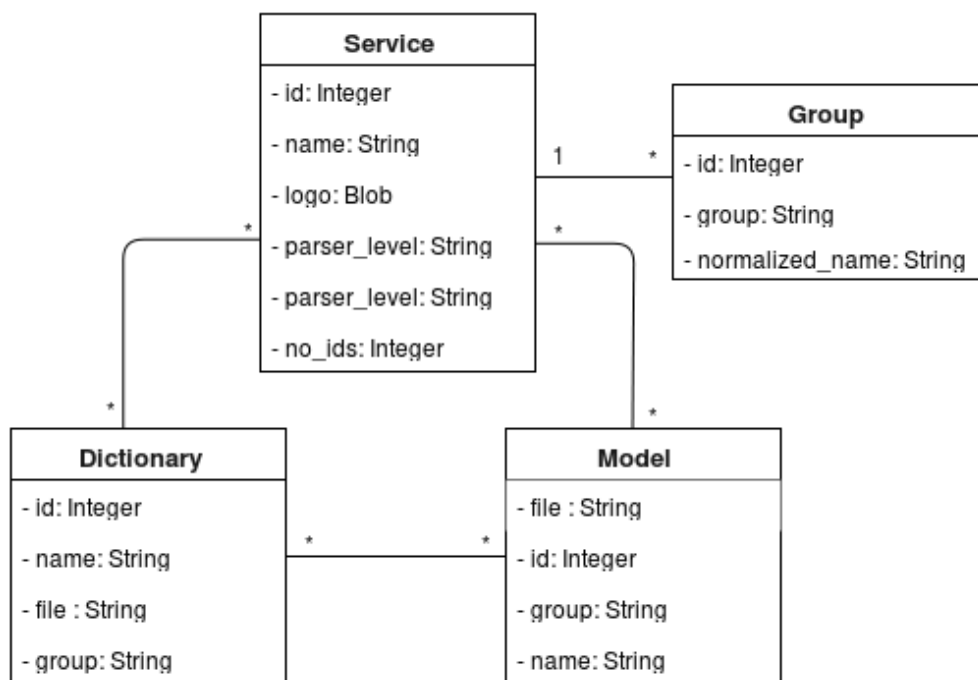


Figure 3.7: Neji web services data structure.

Each service may have multiple resources, dictionaries and ML models. The more resources a service has associated, the more concept names it can identify, since it has a large knowledge set, but the processing time also increases.

ML solutions do not provide unique identifiers to the recognized concept names, so a user can associate multiple dictionaries to models, providing a way to normalize the concept names.

⁵<https://www.sqlite.org/>

⁶<https://jersey.java.net/>

Finally a service may have multiple groups. A group is an entity type, such as PRGE (protein or gene) or CHEM (Chemicals). Sometimes the group entity names are not intuitive to users, so they can be normalized to a designation defined by an user.

Web interface

The additional management and annotation web interface was developed taking into account cross-browser and cross-device support, and thus standard technologies such as HTML5, CSS3 and JavaScript, which are supported by the most popular web browsers on both desktop and mobile devices were used. When developing a web interfaces one important point to take in consideration is its usability. It should be simple, intuitive and easy to use, so users can annotate texts and documents in a fast and uncomplicated way. In order to provide such interface, popular front-end technologies, such as Bootstrap⁷, jQuery⁸ and Handlebars⁹. Additionally, in order to provide a fast interaction to the users the processing in client-side uses simple and fast algorithms.

RESOURCES

Neji web services were built on top of Neji and thus they use the same type of resources to annotate documents. There are two types of annotation resources: dictionaries and ML models.

1. **Dictionaries:** The dictionaries need to be provided in Tab-separated values (TSV) format;
2. **ML models:** In order to add a ML model to the server it is needed to provide the three files that compose the model: the model itself, the properties file, which contains information relative to the model, such as parsing direction and group entities, and the configuration file, which defines what features were used in the training process. After adding a model to the server, a set of server dictionaries can be associated with it in order to allow concept names normalization.

One of the Neji processing tasks that take longer is the loading of resources. Each time Neji is executed it needs first to load all provided dictionaries and ML models into memory and only then it can start to annotate the documents. In Neji web services each time a resource is added to the server it is immediately loaded into memory, and stays loaded till the resource is removed or the server is shutdown. This way when someone annotates a document, the needed resources are always loaded, and thus the annotation is much more faster.

SERVICES

Neji web services is able to provide different annotation services, in which a service is an annotation pipeline with a custom set of resources (dictionaries and ML models) and processing properties. Each service has a set of configurable attributes:

- **Name:** Identifies the service and is used to access it through the REST API (can be seen as the service identifier since there can not be two services with the same name);

⁷<http://getbootstrap.com/>

⁸<https://jquery.com/>

⁹<http://handlebarsjs.com/>

- **Logo:** Image that represents the service and is presented in service annotation page;
- **Dictionaries:** Set of dictionaries to use in annotation (this dictionaries are already loaded in the server);
- **ML models:** Set of ML models to use in annotation (this models are already loaded in the server);
- **Semantic groups mapping:** Sometimes the group entity names are not intuitive, so this attribute is a map that associates each group entity from the service resources to a normalization name chosen by the user.
- **Parsing level:** Parsing level to be used in annotation. It can be tokenization, lemmatization, POS, chunking or dependency;
- **Annotation without identifiers:** Flag to declare if the recognized annotations without an associated identifier should be taken in account.

Services can only be created and edited by the server administrators, but once it is created it can be publicly accessed by anyone through its annotation page or using the REST API.

WEB SERVICES

Neji web services offers a RESTful API that implements two major types of web services. The annotation web services and the export web services.

For the annotation web services were implemented two methods, one to annotate texts, which receives the text, a service name and a set of entity groups, used to select what concepts should be identified, processes the data through a pipeline, and provides the annotation results in a predefined JSON format. And another, specific for annotating PDF documents, which receives a PDF file, a service name and a set of entity groups, and provides the annotation results using the same predefined JSON format (see listing 4).

```

1  @POST
2  @Path("/annotate/{serviceID}")
3  @Consumes(MediaType.MULTIPART_FORM_DATA)
4  public Response annotate(@FormParam("pdf_file") InputStream file,
5                          @FormParam("pdf_file") FormDataContentDisposition fileInfo,
6                          @FormParam("ann_data") String annData,
7                          @PathParam("serviceID") String serviceID) {
8
9      // Get server instantiation
10     Server server = Server.getInstance();
11
12     // Get service
13     service = Server.getInstance().getService(serviceID);
14
15     ...
16
17     // Instantiate pdf request object
18     Gson gson = new Gson();
19     PdfRequest request = gson.fromJson(annData, PdfRequest.class);
20
21     ...
22
23     // Annotate document
24     ServerBatchExecutor executor = new ServerBatchExecutor(service,
25                                                             server.getExecutor(), "", request.getGroups(), null,
26                                                             InputFormat.PDF, file);
27
28     executor.run(ServerProcessor.class, server.getContext());
29     Corpus corpus = executor.getProcessedCorpora().get(0);
30
31     // Return the annotation results
32     return convertCorpusToJson(corpus.getText(), corpus);
33 }

```

Listing 4: Web service to annotate a PDF file snippet.

The export web services follow the same structure, so were implemented two methods, one to annotate texts, which receives the text, a service name, a set of entity groups and an output format, from the ones supported by Neji, processes the data through a pipeline, and provides the annotation results in the requested output format. And another specific for PDF documents, which receives a PDF file, a service name, a set of entity groups and an output format, and provides the annotation results in the requested output format (see listing 5).

```

1  @POST
2  @Path("/export/{serviceID}")
3  @Consumes(MediaType.MULTIPART_FORM_DATA)
4  public Response export(@FormParam("pdf_file") InputStream file,
5                        @FormParam("pdf_file") FormDataContentDisposition fileInfo,
6                        @FormParam("ann_data") String annData,
7                        @PathParam("serviceID") String serviceID) {
8
9      Server server = Server.getInstance();
10
11     // Instantiate pdf request object
12     Gson gson = new Gson();
13     final PdfRequest request = gson.fromJson(annData, PdfRequest.class);
14     OutputFormat outFormat = getOutputFormat(server.getContext(),
15                                             request.getType());
16
17     ...
18
19     return Response.ok(new StreamingOutput() {
20         @Override
21         public void write(OutputStream outputStream) throws IOException,
22                         WebApplicationException {
23
24             service = Server.getInstance().getService(serviceID);
25
26             ...
27
28             ServerBatchExecutor executor = new ServerBatchExecutor(service,
29                             server.getExecutor(), "", request.getGroups(),
30                             outFormat, InputFormat.PDF, file);
31             executor.run(ServerProcessor.class, server.getContext());
32             String outputText = executor.getAnnotatedText();
33             IOUtils.write(outputText, outputStream, "UTF-8");
34
35         }
36     }, MediaType.APPLICATION_OCTET_STREAM)
37         .header("Content-Disposition",
38               "attachment; filename=\"text.\" +
39               outFormat.name().toLowerCase() + \".\"")
40         .build();
41 }

```

Listing 5: Snippet of the web service to annotate a PDF document and export the results in a specified format.

Additionally, was also implemented a web service that only extracts the text from a PDF document,

not performing the concept recognition steps. It receives a PDF file and returns the extracted text from it. To do this was built a custom pipeline that is composed by only two modules: the PDF reader module and a writer module.

Each annotation service is associated to a particular processing pipeline, which processes the data using the resources and properties configured for that service. Neji allows the creation of pipelines with multiple output formats, but just only one input format. In order to allow the use of the same pipeline to process both RAW texts and PDF files, Neji behaviour was adapted to allow the update of a pipeline input format in real time, without restarting the server or introducing any delays. So when an annotation service is called, if the input format is not the desired one, then the pipeline context is updated and the correct reader module will be used in the processing (see listing 6).

```
1   ...
2
3   ContextConfiguration config = context.getConfiguration();
4
5   if (!config.getInputFormat().equals(inputFormat)) {
6       ContextConfiguration newConfig = new ContextConfiguration.Builder()
7           .withInputFormat(inputFormat)
8           .withOutputFormats(config.getOutputFormats())
9           .withParserTool(config.getParserTool())
10          .withParserLanguage(config.getParserLanguage())
11          .withParserLevel(config.getParserLevel())
12          .build();
13
14       context.setConfiguration(newConfig);
15   }
16
17   ...
```

Listing 6: Snippet of updating context to generate a pipeline with a reader associated with the requested input format.

In conclusion, Neji web services offers five web services to process texts and PDF files:

- Annotation of RAW texts;
- Annotation of PDF files;
- Export annotation results from RAW texts in a specific format;
- Export annotation results from PDF files in a specific format;
- Extract the text from a PDF file.

In order to facilitate the use and integration of these web services in other projects were also developed a Java library and a Python module that allows the use of these web services programmatically.

3.3 EGAS

Egas is a web-based platform for biomedical text mining and collaborative curation. It allows users to annotate texts with concept occurrences and relation between these concepts. Annotation can be performed automatically, using the available services for automatic concept and relation identification, or manually, wherein a user can add new annotations and also edit or remove any automatically added annotations. The results can be then exported in various standard formats. But Egas was also developed focusing on collaborative curation, and therefore a team can use this web tool to annotate documents in group, being able to configure it according to their needs. Finally, Egas has also a strong focus in usability, presenting a simple, intuitive and easy to use interface.

3.3.1 ARCHITECTURE

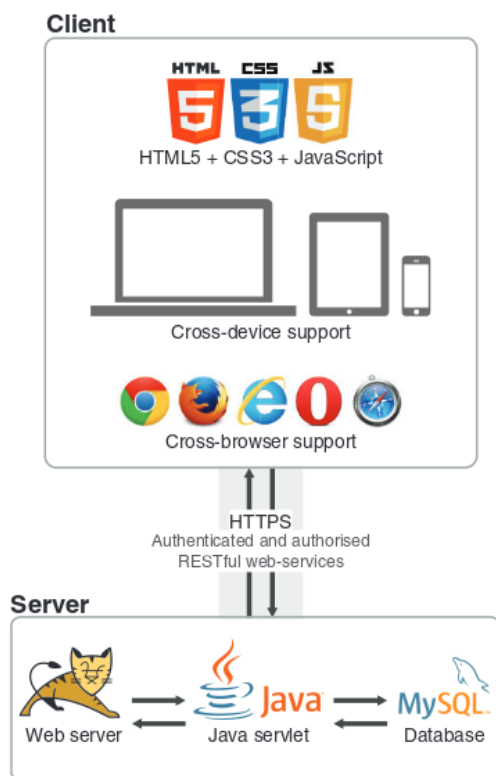


Figure 3.8: Egas architecture diagram. (from [34])

Figure 3.8 illustrates the architecture of Egas, which is divided in two core parts: server and client. The server-side is responsible for processing the data and storing it and client-side is responsible for the interaction with the users web browsers.

SERVER

The server-side stores all the information data in a MySQL¹⁰ relational database, and provides services to access and interact with it through a RESTful web-service developed in Java and deployed in an Apache Tomcat¹¹ web server. To secure those web services, their access requires authentication and authorization per user, and to guarantee that the exchanged data is fully protected, the communication between the server and client is done through a secured and encrypted channel using HTTPS.

CLIENT

The client-side was developed focusing on usability, user interface and compatibility across web browsers and devices. Therefore, were used standard web technologies such as, HTML, CSS and JavaScript, which are supported by the most popular web-browsers in different devices. The use of these technologies in combination with simple and fast algorithms provide a fast representation of the documents text and all annotations and relations.

User interface

When developing a web application one important point to take in consideration is its usability. It should be simple, intuitive and easy to use, so users can use it to import and annotate documents in a fast and uncomplicated way. In order to provide such an interface, popular front-end technologies, such as Bootstrap, jQuery and Handlebars, were used.

3.3.2 PDF HANDLING AND VISUALIZATION

As we have seen, Egas is a web-based platform for biomedical text mining and collaborative curation, that supports both manual and automatic annotation and offers a user-friendly interface.

The final objective of this thesis is to adapt Egas to support the processing of PDF articles, not only the extraction of text, but the interface should also be prepared to allow the simultaneous visualization of the extracted text alongside the original PDF file, allowing the "navigation" between both zones, synchronizing the text annotation area and the PDF visualization area.

Fist Egas was extended to allow the import of article in PDF format. For each import format Egas has an individual web service that handles the import of article, extracting its text and creating the whole data structure needed for allowing the eventual annotation of that article. Thus, a new web service, responsible for importing PDF articles was developed. This method receives a PDF file and uses the Neji web services to extract the text from it. The remaining needed data structure is then created following the exact same steps of the other import web services. The PDF files are stored in Egas file system.

When an article is requested by a user, Egas splits the text of that article into sentences and

¹⁰<https://www.mysql.com/>

¹¹<http://tomcat.apache.org/>

presents them in its interface. However, Neji already produces the sentences for the documents it processes, and that sentences are associated with extra information extracted from the PDF files, like sentences positions and rhetorical categories. Therefore, Egas data structure was extended to allow the storing of the sentences of an article alongside with the extra data provided by the PDF files. A new sentence structure was thus created in the database, and an article can be associated with one or more sentences. This structure contains the following fields:

- Sentence index: Index of the sentence (allows the sorting of the sentences);
- Start: Start index of the sentence in the full article text;
- End: End index of the sentence in the full article text;
- Page: PDF page number where the sentence occurs;
- Start position: Indicates the position where a sentence starts in a PDF page (position of the first word of the sentence). This field is divide in two, one gives the position of the upper left corner of the word, and the other gives the position of the lower right corner of the word;
- End position: Indicates the position where a sentence ends in a PDF page (position of the last word of the sentence). This field is divide in two, one gives the position of the upper left corner of the word, and the other gives the position of the downer right corner of the word;
- Leftmost x position: Leftmost x position from all sentence words;
- Rightmost x position: Rightmost x position from all sentence words;
- Type: Rhetorical category of the sentence.

In order to improve the performance of Egas, the import methods of all formats were adapted to perform the splitting process of the article text and store the indexes of the sentences in the database. So when an article is requested, it is not needed to repeat the splitting process because the sentences information are already stored in the database, reducing the processing time of the request.

To allow the embedding of the original PDF article in Egas web interface, several JavaScript libraries were studied, and PDF.js¹² seemed to be most suitable one. PDF.js is a JavaScript library developed to parse and render PDF files using web standards such as HTML5, and can be used as a part of a website or browser. Besides that it offers a large set of features tha can be useful in the interaction and visualization of the PDF articles. To mantain cross-browser support, the PDF.js visualizer was embedded in an iframe HTML element (see listing 7). This solution was implemented to overcome the possibility of browsers can use their own PDF visualizers. With this solution all browsers will always use the PDF.js visualizer.

```
1 <div id="pdf_viewer" class="pdf_viewer">
2   <iframe id="pdf_iframe" name="pdf_iframe"
3       src="assets/js/libs/pdfjs/web/viewer.html"/>
4 </div>
```

Listing 7: PDF.js visualizer embedded in an iframe element snippet.

¹²<https://mozilla.github.io/pdf.js/>

So, when a PDF is imported to a project, the file is sent to the server and is stored in its file system. In order to retrieve the PDF file from the server a new method was implemented in Egas RESTful API, that takes as input parameter an article id, fetches the file from the file system, converts it to a base64 string and returns that string to the client. PDF.js visualizer is able to render PDFs from an array of 8-bit unsigned integers, so the base64 string received, that represents the PDF, is converted by a simple algorithm to an array of that type, that it is directly passed to the PDF.js and therefore the PDF is presented in the visualizer.

In order to extend the automatic annotation services provided by Egas to annotate articles, was integrated the possibility to use all existent Neji annotation services in Egas, through the use of Neji's RESTful API.

To synchronize the PDF visualizer and the text area were implemented some features in the client, such as automatic scroll of the PDF visualizer, mapping between the text area and PDF visualizer, and visualization of the annotations over the PDF article.

Automatic scroll

In order to automatically scroll the PDF visualizer when a user is navigating in the text area, an algorithm was implemented to check which sentences are visible in the text area, each time it is scrolled. When it detects that the sentences of another PDF page are reached it automatically scrolls the PDF visualizer to the corresponding page (see listing 8). This feature can be enabled or disabled by the user at any time.

```

1  // Check which sentences are visible in annotation area
2  $.each(arr, function (index, sentence) {
3
4      var $sentence = $("#line_" + sentence + "_number");
5
6      if ($sentence.position().top + $sentence.height() > 0
7          && $sentence.position().top < $ann_area.height() ) {
8          // Get sentence id
9          var elemId = $sentence.attr("id");
10         var endIndex = elemId.lastIndexOf("_");
11         sentenceId = elemId.substring(5, endIndex);
12         return;
13     }
14 });
15
16 if (sentenceId) {
17     // Get current page in pdf viewer
18     var pdfIframe = document.getElementById("pdf_iframe");
19     var pdfApp = pdfIframe.contentWindow.PDFViewerApplication;
20     var pdfViewer = pdfApp.pdfViewer;
21     var currentPage = pdfViewer._currentPageNumber;
22
23     // Get sentence page
24     var sentencePage = AnnText.sentencePageMap[sentenceId];
25
26     if (currentPage !== sentencePage) {
27         // Scroll to the new page
28         pdfViewer.scrollPageIntoView(sentencePage, null);
29     }
30 }

```

Listing 8: Automatic scroll snippet.

Mapping between the text area and PDF visualizer

Another important feature is to allow a user to identify a sentence that he is viewing in the text area, in the PDF article. In order to achieve this behaviour, when an PDF article is requested by a user, are calculated the areas of all sentences in the PDF, using the sentences positions, and also the PDF visualizer current zoom (see listing 9). This areas are associated with the corresponding sentences in the text area. So, when a user selects a sentence in the text area, the PDF viewer is scrolled to the sentence position (see listing 10), waits for the page rendering if necessary, and then the sentence is highlighted in the PDF page using its area. To perform the scroll of the PDF article were used some features of PDF.js. The highlight of areas is not supported by PDF.js and thus it was implemented on top of it using some jQuery features.

```

1  // Add safety margin
2  var safetyMargin = 3;
3  if (startX1 - safetyMargin >= 0) startX1 -= safetyMargin;
4  if (leftX - safetyMargin >= 0) leftX -= safetyMargin;
5  if (rightX + safetyMargin <= pageWidth) rightX += safetyMargin;
6  if (endX2 + safetyMargin <= pageWidth) endX2 += safetyMargin;
7  if (startY1 - safetyMargin >= 0) startY1 -= safetyMargin;
8  if (endY2 + safetyMargin <= pageHeight) endY2 += safetyMargin;
9
10 // Handles first and last line individually, and the remaining
11 // of the sentence
12 if ((startX1 > leftX) && (endX2 < rightX)) {
13     areas.push([startX1, startY1, rightX, startY2]);
14     areas.push([leftX, startY2, rightX, endY1]);
15     areas.push([leftX, endY1, endX2, endY2]);
16 // Handles the first line and the remaining of the sentence
17 } else if (startX1 > leftX) {
18     areas.push([startX1, startY1, rightX, startY2]);
19     areas.push([leftX, startY2, rightX, endY2]);
20 // Handles the last line and the remaining of the sentence
21 } else if (endX2 < rightX) {
22     areas.push([leftX, startY1, rightX, endY1]);
23     areas.push([leftX, endY1, endX2, endY2]);
24 // The sentence represents just one line in the pdf
25 } else {
26     areas.push([startX1, startY1, endX2, endY2]);
27 }
28
29 return areas;

```

Listing 9: Calculation of a sentence areas snippet.

```

1  // Calculate page position
2  var pageYpos = baseHeight - sentenceData.startY1
3    + borderMargin;
4  var pageXpos = sentenceData.leftX - borderMargin;
5  var pagePos = [null, {name: 'XYZ'}, pageXpos, pageYpos];
6
7  // Move pdf to the sentence
8  pdfViewer.scrollPageIntoView(pageNumber, pagePos);

```

Listing 10: Snippet of the automatic scroll of the PDF viewer to a sentence position.

The inverse behaviour is also an important feature, i.e, allow a user to identify a sentence

he is viewing in the PDF article, in the text area. Thus, this behaviour was also implemented using the information of the sentences areas. When a double click is performed over the PDF article, are identified the page where it was done, and the exact position of the click within that page. An algorithm then checks if that position is inside any sentence area of that page. If the position matches an area, the text area is automatically scrolled to the corresponding sentence, and the sentence is highlighted using some jQuery features. All the developed algorithms and methods for synchronization between the two areas take into account the current zoom of the PDF visualizer.

Visualization of the annotations over the PDF article

Since the annotations positions in the PDF pages are also provided by Neji, was also implemented a method to allow the visualization of the annotations over the PDF article. To perform this, the method starts by calculating the areas of all annotations, and when a page of the PDF is rendered in the PDF viewer, it draws a rectangle upon the area of each annotation from that page, with the same colors that are presented in the text area. Additionally if an annotation in the PDF is mouse hovered then the corresponding annotation in the text area is automatically selected. This feature can be enabled or disabled by the user at any time.

APPLICATION AND RESULTS

This chapter presents the resultant systems and applications, as well as the new features of Neji, Neji web services and Egas. Besides that, the general usage of those systems will also be addressed.

4.1 NEJI

Neji is a modular framework for biomedical concept recognition that allows the extraction of biomedical concepts from documents and texts, using the most appropriate and optimized techniques, based on dictionary matching and Machine Learning.

Neji integrates a Command Line Interface (CLI), which tries to be simple and flexible at the same time, allowing it to be used easily by developers and researchers. Figure 4.1 presents Neji Command Line Interface, which offers the following features:

- Various input formats: raw, XML, BioC and PDF. When XML is used, a set of tags can be provided by the user, indicating what text should be taken in account;
- Various parsing levels: tokenization, lemmatization, POS, chunking and dependency parsing. If not provided, Neji selects an appropriated level depending on the situations, tokenization if it does not use ML in the annotation process, chunking otherwise;
- Support for multithreading, allowing a user to choose the number of threads to be used;
- Support for compressed files. If the files are compressed, then a decompression process is performed first;
- Different annotation techniques: dictionary matching and Machine Learning;
- Various output formats: XML, NEJI, A1, CoNLL, JSON, B64, JSONPDF and others;
- And others.

So to annotate a corpus with Neji it is only needed to use a bash command line, such as:

```

usage: ./neji.sh -i <folder> -if [XML|RAW|BIOC|PDF] [-x <tags>] [-f <wildcard filter>] [-r <rules file>] -o <folder> -of
[XML|NEJI|A1|CONLL|JSON|JSONPDF|B64|BIOC|PIPE|PIPEXT|BC2] [-p <folder>] [-d <folder>] [-m <folder>] [-custom
MODULE1-ARG1-ARG2|MODULE2-ARG1-ARG2|MODULE3-ARG1-ARG2] [-ptool <PARSING_TOOL>] [-plang <PARSING_LANGUAGE>] [-plvl <PARSING_LEVEL>]
[-pcls <processor.class>] [-c] [-t <threads>] [-v] [-s]

Neji: modular biomedical concept recognition made easy, fast and accessible.
-c,--compressed                If files are compressed using GZip.
-custom,--custom-modules <arg> Names of custom modules to be used in order, separated by pipes. If a specified module are not a reader or a
writer, it will be executed after dictionary and model processing.
-d,--dictionaries <arg>       Folder that contains the dictionaries.
-f,--input-filter <arg>       Wildcard to filter files in input folder
-h,--help                      Print this usage information.
-l,--input <arg>              Folder with corpus files.
-if,--input-format <arg>      BIOC, RAW, XML or PDF
-m,--models <arg>            Folder that contains the ML models.
-noids,--include-no-ids       If annotations without IDs should be included.
-o,--output <arg>             Folder to save the annotated corpus files.
-of,--output-formats <arg>    A1, B64, BIOC, CONLL, JSON, JSONPDF, NEJI or XML
-p,--parser <arg>            Folder that contains the parsing tool.
-pcls,--processor-class <arg> Full name of pipeline processor class.
-plang,--parsing-language <arg> DANISH, DUTCH, ENGLISH, FRENCH, GERMAN, PORTUGUESE or SWEDISH (ENGLISH is set by default)
-plvl,--parsing-level <arg>   TOKENIZATION, POS, LEMMATIZATION, CHUNKING or DEPENDENCY (TOKENIZATION is set by default)
-ptool,--parsing-tool <arg>   GDEP or OPENNLP (GDEP is set by default)
-r,--rules-file <arg>         Rules file for pdf extraction.
-s,--server                   Use parameters to generate a deployable server.
-t,--threads <arg>           Number of threads. By default, if more than one core is available, it is the number of cores minus 1.
-v,--verbose                  Verbose mode.
-x,--xml-tags <arg>          XML tags to be considered, separated by commas.

Usage examples:
1: ./neji.sh -i input -if XML -o output -of XML -x text -d folder -t 2
2: ./neji.sh -i input -if RAW -o output -of A1 -m folder -t 4
3: ./neji.sh -i input -if XML -o output -of XML -x text -d folder1 -m folder2 -c -t 6
For more instructions, please visit http://bioinformatics.ua.pt/neji.

```

Figure 4.1: Neji command line interface.

```
./neji.sh -i input/ -if RAW -o output/ -of NEJI -d dictionaries/ -m models/
```

In the above example, a set of documents in raw format, located at “input/” directory, is provided to Neji to be annotated using the dictionaries located at “dictionaries/” directory, and the ML models located at “models/” directory. The output results will be stored at “output” directory in NEJI format.

In the scope of this thesis, one of the goals was to extend Neji framework to process PDF documents. As explained in section 3.2.2, a new input and output format were created, they were called PDF and JSONPDF respectively. If the input format is PDF, a rules file can be provided, allowing a custom identification of rhetorical categories for the text blocks from the PDF document.

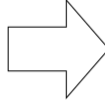
So to annotate a set of PDF documents the following command line could be used:

```
./neji.sh -i input/ -if PDF -o output/ -of JSONPDF -d dictionaries/ -m models/
```

The usage is similar to the previous explained command line, only the input format (PDF) and output format (JSONPDF) were changed. Note that any of the supported output formats could be used, but just JSONPDF contains the extra data extracted from the PDF documents. Figure 4.2 illustrates an example of the annotation results when using Neji to annotate a simple PDF document. At left is the original text of the PDF file, and at right is the output produced by Neji in NEJI format.

These functionalities are also available in the programming API of Neji. Listing 11 presents an example of annotating a PDF programmatically.

Histidinemia is an inborn error of metabolism of amino acids caused by deficiency of histidase activity in liver and skin with consequent accumulation of histidine in plasma and tissues. Histidinemia is an autosomal recessive trait usually considered harmless to patients and their offspring, but some patients and children born from histidinemic mothers have mild neurologic alterations. Considering that histidinemia is one of the most frequently identified metabolic conditions, in the present study we investigated the effect of L-histidine load to female rats during pregnancy and lactation on some parameters of phosphoryltransfer network in cerebral cortex and hippocampus of the offspring. Pyruvate kinase, cytosolic and mitochondrial creatine kinase activities decreased in cerebral cortex and in hippocampus of rats at 21 days of age and this pattern remained in the cerebral cortex and in hippocampus at 60 days of age. Moreover, adenylate kinase activity was reduced in the cerebral cortex and in hippocampus of the offspring at 21 days of age, whereas the activity was increased in the two tissues at 60 days of age. These results suggest that administration of L-histidine to female rats in the course of pregnancy and lactation could impair energy homeostasis in the cerebral cortex and hippocampus of the offspring. Considering that histidinemia is usually a benign condition and little attention has been given to maternal histidinemia, it seems important to perform more studies in the children born from histidinemic mothers.



```
S1      0 186 Histidinemia is an inborn error of metabolism of amino acids caused by
in plasma and tissues.
T1      0 12 Histidinemia UMLS:C0229992:T047:DISO
T2      19 45 Inborn error of metabolism UMLS:C0025521:T047:DISO
T3     107 112 Liver UMLS:C0023884:T023:ANAT

S2     186 387 Histidinemia is an autosomal recessive trait usually considered harmless
to patients and their offspring, but some patients and children born from histidinemic
mothers have mild neurologic alterations.
T1     186 198 Histidinemia UMLS:C0229992:T047:DISO

S3     387 695 Considering that histidinemia is one of the most frequently identified
metabolic conditions, in the present study we investigated the effect of L-histidine load to
female rats during pregnancy and lactation on some parameters of phosphoryltransfer network
in cerebral cortex and hippocampus of the offspring.
T1     404 416 Histidinemia UMLS:C0229992:T047:DISO
T2     646 681 cerebral cortex UMLS:C0007776:T023:ANAT
T3     666 677 hippocampus UMLS:C0019564:T023:ANAT

S4     695 927 Pyruvate kinase, cytosolic and mitochondrial creatine kinase activities
decreased in cerebral cortex and in hippocampus of rats at 21 days of age and this pattern
remained in the cerebral cortex and in hippocampus at 60 days of age.
T1     780 795 cerebral cortex UMLS:C0007776:T023:ANAT
T2     803 814 hippocampus UMLS:C0019564:T023:ANAT
T3     874 889 cerebral cortex UMLS:C0007776:T023:ANAT
T4     897 908 hippocampus UMLS:C0019564:T023:ANAT

S5     927 1125 Moreover, adenylate kinase activity was reduced in the cerebral cortex
and in hippocampus of the offspring at 21 days of age, whereas the activity was increased in
the two tissues at 60 days of age.
T1     982 997 cerebral cortex UMLS:C0007776:T023:ANAT
T2    1005 1016 hippocampus UMLS:C0019564:T023:ANAT

S6    1125 1326 These results suggest that administration of L-histidine to female rats
in the course of pregnancy and lactation could impair energy homeostasis in the cerebral
cortex and hippocampus of the offspring.
T1    1277 1292 cerebral cortex UMLS:C0007776:T023:ANAT
T2    1297 1308 hippocampus UMLS:C0019564:T023:ANAT
```

Figure 4.2: Example of annotation using Neji with a simple PDF file.

```
1 File pdfFile = new File("pdf_document.pdf");
2 InputStream pdfInputStream = new FileInputStream(pdfFile);
3
4 // Intantiate parser
5 Parser parser = new GDepParser(ParserLanguage.ENGLISH,
6     ParserLevel.TOKENIZATION, new LingpipeSentenceSplitter(), false);
7
8 // Instantiate reader module
9 Reader reader = new PdfReader(parser, pdfInputStream);
10
11 // Instantiate dictionary tagger module
12 DictionaryTagger dicTagger = new DictionaryTagger(
13     new VariantMatcherLoader(true).
14         load("UMLS:C2930957:T047:DISO\thantavirosis|hantavirus fever").
15         load("UMLS:C0004096:T047:DISO\tasthma").
16         getMatcher());
17
18 // Intantiate writer module
19 Writer writer = new JSONPdfWriter();
20
21 // Instatiate pipeline for PDF processing
22 Pipeline pipeline = new DefaultPipeline().
23     add(reader).
24     add(dicTagger).
25     add(writer);
26
27 // Run processing pipeline
28 OutputStream out = pipeline.run(new FileInputStream(pdfFile)).
29     get(0);
30
31 String result = out.toString();
```

Listing 11: Snippet of using the Neji programming API to annotate a PDF document.

4.1.1 NEJI WEB SERVICES

Neji web services is a web services platform that facilitates the use and access to Neji functionalities by providing an easy and intuitive web solution to manage, and use, annotation services. The RESTful API allows developers and researchers to send their input documents and receive the annotation results. Besides Neji features, Neji web services offers also some other features:

- Management of concurrent annotation services. Allows an admin to create, edit and delete one or more annotation services;
- Flexible configuration of annotation services. Each service has its own resources (dictionaries and ML models) and properties;
- Pre-loading of resources: When a resource is added to the server, it is immediately loaded into the server memory. Therefore, on an annotation request it is not needed to wait for the load of any resource because they are already ready to be used, reducing the annotation time;
- Simple and intuitive user interface for management and annotation;
- Cross-platform and cross-browser support.

In the developed web interface, an administrator and a common user have different permissions. An administrator can add, edit and remove resources from the server, create and manage new annotation services, and annotate documents using the provided web services or annotation interface. A common user can not add resources or create new annotation services, but can use any of the provided annotation services, using both the web services and annotation interface.

Home page

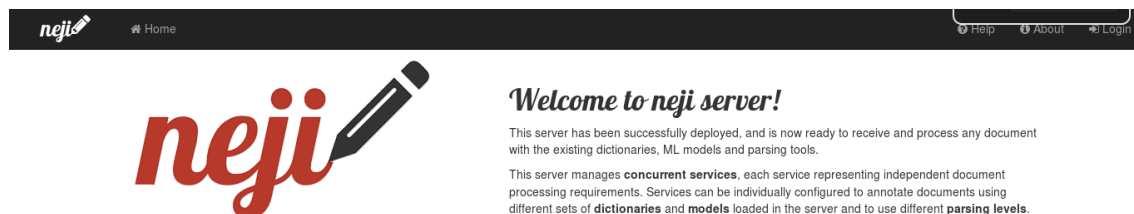


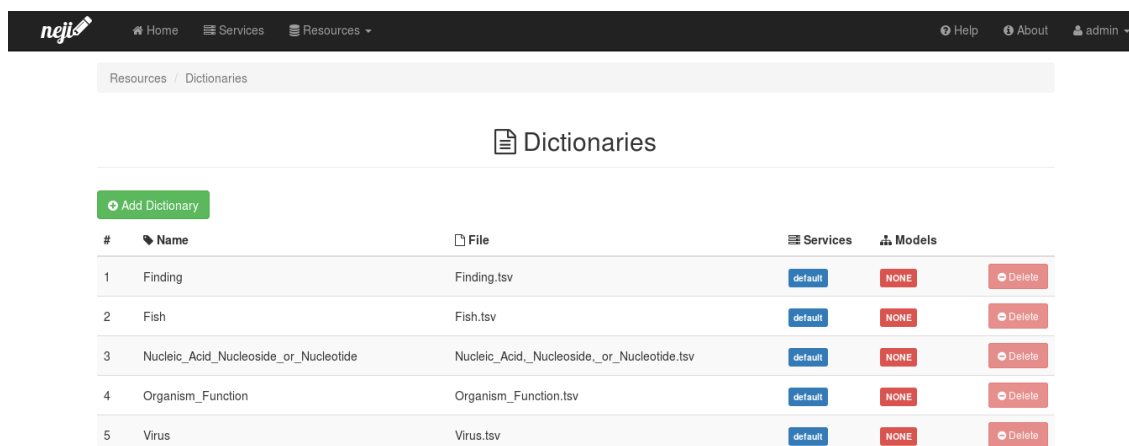
Figure 4.3: Neji web services home page.

Figure 4.3 presents Neji web services home page, which has informative content about Neji web services and provides access to an help page. Administrators can authenticate themselves using the Login button.

RESOURCES

In the dictionaries page (see figure 4.4) an admin can see a list of all loaded dictionaries in the server. For each dictionary the following information is provided: name, original file name, list of services that use it in the annotation process and list of models that use it in the normalization process.

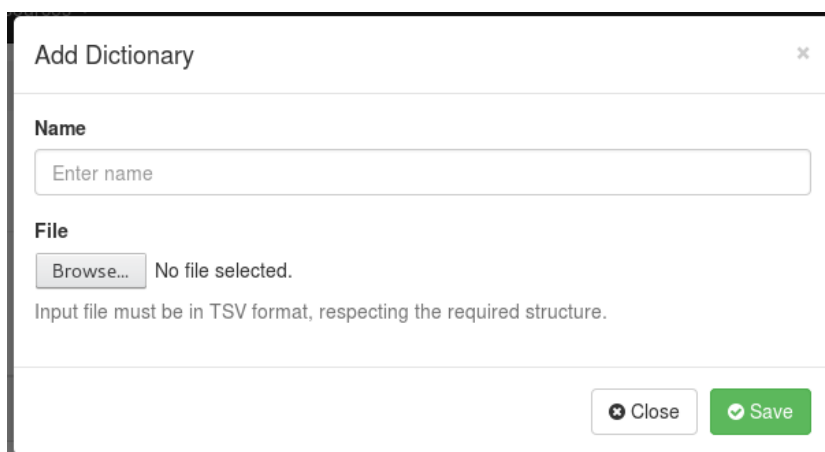
New dictionaries can be added and deleted in this page.



#	Name	File	Services	Models	
1	Finding	Finding.tsv	default	NONE	Delete
2	Fish	Fish.tsv	default	NONE	Delete
3	Nucleic_Acid_Nucleoside_or_Nucleotide	Nucleic_Acid_Nucleoside_or_Nucleotide.tsv	default	NONE	Delete
4	Organism_Function	Organism_Function.tsv	default	NONE	Delete
5	Virus	Virus.tsv	default	NONE	Delete

Figure 4.4: Neji web services dictionaries page.

To add a new dictionary an administrator needs to press the Add Dictionary button and fill a form, giving a dictionary name and pointing the dictionary file. The file will then be uploaded to the server and associated with the chosen name (see figure 4.5).



Add Dictionary

Name

File

No file selected.

Input file must be in TSV format, respecting the required structure.

Figure 4.5: Neji web services add dictionary form.

In the ML models page (see figure 4.6) an admin can see a list of all loaded ML models in the server. For each model the following information is provided: name, original file name, list of normalization dictionaries and list of services that use it in the annotation process. New models can be added, edited and deleted in this page.

To add a new dictionary an admin needs to press the Add Model button and fill a form, choosing a model name, pointing the model files and selecting the normalization dictionaries for this model. The model files will then be uploaded to the server and associated with the chosen name and normalization dictionaries (see figure 4.7).

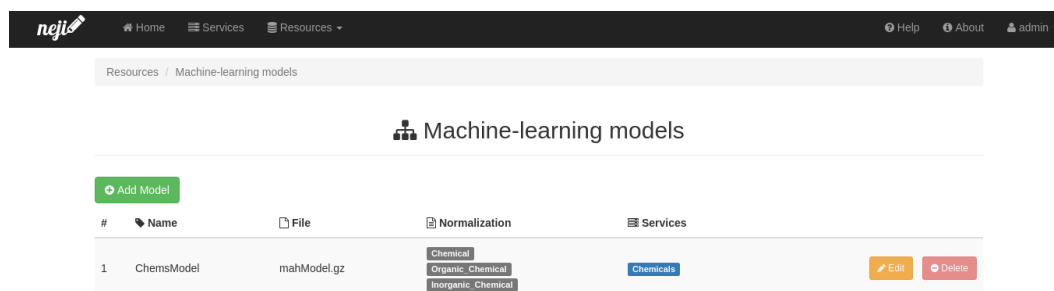


Figure 4.6: Neji web services models page.

Add Machine-learning model

Name
Enter name

File
Choose File No file chosen
Input file must be in GZIP format, generated by Neji training features.

Configuration File
Choose File No file chosen

Properties File
Choose File No file chosen

Normalization

Available

- Finding
- Fish
- Nucleic_Acid_Nucleoside_or_Nucleotide
- Organism_Function
- Virus
- Entity
- Individual_Behavior
- Vertebrate
- Food
- Chemical

Selected

Close Save

Figure 4.7: Neji web services add model form.

SERVICES

In the services page (see figure 4.8) an administrator can see a list of all active services running in the server. For each annotation service the following information is provided: name, number of dictionaries it uses, number of models it uses and an illustrative image or logo. New services can be created, accessed, edited and deleted in this page.

To add a new annotation service an administrator needs to press the Add Service button and fill a form, with a service name, an image or logo, the dictionaries and models to be used in the annotation process, the parsing level and select if the annotations without identifiers should be taken into account or not (see figure 4.9). Optionally, semantic group mapping can be used to map the default entity type from the dictionaries or ML models to the desired name.

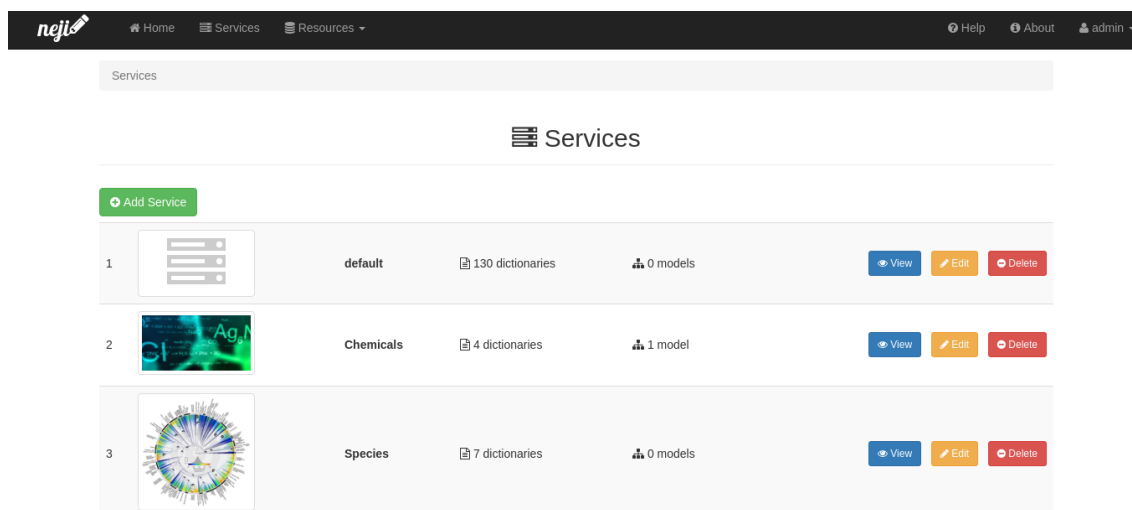


Figure 4.8: Neji web services annotation services page.

The screenshot shows the 'Add Service' form in the Neji web interface. The form includes the following sections:

- Name:** A text input field containing 'GenesAndDisorders' with a green checkmark.
- Logo:** A 'Choose File' button and the text 'No file chosen'.
- Dictionaries:** A list of available dictionaries on the left and a 'Selected' dropdown on the right containing 'Anatomical_Abnormality_T190_DISO'.
- Machine-learning models:** A list of available models on the left and a 'Selected' dropdown on the right containing 'ChemsModel'.
- Semantic groups mapping:** Two input fields for 'DISO' (containing 'DISO') and 'PRGE' (containing 'Gene').
- Parsing level:** A dropdown menu set to 'Chunking'.
- Include annotations without identifiers:** A checked checkbox.

At the bottom right, there are 'Close' and 'Save' buttons.

Figure 4.9: Neji web services add service form.

ANNOTATION

The annotation services pages can be accessed by anyone. These are accessed through a general hyperlink, composed by the website domain and the service name:

`https://neji-web-services-domain.com/annotate/SERVICE_NAME`

If the name of the service that we want to access is Chemicals, then the hyperlink for that page is `https://neji-web-services-domain.com/annotate/Chemicals`

The annotation input page is presented in figure 4.10, and it contains two major areas:

- Semantic groups control (on the left): allows the selection of the entity groups that should be recognized and annotated. One semantic group needs to be selected in order to perform the annotation;
- Text input controls (on the right): has an area to insert the text to annotate, and offers three input methods: upload a file, annotate PubMed article and try sample.

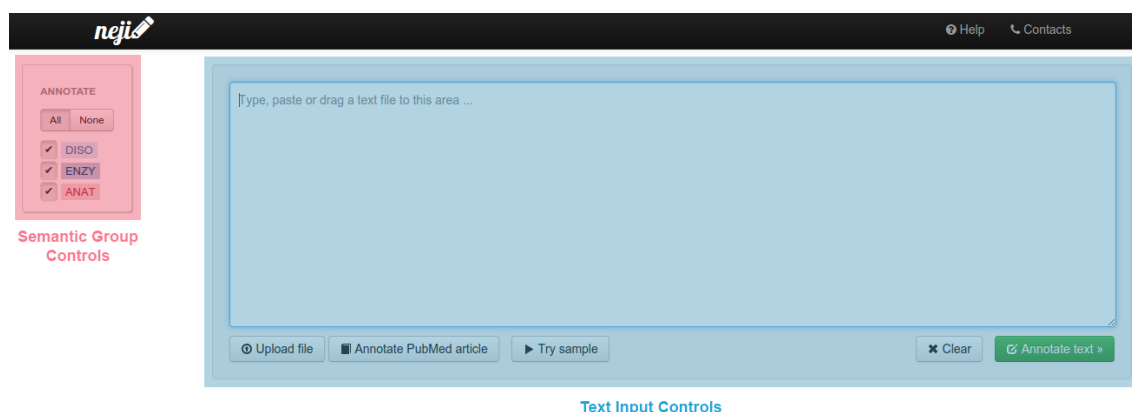


Figure 4.10: Neji web services annotation input page.

The Try Sample button allows a quick testing and experimentation of the system using the available text samples.

Plain text annotation

To annotate text an user can directly type or paste text in the text-area, click in the Upload file button and select a file to annotate, or drag a text file or a snippet of text from his file system to the text-area. After typing or pasting text in the text-area, an user just needs to click in the Annotate text button to trigger the annotation process. If a file is uploaded or dragged in the text-area, the annotation starts immediately without further input.

PubMed abstract annotation

This option allows automatically fetching of PubMed abstracts text from NCBI servers. When a

user clicks in the Annotate PubMed article button on the text input controls, a pop-up dialog will ask him to enter the PMID of the publication he wants to annotate (see figure 4.11).

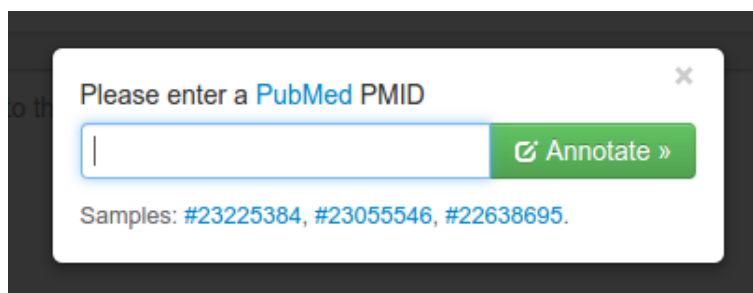


Figure 4.11: Neji web services pubmed article pop-up.

After submitting text, a file or a PMID, the output interface is presented, which is composed of two primary elements (see figure 4.12):

- Semantic groups control (on the left): allows the selection of the concept types that should be presented;
- The annotated text area (on the right): contains the text and the highlight of the annotations.

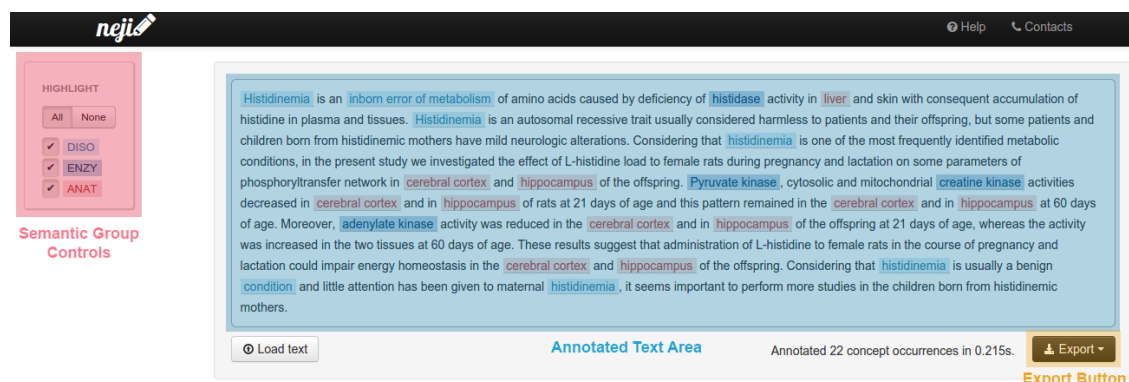


Figure 4.12: Neji web services annotation output page.

Plain text content or PubMed articles' title and abstract are displayed in the annotated text area, where all recognized concepts are highlighted in the color corresponding to the concept type, or semantic group. Terms tagged with more than one concept from different types are deemed ambiguous.

Concept popovers

When a user hovers the mouse or clicks in any highlighted term, a concept popover containing details about the identified concepts is shown. Figure 4.13 illustrates an example of a concept popover.

Exporting results

Allows exporting the recognized concepts in multiple popular formats. A user just needs to click in the Export button and select his preferred format (see figure 4.14).

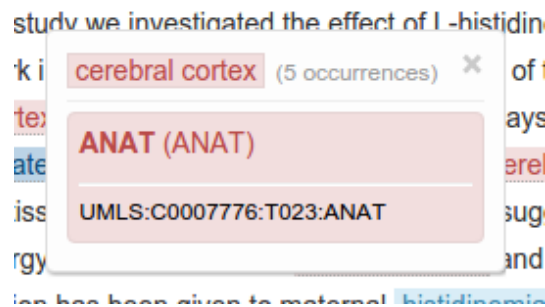


Figure 4.13: Neji web services concept popover example.

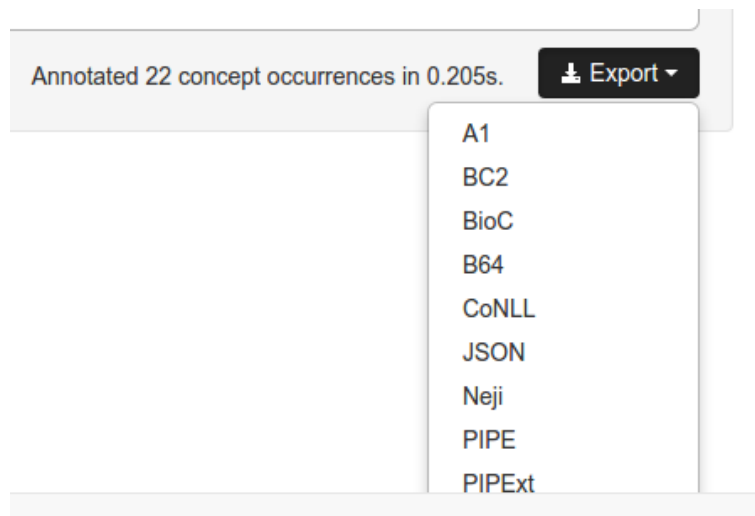


Figure 4.14: Neji web services exporting results example.

WEB SERVICES

The developed RESTful API offers a set of web services that allow an easy and fast annotation of plain texts and PDF documents. All web services are accessed through an endpoint, composed by a base domain and a specific path for each web service, such as:

[https://neji-web-domain.com/annotate/\[PATH\]](https://neji-web-domain.com/annotate/[PATH])

Annotate text web service

The annotate text web service can be accessed through an endpoint like [https://neji-web-domain.com/annotate/\[service name\]/annotate/](https://neji-web-domain.com/annotate/[service name]/annotate/), where [service name] is the name of the service that should be used to annotate the text. Table 4.1 contains the service parameters.

Table 4.1: Parameters of annotate text web service.

Name	Type	Default	Description
text	string: "..."	None (required)	Text to annotate.
groups	object: {"GROUP":true false, ...}	If omitted: all groups true; If any groups present: all others false.	Types of concepts to annotate.

The response is a JSON object with the following keys:

Table 4.2: Response of annotate text web service.

Name	Type	Description
text	string: "..."	Submitted text.
entities	array of strings: ["...", ...]	Concepts recognized in text.
ids	object of objects: { "CID": { "name": "...", "refs": ["...", ...], ...} }	Concept metadata, with concept IDs.

Figure 4.15 presents an example of using the annotate text web service.

The screenshot shows a REST client interface with a POST request to the endpoint `https://srv7-ieeta.ieeta.pt:8017/annotate/default/annotate`. The request body is a JSON object with the following structure:

```
{
  "groups": {},
  "text": "In Duchenne muscular dystrophy (DMD), the infiltration of skeletal muscle by immune cells aggravates disease, yet the precise mechanisms behind these inflammatory responses remain poorly understood. Chemotactic cytokines, or chemokines, are considered essential recruiters of inflammatory cells to the tissues. We assayed chemokine and chemokine receptor expression in DMD muscle biopsies (n = 9, average age 7 years) using immunohistochemistry, immunofluorescence, and in situ hybridization. CXCL1, CXCL2, CXCL3, CXCL8, and CXCL11, absent from normal muscle fibers, were induced in DMD myofibers. CXCL11, CXCL12, and the ligand-receptor couple CCL2-CCR2 were upregulated on the blood vessel endothelium of DMD patients. CD68(+) macrophages expressed high levels of CXCL8, CCL2, and CCL5. Our data suggest a possible beneficial role for CXCR1/2/4 ligands in managing muscle fiber damage control and tissue regeneration. Upregulation of endothelial chemokine receptors and CXCL8, CCL2, and CCL5 expression by cytotoxic macrophages may regulate myofiber necrosis."
}
```

The response body is a JSON object with the following structure:

```
{
  "entities": [
    "blood|UMLS:C0851353:T047:DISO|679",
    "Upregulation|UMLS:C0841904:T044:PHYS|929"
  ],
  "ids": {},
  "text": "In Duchenne muscular dystrophy (DMD), the infiltration of skeletal muscle by immune cells aggravates disease, yet the precise mechanisms behind these inflammatory responses remain poorly understood. Chemotactic cytokines, or chemokines, are considered essential recruiters of inflammatory cells to the tissues. We assayed chemokine and chemokine receptor expression in DMD muscle biopsies (n = 9, average age 7 years) using immunohistochemistry, immunofluorescence, and in situ hybridization. CXCL1, CXCL2, CXCL3, CXCL8, and CXCL11, absent from normal muscle fibers, were induced in DMD myofibers. CXCL11, CXCL12, and the ligand-receptor couple CCL2-CCR2 were upregulated on the blood vessel endothelium of DMD patients. CD68(+) macrophages expressed high levels of CXCL8, CCL2, and CCL5. Our data suggest a possible beneficial role for CXCR1/2/4 ligands in managing muscle fiber damage control and tissue regeneration. Upregulation of endothelial chemokine receptors and CXCL8, CCL2, and CCL5 expression by cytotoxic macrophages may regulate myofiber necrosis."
}
```

Figure 4.15: Example of using the annotate plain text web service.

Export web service

The export web service can be accessed through a endpoint like `https://neji-web-domain.com/annotate/[service name]/export/`, where [service name] is the name of the service that should be used to annotate the text. Table 4.3 contains the service parameters.

Table 4.3: Parameters of export web service.

Name	Type	Default	Description
text	string: "..."	None (required)	Text to annotate.
groups	object: {"GROUP":true false, ...}	If omitted: all groups true; If any groups present: all others false.	Types of concepts to annotate.
format	string: "..."	"a1"	Desired output format.

The response are the annotation results in the selected output format. Figure 4.16 presents an example of using the export web service.

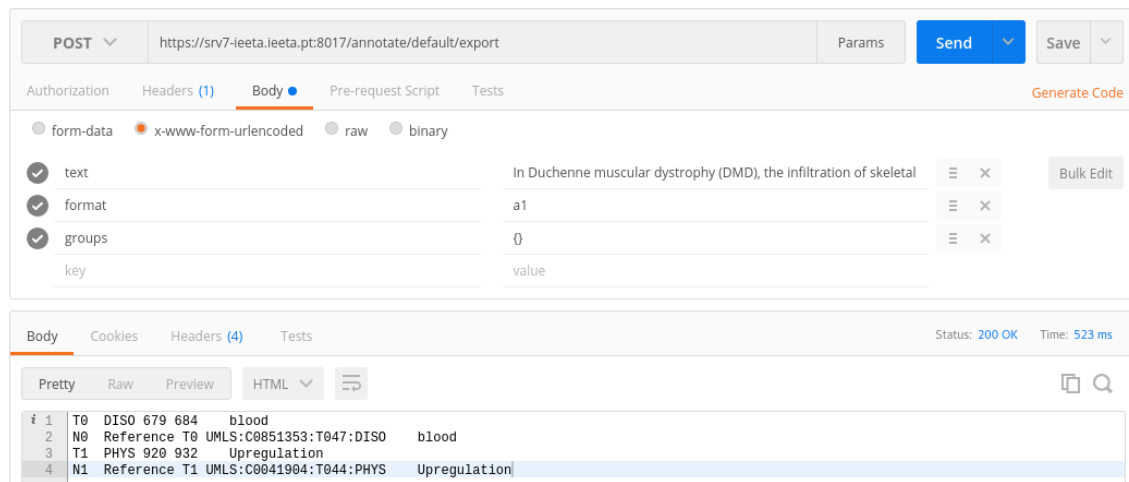


Figure 4.16: Example of using the export plain text web service.

Extract PDF document text web service

The extract Portable Document Format document text web service can be accessed through a endpoint like <https://neji-web-domain.com/annotate/pdf/extract/>. Table 4.4 contains the service parameters.

Table 4.4: Parameters of extract PDF document text web service.

Name	Type	Default	Description
pdf_file	file	None (required)	PDF file to extract text.

The response is the full text of the provided PDF file. Figure 4.17 presents an example of using the extract PDF document text web service.

Annotate PDF document web service

The annotate PDF document web service can be accessed through an endpoint like [https://neji-web-domain.com/annotate/pdf/annotate/\[service name\]/](https://neji-web-domain.com/annotate/pdf/annotate/[service name]/), where [service name] is the name of the service that should be used to annotate the text of the PDF. Table 4.5 contains the service parameters.

Table 4.5: Parameters of annotate PDF document web service.

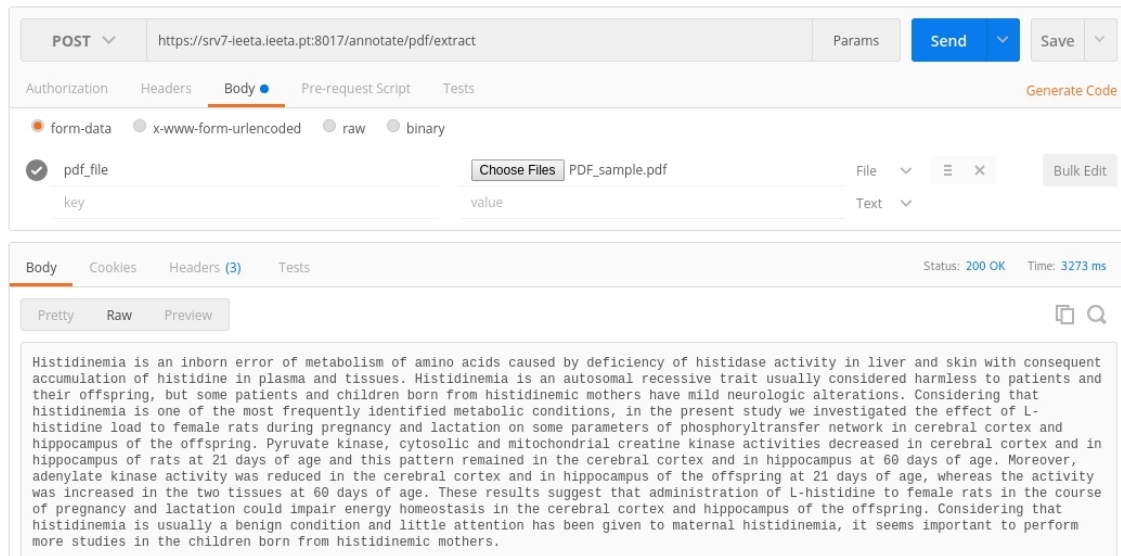


Figure 4.17: Example of using the extract PDF document text web service.

Name	Type	Default	Description
pdf_file	file	None (required)	PDF file to extract text.
groups	object: {"GROUP":true false, ...}	If omitted: all groups true; If any groups present: all others false.	Types of concepts to annotate.

The response is a JSON object similar to the one already presented for the annotate plain text web service (see table 4.2). Figure 4.18 presents an example of using the annotate PDF document web service.

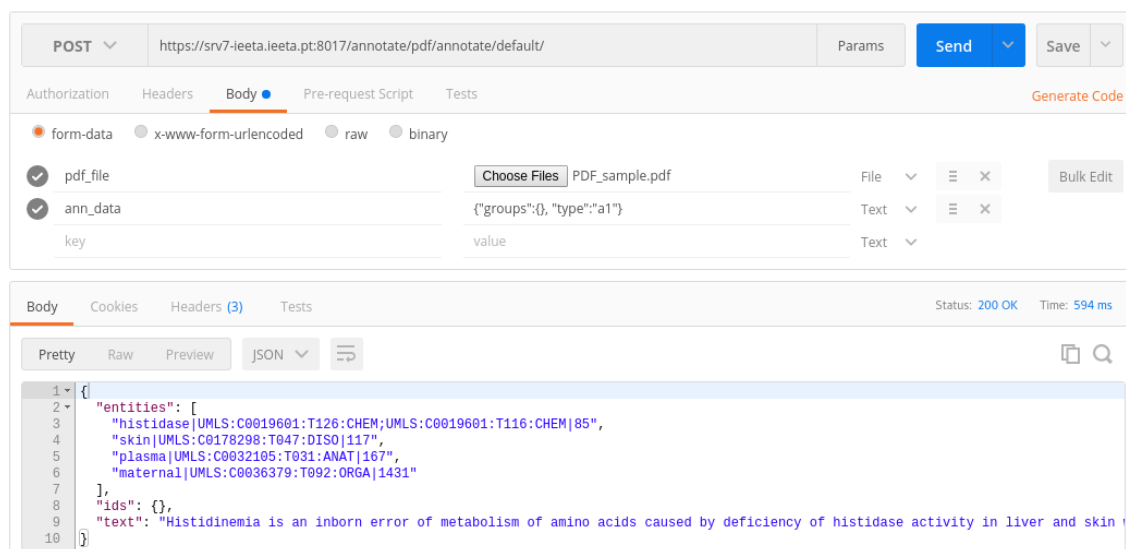


Figure 4.18: Example of using the annotate PDF document web service.

Export web service for PDF documents

The export web service for PDF documents can be accessed through an endpoint like `https://neji-web-domain.com/annotate/pdf/export/[service name]/`, where [service name] is the name of the service that should be used to annotate the text of the PDF. Table 4.6 contains the service parameters.

Table 4.6: Parameters of export web service for PDF documents.

Name	Type	Default	Description
pdf_file	file	None (required)	PDF file to extract text.
groups	object: {"GROUP":true false, ...}	If omitted: all groups true; If any groups present: all others false.	Types of concepts to annotate.
type	string: "..."	"a1"	Desired output format.

The response are the annotation results in the selected output format. Figure 4.19 presents an example of using the export PDF document web service.

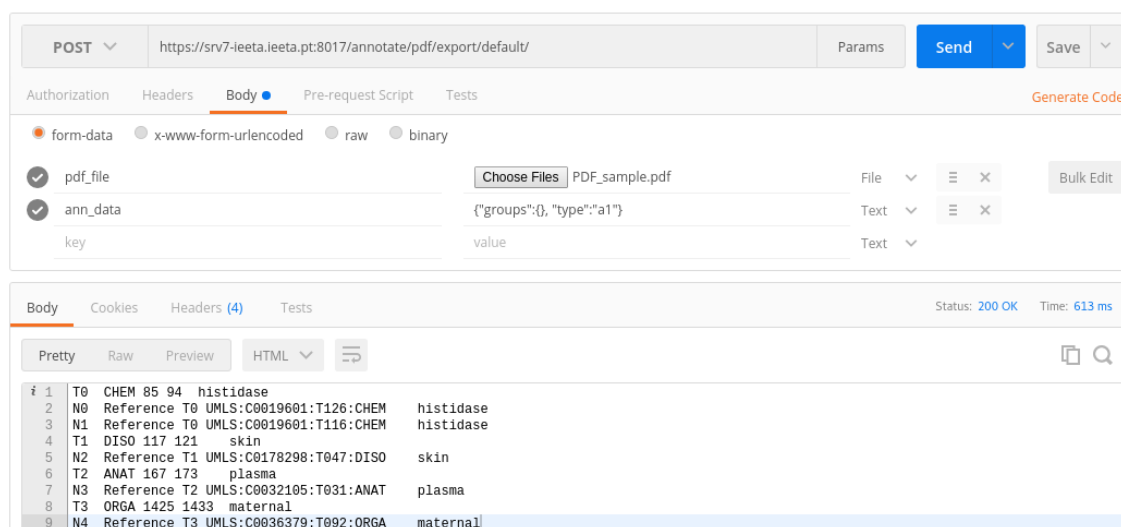


Figure 4.19: Example of using the export web service for PDF document.

These functionalities are also available in the programming API, provided by the developed Java library and Python module. Listing 12 presents an example of using the Java library to annotate a PDF document.

```
1  ServicesClient client = new ServicesClient(url, port);
2
3  String service = "Chemicals";
4  File pdfFile = new File("pdf_document.pdf");
5  List<String> groups = new ArrayList();
6  groups.add("CHEM");
7
8  JSONObject response = client.annotatePdf(service, pdfFile, groups);
```

Listing 12: Snippet of using the Neji Java library to annotate a PDF document.

4.2 EGAS

Egas is a web-based platform for text mining and collaborative curation that offers a simple and easy to use interface, as well as fast automatic annotation services. In the scope of this work, one of the main goals was to allow Egas to handle and manipulate PDF articles, and thus the following features were added:

- Import and processing of articles in PDF format;
- Simultaneous visualization of the original PDF article and extracted text;
- Synchronization between the annotation area and the PDF visualizer area;
- Visualization of the annotations over the PDF article;
- Support for automatic annotations using Neji web services.

Importing a PDF article

The import of a PDF article follows the exact same procedure as the import of any article: a user clicks in the “Import” option, on the “Tools” menu, which opens a form where the user selects the PDF article format and the file to upload, and finally by hitting the “Import” button the file is uploaded to the server. Figure 4.20 illustrates the import of a PDF article.

Visualization of a PDF article

The visualization of a PDF article was implemented using a customized version of the PDF.js visualizer, which offers a set of features such as zoom in and zoom out, text search in the PDF article, page selection, text selection and page thumbnails view. When a PDF article is selected, the page is divided in two zones (see figure 4.21), the left zone is the annotation area, which contains the full extracted text from the PDF, and allows the manual curation of the annotations. The right zone contains the PDF visualizer. If the selected article is not a PDF, then only the annotation area is shown, occupying the whole web page (see figure 4.22).

Synchronization between annotation area and PDF visualizer

As stated in the section 3.3, some methods to allow the synchronization between the annotation

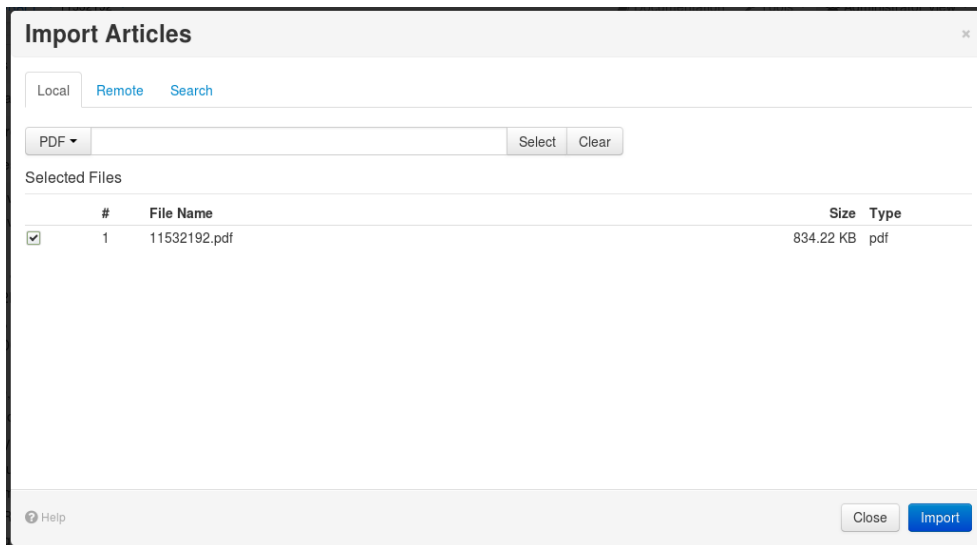


Figure 4.20: Example of importing a PDF article.

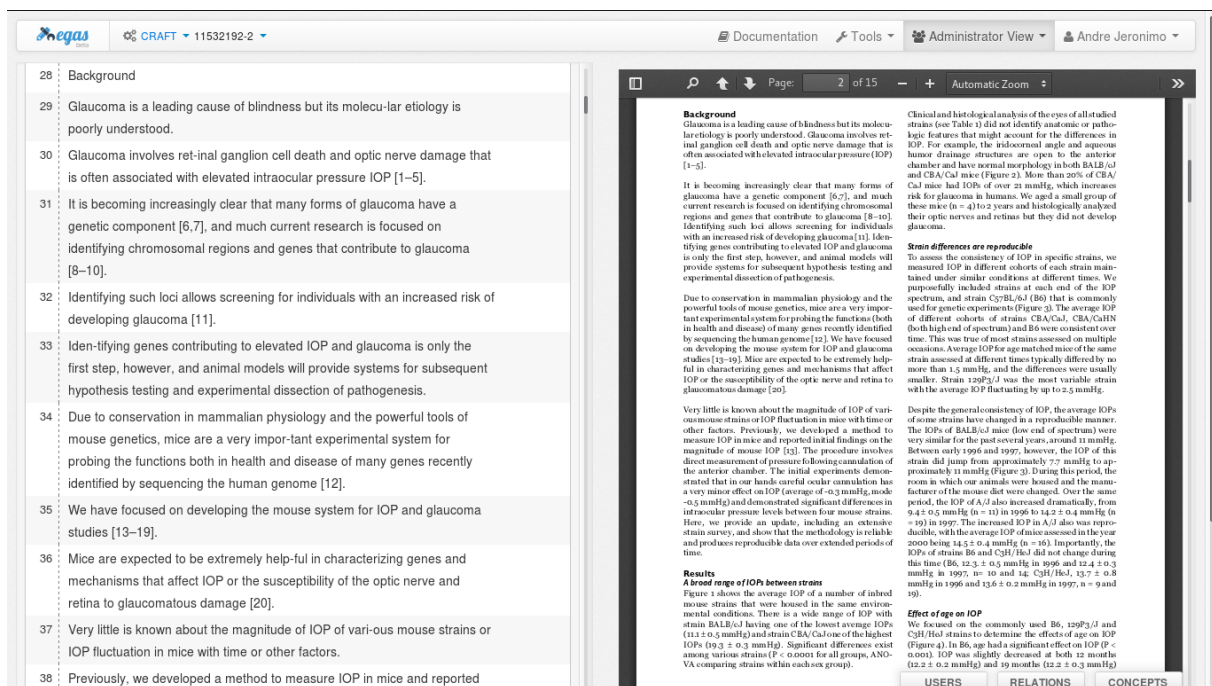


Figure 4.21: Visualization of a PDF article in Egas.

area and the PDF visualizer were implemented. When a user clicks in a line number of a sentence from the text area, the PDF visualizer automatically scrolls the PDF article to the correspondent sentence, and then performs a smooth highlight of that sentence. Figure 4.23 illustrates this behaviour.

On the other hand, when a user double clicks on a sentence in a PDF page, the annotation area also scrolls automatically to the corresponding sentence, and performs a smooth highlight of that sentence. Figure 4.24 illustrates this behaviour.

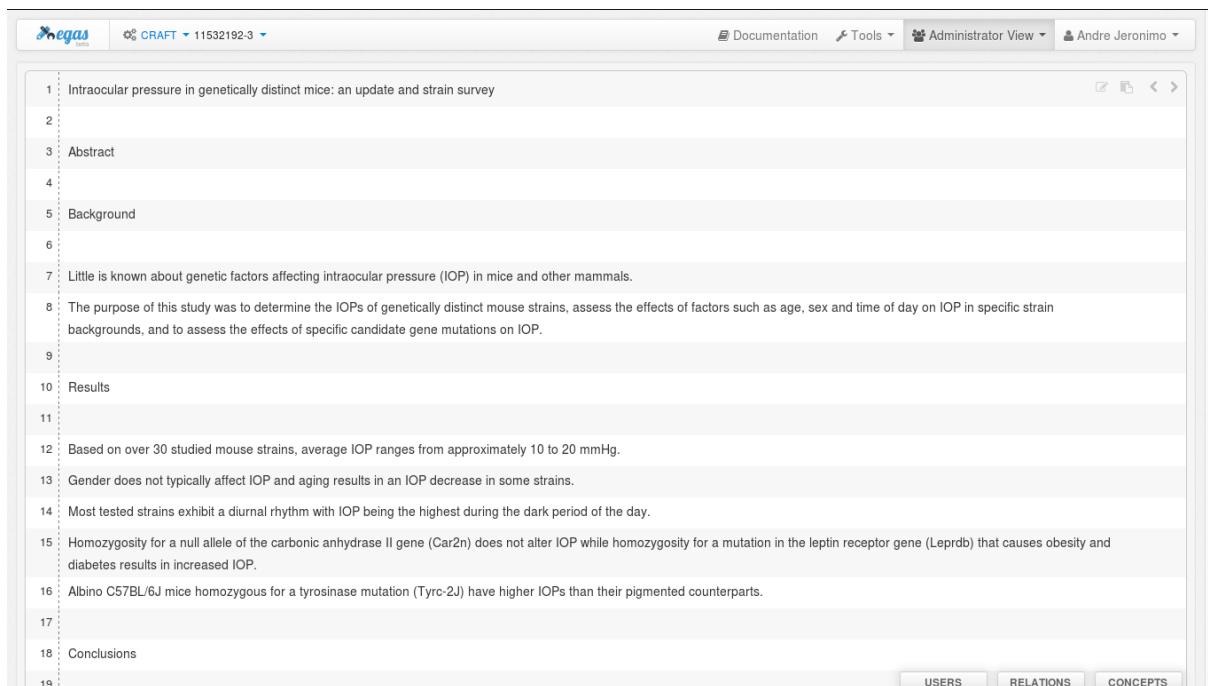


Figure 4.22: Visualization of a text article.

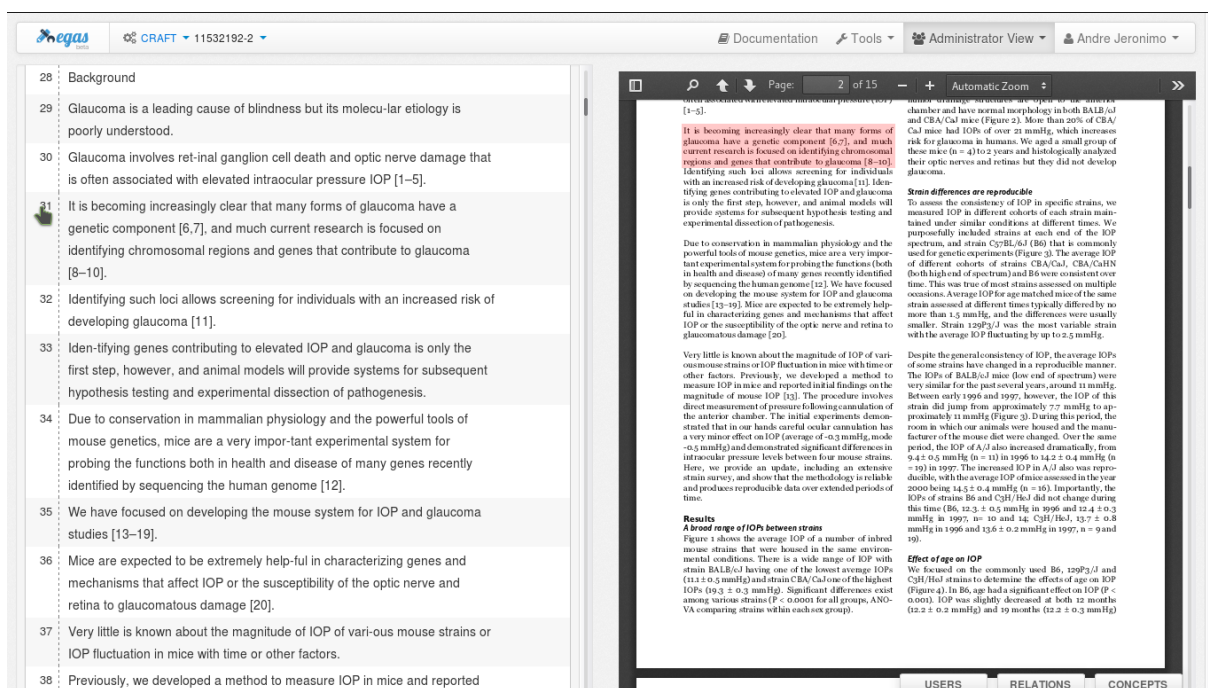


Figure 4.23: Synchronization between annotation area and PDF visualizer.

Associating a new annotation service to a project

The association of new annotation services from Neji web services to a project can only be made by an administrator. To associate a service an administrator needs to open the global administration panel by selecting the “Administration” option under his personal menu, and then select the “Services” tab. This tab presents a list of all projects and a list of all available annotation services, so the user

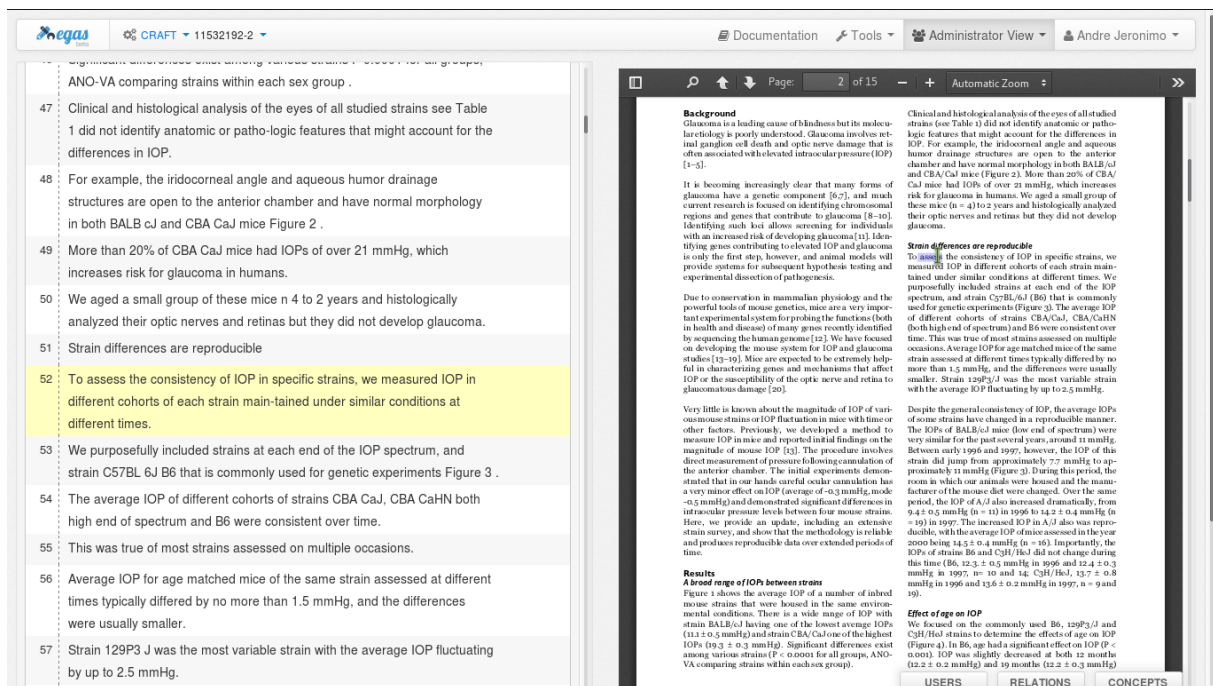


Figure 4.24: Synchronization between PDF visualizer and annotation area.

can select a project and choose one or more services to associate to it (see figure 4.25).

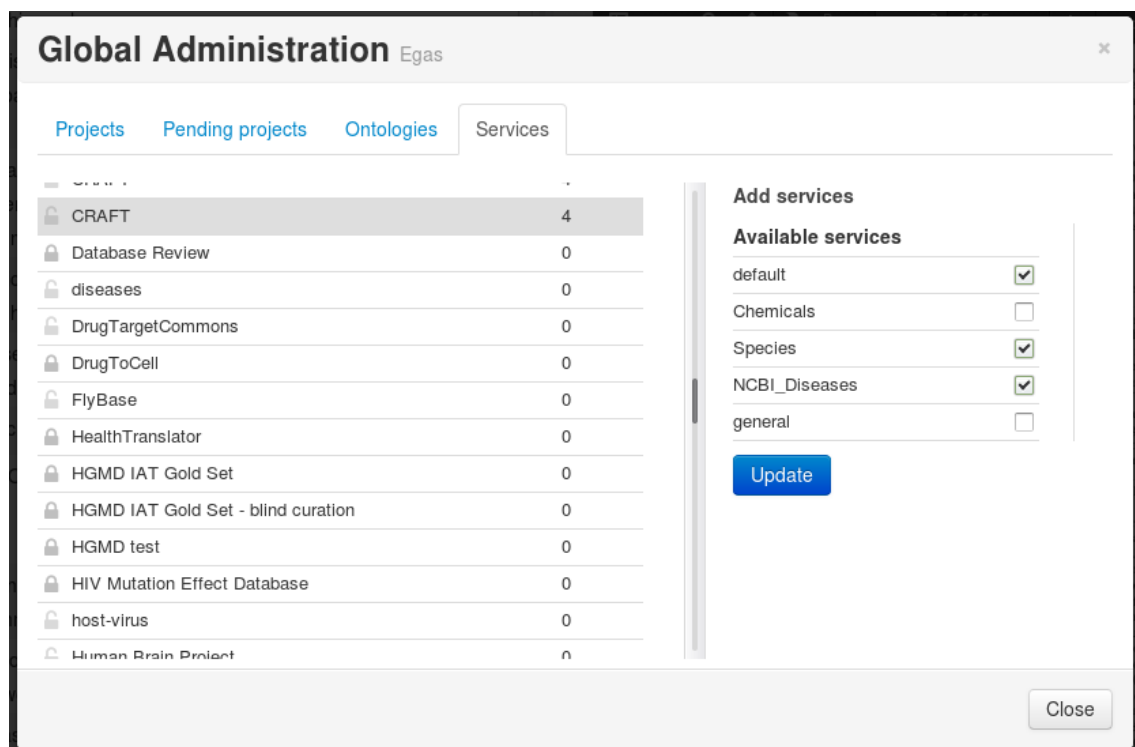


Figure 4.25: Adding a new annotation service to a project.

Annotation of an article using a new annotation service

To annotate an article using a new annotation service, a user selects the service he wants to use from the “Services” submenu, under “Tools” menu, and a form is open where a user can make a map between the service entity types and the project entity types (see figure 4.26). Finished the mapping, the user presses the “Annotate” button and the annotation is performed using the selected service. Figure 4.27 presents an example of the result of annotating an article.

Check	Entities to annotate	Corresponding concept type
<input checked="" type="checkbox"/>	Disorders	→ DISO
<input type="checkbox"/>	OCCU	→ Choose concept
<input checked="" type="checkbox"/>	PHEN	→ PHEN
<input type="checkbox"/>	OBJC	→ Choose concept
<input type="checkbox"/>	ACTI	→ Choose concept
<input checked="" type="checkbox"/>	LIVB	→ LIVB
<input type="checkbox"/>	DEVI	→ Choose concept
<input checked="" type="checkbox"/>	Chemicals	→ CHEM
<input type="checkbox"/>	Biomedical Processes	→ Choose concept
<input type="checkbox"/>	Genes and Proteins	→ Choose concept

Figure 4.26: Selecting a new annotation service.

Visualization of the annotations over the PDF

The annotations can also be seen directly over the PDF article. Figure 4.28 illustrates an example of this feature.

The automatic scroll feature and the visualization of annotations over the PDF can be enabled or disabled by the users at any time (see figure 4.29).

PDF VIEWER

- ☒ Automatic scroll
- ☒ Annotations in PDF

Figure 4.29: Enable and disable of PDF viewer features.

- Speed: how long it takes to process PDF documents in comparison to text documents.

To perform the evaluation, corpora, dictionaries and ML models were collected.

4.3.1 CORPORA

The evaluation of PDF processing was centered on the CRAFT corpus [52], one of the largest Gold Standard Corpora for the biomedical domain. It is composed of a set of 67 full-text articles, from 21 different journals, manually annotated by expert curators, with more than 21 thousand sentences and more than 560 thousand tokens. Overall, it contains almost 100 thousand annotations focused on nine biomedical ontologies and terminological resources: Cell Type Ontology (CL), Chemical Entities of Biological Interest ontology (ChEBI), NCBI Taxonomy (NCBITaxon), Protein Ontology (PRO), Sequence Ontology (SO), Entrez Gene database (EG) and three subontologies of the Gene Ontology, i.e, biological processes (BP), molecular functions (MF), and cellular components (CC).

This corpus contains a plain text version of the full-text of each article, that was derived from the original XML files distributed by the PubMed Central Open Access collection. All 67 articles were also published in PDF format and therefore all the original PDF articles were obtained. With the original PDF files and a plain text version of all articles, it is possible to evaluate the quality of the text extracted using Neji.

4.3.2 RESOURCES

In order to evaluate the processing speed of annotating the PDF files from the CRAFT corpus, a set of 34 dictionaries and one ML model were collected, to recognize biomedical concepts of various types: genes and proteins, chemicals, species, cells, cellular components, biological processes, molecular functions, disorders and anatomical entities.

4.3.3 PDF TEXT EXTRACTION EVALUATION

To evaluate the PDF text extraction, Neji was used to extract the text of the 67 PDF articles. Two different evaluation approaches were performed, in order to assess the quality of the extracted text from PDF documents:

- Exact matching: in this technique the sentences from the plain text versions of the articles are accepted only if they exist in the text extracted from the corresponding PDF article;
- Approximate matching: in this technique the extracted sentences from the PDF articles are accepted if they exist in the plain text version of the same article, or if that sentence is similar to a sentence that exists in the plain text version. A sentence is considered similar to another if the Levenshtein distance between them is lower than 10% of the length of the largest sentence. The Levenshtein distance between two sentences is the minimum number of character edits (insertions, deletions or substitutions) required to change one sentence into the other.

Table 4.7 presents the obtained results of both exact and approximate matching techniques.

Table 4.7: PDF text extraction evaluation using the Neji framework.

	Extraction quality
Exact matching	89.9%
Approximate matching	93.9%

Analysing the results from table 4.7 we can conclude that 89.9% of the text that exists in the plain text versions of the articles is identified and extracted by the Neji framework when using the PDF articles.

Additionally, the extracted text from PDF articles contains some text that are not included in the plain text versions, such as the headers and footers of the pages, the authors and other sections, that can have important data.

4.3.4 PDF PROCESSING SPEED EVALUATION

In order to evaluate the speed of processing and annotating PDF documents with Neji, the time of processing the plain text versions of the CRAFT corpus, and the time of processing the corresponding PDF articles, using the resources described in section 4.3.2, were measured. Additionally the time of just extracting the text from the PDF files, without performing the concepts recognition steps, was also measured.

The various processing speed experiments were performed in a machine with 12 processing cores @ 2.30 GHz and 192GB of RAM.

Table 4.8 presents the results obtained. Note that these results do not include the time needed to load all the resources, which is approximately one minute.

Table 4.8: PDF processing speed evaluation using the Neji framework.

Processing of plain texts		Processing of PDF articles		Text extraction from PDF articles		Annotation of PDF articles	
All articles	Per article	All articles	Per article	All articles	Per article	All articles	Per article
175 s	2.61 s	290 s	4.33 s	98 s	1.47 s	192 s	2.87 s

Analysing the results from table 4.8 we can verify that the processing of a plain text article takes on average 2.61 seconds, while the processing of a PDF article takes on average 4.33 seconds. However, the processing of a PDF article has the extra step of extracting the text from the PDF article, that takes on average 1.47 seconds to be performed. If the text extraction time is subtracted from the overall processing time of a PDF document, then on average it takes 2.87 seconds to be annotated, which is very close to the processing time of a plain text.

The same processing times were also calculated for the annotation web services provided by Neji. To perform this an annotation service was created in Neji web services with the same set of dictionaries and ML model used before. Table 4.9 presents the results obtained.

Table 4.9: PDF processing speed evaluation using the Neji web services.

Processing of plain texts		Processing of PDF articles		Text extraction from PDF articles		Annotation of PDF articles	
All articles	Per article	All articles	Per article	All articles	Per article	All articles	Per article
181 s	2.69 s	472 s	5.71 s	264 s	4.40 s	208 s	3.10 s

The processing of a plain text article using the web services takes on average 2.69 seconds, which is very close to the processing time of Neji (not taking into consideration the time required to load the resources in Neji).

The processing or text extraction from a PDF article using web services takes longer than using Neji because the PDF file needs to be uploaded to the server in order to be processed. But if we subtract the text extraction time (which includes also the upload time) from the overall processing time of a PDF document using web services, then on average a PDF takes 3.10 seconds to be annotated, which is also very close to the annotation time when using the Neji framework.

CONCLUSION

The main goal of this work was the study, evaluation, development and integration of several systems in order to deliver a solution that is able to automatically identify concept mentions in PDF scientific articles, using state-of-the-art annotation techniques, and include this feature in a simple and intuitive collaborative literature curation platform. Therefore, this work started with a detailed analysis and study of the existing solutions for PDF text extraction and biomedical concept recognition, in order to understand the current limitations and challenges of the available solutions. After analysing the state-of-the-art solutions and requirements, several systems were improved, developed and integrated in a final solution. As a result, improvements in Neji framework, Egas platform and the development of Neji web services, a web-based application for flexible and configurable automatic annotation of documents, are presented.

Our first contribution was made on Neji, a framework for biomedical concept recognition, extending it to allow the processing of PDF articles, by combining it with a high performance PDF text extractor library, and generating an adapted processing pipeline that outputs not just the recognized concepts, but also important data extracted from the PDF articles, such as sentences and annotations positions.

The second contribution ended up with the development of Neji web services, a web-services platform created with the objective of facilitating the access to Neji features by users and developers, through an intuitive and simple interface and a public set of web services for annotating documents in various formats. This platform provides an easy way to generate and configure different annotation services, which use different resources and techniques, depending on the user needs, and that can be updated and improved over the time. By providing annotation tools through web services that are constantly being executed on a server, the annotation processing time is reduced since resources do not need to be initialized.

The last contribution of the work presented in this thesis was the integration of PDF curation in Egas, a web-based platform for biomedical text mining and collaborative curation, allowing not just the processing of PDF articles, but also the visualization and interaction with them. Focusing on usability, the interface of Egas was divided in two main areas, allowing the side-by-side visualization of

both the original PDF article and the extracted text. A set of features was implemented to synchronize both areas and offer a better user experience. This way curator teams can use Egas directly with PDF articles and navigate between the annotations and the original file in an intuitive and easy way. Additionally, the annotation services provided by Neji web services were also integrated in Egas, offering the users more flexible services to automatically add annotations to documents, which they can then revise and refine.

Even though the presented contributions already provide a good set of features, the produced results, such as the public annotation services from Neji web services, can also be used by other investigators and developers as a base to develop new and innovative solutions and applications.

The developed systems present good performance results, both in terms of text extraction, processing speed and representation, which contributes also for a better user experience. In conclusion, the addressed contributions present several advantages for the biomedical community, allowing the direct annotation of PDF articles and simplifying the use, configuration and integration of annotation systems.

BIBLIOGRAPHY

- [1] M. Hilbert, P. López, C. Perez, Y. Ito, I. D. S. Pool, D. Sahal, S. Lloyd, G. E. Moore, X. Chen, M. Li, B. Ma, and J. Tromp, “The world’s technological capacity to store, communicate, and compute information.”, *Science (New York, N.Y.)*, vol. 332, no. 6025, pp. 60–5, Apr. 2011, ISSN: 1095-9203. DOI: 10.1126/science.1200970. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/21310967>.
- [2] K. Franzén, G. Eriksson, F. Olsson, L. Asker, P. Lidén, and J. Cöster, “Protein names and how to find them”, *International Journal of Medical Informatics*, vol. 67, no. 1, pp. 49–61, 2002, ISSN: 13865056. DOI: 10.1016/S1386-5056(02)00052-7. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1386505602000527>.
- [3] U.S. National Library of Medicine, *Detailed indexing statistics: 1965-2015*, 2016. [Online]. Available: http://www.nlm.nih.gov/bsd/index_stats_comp.html (visited on 05/31/2016).
- [4] J. Warnonck, “The camelot project”, 1991. [Online]. Available: <http://blogs.adobe.com/acrobat/files/2013/09/Camelot.pdf>.
- [5] A. M. Cohen and W. R. Hersh, “A survey of current work in biomedical text mining”, *Briefings in Bioinformatics*, vol. 6, no. 1, pp. 57–71, Mar. 2005, ISSN: 14675463. DOI: 10.1093/bib/6.1.57. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/15826357>.
- [6] A. Vlachos, “Semi-supervised learning for biomedical information extraction”, no. 791, 2010, ISSN: 1476-2986. [Online]. Available: <http://www.cl.cam.ac.uk/~mgk25/u../techreports/UCAM-CL-TR-791.pdf>.
- [7] W. J. Wilbur, A. Rzhetsky, and H. Shatkay, “New directions in biomedical text annotation: definitions, guidelines and corpus construction.”, *BMC bioinformatics*, vol. 7, p. 356, 2006, ISSN: 1471-2105. DOI: 10.1186/1471-2105-7-356. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/16867190>.
- [8] D. Swanson, “Medical literature as a potential source of new knowledge”, *Bulletin of the Medical Library Association*, vol. 78, no. 1, pp. 29–37, Jan. 1990. [Online]. Available: https://www.researchgate.net/publication/20751936_Medical_literature_as_a_source_of_new_knowledge.
- [9] R. A. DiGiacomo, J. M. Kremer, and D. M. Shah, “Fish-oil dietary supplementation in patients with raynaud’s phenomenon: a double-blind, controlled, prospective study.”, *The American journal of medicine*, vol. 86, no. 2, pp. 158–64, Feb. 1989, ISSN: 0002-9343. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/2536517>.
- [10] P. Schiapparelli, G. Allais, I. Castagnoli Gabellari, S. Rolando, M. G. Terzi, and C. Benedetto, “Non-pharmacological approach to migraine prophylaxis: part ii.”, *Neurological sciences : Official journal of the Italian Neurological Society and of the Italian Society of Clinical Neurophysiology*, vol. 31 Suppl 1, S137–9, Jun. 2010, ISSN: 1590-3478. DOI: 10.1007/s10072-010-0307-4. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/20464605>.

- [11] C. Martinez-Cruz, I. J. Blanco, and M. A. Vila, “Ontologies versus relational databases: are they so different? a comparison”, *Artificial Intelligence Review*, vol. 38, no. 4, pp. 271–290, Dec. 2012, ISSN: 0269-2821. DOI: 10.1007/s10462-011-9251-9. [Online]. Available: <http://link.springer.com/10.1007/s10462-011-9251-9>.
- [12] C. H. Wu, R. Apweiler, A. Bairoch, D. A. Natale, W. C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M. J. Martin, R. Mazumder, C. O’Donovan, N. Redaschi, and B. Suzek, “The universal protein resource (uniprot): an expanding universe of protein information.”, *Nucleic acids research*, vol. 34, no. Database issue, pp. D187–91, Jan. 2006, ISSN: 1362-4962. DOI: 10.1093/nar/gkj161. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/16381842>.
- [13] T. G. O. Gene Ontology Consortium, “Gene ontology consortium: going forward.”, *Nucleic acids research*, vol. 43, no. Database issue, pp. D1049–56, Jan. 2015, ISSN: 1362-4962. DOI: 10.1093/nar/gku1179. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/25428369>.
- [14] M. J. Schuemie, M. Weeber, B. J. A. Schijvenaars, E. M. van Mulligen, C. C. van der Eijk, R. Jelier, B. Mons, and J. A. Kors, “Distribution of information in biomedical abstracts and full-text publications.”, *Bioinformatics (Oxford, England)*, vol. 20, no. 16, pp. 2597–604, Nov. 2004, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bth291. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/15130936>.
- [15] R. Saetre, K. Yoshida, A. Yakushiji, Y. Miyao, Y. Matsubayashi, and T. Ohta, *Akana system: protein-protein interaction pairs in the biocreative 2 challenge, ppi-ips subtask*. 2007.
- [16] Y. Tsuruoka, Y. Tateishi, J.-D. Kim, T. Ohta, J. McNaught, S. Ananiadou, and J. Tsujii, “Developing a robust part-of-speech tagger for biomedical text”, in *Springer Berlin Heidelberg*, 2005, pp. 382–392. DOI: 10.1007/11573036_36. [Online]. Available: http://link.springer.com/10.1007/11573036_36.
- [17] H. Liu, T. Christiansen, W. A. Baumgartner, and K. Verspoor, “Biolemmatizer: a lemmatization tool for morphological processing of biomedical text.”, *Journal of biomedical semantics*, vol. 3, p. 3, 2012, ISSN: 2041-1480. DOI: 10.1186/2041-1480-3-3. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/22464129>.
- [18] K. Sagae and J. Tsujii, “Dependency parsing and domain adaptation with lr models and parser ensembles”, pp. 1044–1050, 2007.
- [19] Y. Miyao and J. Tsujii, “Feature forest models for probabilistic hpsg parsing”, *Computational Linguistics*, vol. 34, no. 1, pp. 35–80, Mar. 2008, ISSN: 0891-2017. DOI: 10.1162/coli.2008.34.1.35. [Online]. Available: <http://www.mitpressjournals.org/doi/abs/10.1162/coli.2008.34.1.35>.
- [20] D. Campos, “Term expansion methodologies in biomedical information retrieval”, *PhD Thesis, Universidade de Aveiro*, Feb. 2014.
- [21] *[table, stopwords] - pubmed help*, 2005. [Online]. Available: <http://www.ncbi.nlm.nih.gov/books/NBK3827/table/pubmedhelp.T.stopwords/> (visited on 06/01/2016).
- [22] G. Zhou, J. Zhang, J. Su, D. Shen, and C. Tan, “Recognizing names in biomedical texts: a machine learning approach.”, *Bioinformatics (Oxford, England)*, vol. 20, no. 7, pp. 1178–90, May 2004, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bth060. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/14871877>.
- [23] K. Fukuda, A. Tamura, T. Tsunoda, and T. Takagi, “Toward information extraction: identifying protein names from biological papers.”, *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pp. 707–18, 1998, ISSN: 2335-6936. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/9697224>.

- [24] R. Gaizauskas, G. Demetriou, P. J. Artymiuk, and P. Willett, "Protein structures and information extraction from biological texts: the pasta system.", *Bioinformatics (Oxford, England)*, vol. 19, no. 1, pp. 135–43, Jan. 2003, ISSN: 1367-4803. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/12499303>.
- [25] A. M. Cohen, *Unsupervised gene / protein named entity normalization using automatically extracted dictionaries*, 2005. DOI: 10.3115/1641484.1641487. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1641487>.
- [26] C. Funk, W. Baumgartner, B. Garcia, C. Roeder, M. Bada, K. B. Cohen, L. E. Hunter, and K. Verspoor, "Large-scale biomedical concept recognition: an evaluation of current automatic annotators and their parameters.", En, *BMC bioinformatics*, vol. 15, no. 1, p. 59, Jan. 2014, ISSN: 1471-2105. DOI: 10.1186/1471-2105-15-59. [Online]. Available: <http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-15-59>.
- [27] D. Campos, S. Matos, and J. L. Oliveira, "A modular framework for biomedical concept recognition.", En, *BMC bioinformatics*, vol. 14, no. 1, p. 281, Jan. 2013, ISSN: 1471-2105. DOI: 10.1186/1471-2105-14-281. [Online]. Available: <http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-14-281>.
- [28] D. Campos, S. Matos, and J. L. Oliveira, "Gimli: open source and high-performance biomedical name recognition.", En, *BMC bioinformatics*, vol. 14, no. 1, p. 54, Jan. 2013, ISSN: 1471-2105. DOI: 10.1186/1471-2105-14-54. [Online]. Available: <http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-14-54>.
- [29] A. R. Aronson and F.-M. Lang, "An overview of metamap: historical perspective and recent advances.", *Journal of the American Medical Informatics Association : JAMIA*, vol. 17, no. 3, pp. 229–36, Jan. 2010, ISSN: 1527-974X. DOI: 10.1136/jamia.2009.002733. [Online]. Available: <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2995713&tool=pmcentrez&rendertype=abstract>.
- [30] C. Jonquet, N. H. Shah, and M. A. Musen, "The open biomedical annotator.", *Summit on translational bioinformatics*, vol. 2009, pp. 56–60, Jan. 2009, ISSN: 2153-6430. [Online]. Available: <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3041576&tool=pmcentrez&rendertype=abstract>.
- [31] N. F. Noy, N. H. Shah, P. L. Whetzel, B. Dai, M. Dorf, N. Griffith, C. Jonquet, D. L. Rubin, M.-A. Storey, C. G. Chute, and M. A. Musen, "Bioportal: ontologies and integrated data resources at the click of a mouse.", *Nucleic acids research*, vol. 37, no. Web Server issue, W170–3, Jul. 2009, ISSN: 1362-4962. DOI: 10.1093/nar/gkp440. [Online]. Available: <http://nar.oxfordjournals.org/content/early/2009/05/29/nar.gkp440.short>.
- [32] I. S. Michael Tanenblatt Anni Coden, "I.I.: the conceptmapper approach to named entity recognition", 2010. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/citations;jsessionid=683BCA15A9FB2FD2C3C76A70E7E6C773?doi=10.1.1.453.2201>.
- [33] D. Rebholz-Schuhmann, M. Arregui, S. Gaudan, H. Kirsch, and A. Jimeno, "Text processing through web services: calling whatizit.", *Bioinformatics (Oxford, England)*, vol. 24, no. 2, pp. 296–8, Jan. 2008, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btm557. [Online]. Available: <http://bioinformatics.oxfordjournals.org/content/24/2/296.abstract>.
- [34] D. Campos, J. Lourenço, S. Matos, and J. L. Oliveira, "Egas: a collaborative and interactive document curation platform.", *Database : The journal of biological databases and curation*, vol. 2014, Jan. 2014, ISSN: 1758-0463. DOI: 10.1093/database/bau048. [Online]. Available: <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=4207226&tool=pmcentrez&rendertype=abstract>.
- [35] P. Stenetorp, S. Pyysalo, G. Topić, T. Ohta, S. Ananiadou, and J. Tsujii, "Brat: a web-based tool for nlp-assisted text annotation", pp. 102–107, Apr. 2012. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2380921.2380942>.

- [36] D. Salgado, M. Krallinger, M. Depaule, E. Drula, A. V. Tendulkar, F. Leitner, A. Valencia, and C. Marcelle, “Myminer: a web application for computer-assisted biocuration and text annotation.”, *Bioinformatics (Oxford, England)*, vol. 28, no. 17, pp. 2285–7, Sep. 2012, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/bts435. [Online]. Available: <http://bioinformatics.oxfordjournals.org/content/28/17/2285.long>.
- [37] C.-H. Wei, H.-Y. Kao, and Z. Lu, “Pubtator: a web-based text mining tool for assisting biocuration.”, *Nucleic acids research*, vol. 41, no. Web Server issue, W518–22, Jul. 2013, ISSN: 1362-4962. DOI: 10.1093/nar/gkt441. [Online]. Available: <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3692066&tool=pmcentrez&rendertype=abstract>.
- [38] M. Huang, J. Liu, and X. Zhu, “Genetukit: a software for document-level gene normalization.”, *Bioinformatics (Oxford, England)*, vol. 27, no. 7, pp. 1032–3, Apr. 2011, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btr042. [Online]. Available: <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3065680&tool=pmcentrez&rendertype=abstract>.
- [39] C.-H. Wei and H.-Y. Kao, “Cross-species gene normalization by species inference.”, En, *BMC bioinformatics*, vol. 12 Suppl 8, no. 8, S5, Jan. 2011, ISSN: 1471-2105. DOI: 10.1186/1471-2105-12-S8-S5. [Online]. Available: <http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-12-S8-S5>.
- [40] C.-H. Wei, H.-Y. Kao, and Z. Lu, “Sr4gn: a species recognition software tool for gene normalization.”, *PloS one*, vol. 7, no. 6, e38460, Jan. 2012, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0038460. [Online]. Available: <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3367953&tool=pmcentrez&rendertype=abstract>.
- [41] R. Leaman, R. Islamaj Dogan, and Z. Lu, “Dnorm: disease name normalization with pairwise learning to rank.”, *Bioinformatics (Oxford, England)*, vol. 29, no. 22, pp. 2909–17, Nov. 2013, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btt474. [Online]. Available: <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3810844&tool=pmcentrez&rendertype=abstract>.
- [42] C.-H. Wei, B. R. Harris, H.-Y. Kao, and Z. Lu, “Tmvar: a text mining approach for extracting sequence variants in biomedical literature.”, *Bioinformatics (Oxford, England)*, vol. 29, no. 11, pp. 1433–9, Jun. 2013, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btt156. [Online]. Available: <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3661051&tool=pmcentrez&rendertype=abstract>.
- [43] J. M. Cejuela, P. McQuilton, L. Ponting, S. J. Marygold, R. Stefancsik, G. H. Millburn, and B. Rost, “Tagtog: interactive and text-mining-assisted annotation of gene mentions in plos full-text articles.”, *Database : The journal of biological databases and curation*, vol. 2014, bau033, Jan. 2014, ISSN: 1758-0463. DOI: 10.1093/database/bau033. [Online]. Available: <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3978375&tool=pmcentrez&rendertype=abstract>.
- [44] Adobe Systems Incorporated, “Pdf reference, sixth edition: adobe portable document format version 1.7”, Nov. 2006. [Online]. Available: http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/pdf_reference_1-7.pdf.
- [45] B. Litchfield, *Making pdfs portable: integrating pdf and java technology / java iot*, 2005. [Online]. Available: <http://java.sys-con.com/node/48543> (visited on 01/30/2016).
- [46] C. Mattmann and J. Zitting, *Tika in Action*. Manning Publications Co., Dec. 2011, ISBN: 1935182854, 9781935182856. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2207980>.
- [47] *Pdfstream — pdf text, image, and form extraction for java and .net – snowtide*. [Online]. Available: <https://www.snowtide.com/> (visited on 01/30/2016).

- [48] C. Ramakrishnan, A. Patnia, E. Hovy, and G. A. Burns, “Layout-aware text extraction from full-text pdf of scientific articles.”, En, *Source code for biology and medicine*, vol. 7, no. 1, p. 7, Jan. 2012, ISSN: 1751-0473. DOI: 10.1186/1751-0473-7-7. [Online]. Available: <http://scfbm.biomedcentral.com/articles/10.1186/1751-0473-7-7>.
- [49] S. Miller, *Pdf2text. an efficient file conversion utility*, 2007. [Online]. Available: <http://www.articlesfactory.com/articles/computers/pdf2text-an-efficient-file-conversion-utility.html> (visited on 01/30/2016).
- [50] D. C. Comeau, R. T. Batista-Navarro, H.-J. Dai, R. I. Doğan, A. J. Yepes, R. Khare, Z. Lu, H. Marques, C. J. Mattingly, M. Neves, Y. Peng, R. Rak, F. Rinaldi, R. T.-H. Tsai, K. Verspoor, T. C. Wieggers, C. H. Wu, and W. J. Wilbur, “Bioc interoperability track overview.”, *Database : The journal of biological databases and curation*, vol. 2014, 2014, ISSN: 1758-0463. DOI: 10.1093/database/bau053. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/24980129>.
- [51] A. S. Schwartz and M. A. Hearst, “A simple algorithm for identifying abbreviation definitions in biomedical text”, *Pacific Symposium on Biocomputing*, pp. 451–62, 2003, ISSN: 2335-6936. DOI: 10.1.1.13.2481. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.2481>.
- [52] M. Bada, M. Eckert, D. Evans, K. Garcia, K. Shipley, D. Sitnikov, W. A. Baumgartner, K. B. Cohen, K. Verspoor, J. A. Blake, and L. E. Hunter, “Concept annotation in the craft corpus.”, *BMC bioinformatics*, vol. 13, no. 1, p. 161, 2012, ISSN: 1471-2105. DOI: 10.1186/1471-2105-13-161. [Online]. Available: <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3476437&tool=pmcentrez&rendertype=abstract>.