UNIVERSITÀ degli STUDI di CATANIA

Dipartimento di Fisica e Astronomia *"Ettore Majorana"*

CORSO DI DOTTORATO IN SISTEMI COMPLESSI PER LE SCIENZE FISICHE, SOCIO-ECONOMICHE E DELLA VITA

ANTONIO DI MARIA

BIOTAGME: A COMPREHENSIVE PLATFORM FOR BIOLOGICAL KNOWLEDGE NETWORK ANALYSIS

TESI DI DOTTORATO

RELATORI:
CHIAR.MO PROF. A. PULVIRENTI
DOTT. S. ALAIMO

ANNO ACCADEMICO 2019/2020

ABSTRACT

The inference of novel knowledge and the generation of new hypotheses from the analysis of the current literature is a fundamental process in making new scientific discoveries, and gaining knowledge about relations among biological elements. Especially in bio-medicine, given the enormous amount of literature and knowledge bases available, this process allows the researchers to obtain information on aspects already widely investigated by others researchers. The automatic extraction of information in the form of semantically related terms (or tags) is becoming an importance aspect (Kilicoglu et al., 2012; Stewart et al., 2012). Here we propose BioTAGME framework that combines TAGME annotation framework based on Wikipedia corpus (Ferragina and Scaiella, 2010), with the DT-Hybrid methodology. The aim of this combination is to extract biological terms from scientific documents available in PubMed, and predicts possible relationships among them in order to generate a knowledge graph. The proposed framework consist of two pipeline. The first one is entirely implemented in Spark in order to distribute the computing among several machines. It will be employed to annotate documents, and predict relationships among biological terms. Instead, the second one use the Laravel model-view-controller framework for the User Interface realization, and Neo4j graph database for storing the knowledge graph. In this thesis I also describe another project, called NetME, realized during the PhD. It allows to build an on-the-fly knowledge graphs starting from a subset of full texts obtained by a real-time query on PubMed and applying several semantic analysis methodologies.

Keywords: *Knowledge Graph*, *Document Annotation*, *Prediction Algorithms*, *Sentimental Analysis Methodologies*, *Apache SPARK*, *HDFS*, *Neo4j*.

ACKNOWLEDGEMENT

With immense pleasure and deep sense of gratitude, I wish to express my sincere thanks to **Prof. Alfredo Pulvirenti**, **Prof. Alfredo Ferro**, and the **Dr. Salvatore Alaimo**, without their motivation and continuous encouragement, this research would have been difficult to be completed.

I wish to extend my profound sense of gratitude to **my parents** for all the sacrifices they made during my research and also providing me with moral support and encouragement whenever required.

I am also pleased to thank the **Prof. Dennis Shasha**, and **Dr. Jacopo Cirrone** for motivating me to carry out research during the six months spent at the New York University, and also for providing me with infrastructural facilities and many other resources needed for my research..

Last but not the least, I would like to thank my colleagues for their constant encouragement, help, and moral support during the last three years of research activity.

Place: Catania

Date: 15/12/2020 **Antonio**

Contents

Chapter 1

Introduction

The increasing amount of scientific literature is rising new challenges for scientists. Identify the proper set of articles dealing with a certain topic could result as a not straightforward task and therefore the chance to miss important references and indeed relevant research is high.

In particular, in research areas as biology or biomedicine, thanks also to the fast-track publication journals, the number of published papers increases significantly fast. Furthermore, in such research areas, network analysis has became a key enabling technology to help the understanding of mechanisms of life, living organisms, and in general uncover the underlying fundamental biological processes. Example of applications include: i) analyzing the disease networks for identifying the disease-causing genes and pathways [1]; ii) discovering the functional interdependence among molecular mechanisms through functional network querying [2]; and iii) Network based inference with application on drug re-purposing [3].

The large number of open-access articles available in archive such as Pubmed Central [4], arxiv [5], and bioarxiv [6] as well as databases of ontologies which hold entities and their relations [7], is driving the academic community to rely on text mining tools and machine learning algorithms to digest corpus and extract semantic knowledge from them.

The text mining tools [8] and Natural Language Processing (NLP) [9] employ in-

formation extraction methods to translate unstructured textual knowledge in a form that can be easily analyzed and used to build functional network or more in general a biomedical knowledge graphs [10].This technology allows to infer putative relations among molecules such as: understand how a proteins interact each other; which gene mutations are involved in a disease, etc. In this context, the Biological Expression Language (BEL) [11] or Resource Description Framework (RDF) [12] is widely applied to convert a text in semantic triplets having the following form: <subject, predicate, object>.
The subject and object represent biological elements (i.e gene, disease, drug, enzyme, etc). Instead, the predicate represents a logical or physical relationship between them.
Since the implementation of biological text mining tools requires specialist skills in natural language processing that end-users usually do not have, several annotations ecosystem and tools have been implemented and made available to scientist community: j) PubAnnotation [13] is based on the "Agile text mining" concept, and it is a public resource for sharing annotated biomedical texts. jj) PubTator (PTC) [14] is a web service for viewing and retrieving bio-concept annotations (for genes/proteins, genetic variants, diseases, chemicals, species and cell lines) in full text biomedical articles. It annotates all the PubMed [4] abstract, and more than three millions of full texts. The annotations are downloadable in multiple formats (XML, JSON and tab delimited) via the online interface, a RESTful web service and bulk FTP. PTC is synchronized with PubMed and PubMed Central, with new articles added daily. Once the documents have been translated in a list of keywords through the annotation systems, we need to employ one or more prediction algorithms [15] and/or sentiment analysis techniques [16] for building relationships among the just extracted keywords.

Therefore, several frameworks to relations extraction from documents and knowledge network construction have been implemented in the past years: i) STRING [17], ii) Hetionet [18], iii) SemRep [19], iv) etc.

In this thesis we have implemented two frameworks based on the knowledge graph construction context. The first one is called BioTAGME (A comprehensive platform for biological knowledge network analysis) which is capable to extract knowledge from all documents' title and abstract in PubMed. It is composed of two pipelines:

- The first one has been designed to transform the PubMed documents in a knowledge network by applying TagME [20] as annotation technique, and DT-Hybrid algorithm together with relatedness computing method of TagME as prediction one. In addition, several external ontologies and databases i.e Disease Ontology [21], DrugBank [22], HGNC [23], etc, have been employed to mitigate the disambiguation error caused by an excessive generality of the Wikipedia corpus.

  This corpus is used by TagME to extract the annotation terms from a text. Such pipeline is entirely implemented in Spark [24] and Scala to ensure high scalability and distribution of the application.

- Instead, the second one has been designed to make available to the users a "Graphic interace", and several APIs needed for extracting information from the knowledge network, such as the echo network of a biological element, the shortest path between two biological ones, etc. It employs the Laravel model-view-control, in addition, all the views are implemented in React native.

The actual BioTAGME version is able to extract the relationships among biological elements, but no the relationship type due to the limited amount of information within the titles and abstracts. Therefor, we are already working on a second version of BioTAGME which includes full texts, titles, and abstracts in order to solve this issue.

The second frameworks explained in this thesis is NetME which allows to build an on-the-fly knowledge graphs starting from a subset of full texts obtained by a real-time query on PubMed. This framework orchestrates two different technologies which enable the inference of biological networks:

- A customized version of the entity-linker tagme [20], named “ontoTagme”, which is designed to run on a set of biological ontology knowledge-bases (such as GeneOntology [25], Drug-Bank [22],DisGeNET [26], and Obofoundry [27]) and allows to extract *biological entities* (i.e. genes, drugs, diseases) from a collection of full-text articles obtained through a query to PubMed. These biological entities will represent the nodes of the network;

- A software module developed upon the NLTK library [28] resource derives relations type and direction (edges) among the nodes previously extracted. Edges are weighted according to the frequency of appearance of their corresponding relation in the chosen list of full-text articles.

The thesis is organized as follows. The chapter two reports an overview based on already known annotation techniques, knowledge graph construction frameworks, and knowledge representation methods. The third one describes all the annotation and prediction algorithms (including our customization) used by BioTAGME and NetME for nodes and edges inference. The fourth and fifth ones provides the technical details of the back-end and the front-end of BioTAGME and NetME, respectively. We give also a preliminary experimental analysis. The conclusion one ends the thesis and sketches future research directions.

Chapter 2

Background

The concept of semantic networks has a long history (Quillian, 1968) and opened up a basis for knowledge modeling and representation (Helbig, 2006) by providing an adaptable formal framework for scientific developments and applications [29]. These networks allow to model semantic relationships in patterns of interconnected labeled nodes and edges incorporating linguistic information that describes concepts or objects. Semantic standards and technologies facilitate the combination of multiple biomedical association data across multiple domains, which can be highly heterogeneous, and enable the construction of knowledge networks composed of various biological entities, such as compounds, proteins, and genes [17].

Interactions among biological components is basically studied and analyzed through a rich spectrum of methods and metrics grounded on network theory. Most of these methods focus on the comparison of two biological networks (e.g., control vs. disease) [30]. To support challenging biomedical data integration efforts, a computational technique should satisfy the following three basic requirements [31]:

- An inter-operable data model that is capable of capturing and modeling the observed biomedical networks.
- A data integration framework to map and merge network data across disparate data sources.

- A collection of computational services to analyze, discover and validate new associations in integrated biomedical networks.

Based on these requirements, computer scientist and in general life-science researchers are focusing their studies and research topics on Annotation echo-systems and relations extraction frameworks.

In this chapter, we explain the concept of Document annotation including a brief overview of the most widely used tools and frameworks. Then, we give a description of the semantic languages have been established to represent and query semantic meaning of data and relationship. Finally, we conclude this chapter by listing a set of tools used to extract relationships and build knowledge networks.

2.1 Knowledge Graph

A Knowledge Graph (also known as semantic network) is a systematic way to connect information and data to knowledge. It represents a collection of interlinked descriptions of entities, real-world objects and events, or abstract concepts, obtained from several structured knowledge-bases such as ontologies (O_1, O_2, ..., O_k) [32]. An ontology is a formal description of knowledge as a set of concepts within a domain and the relationships that hold between them. To enable such a description, we need to formally specify components such as individuals (instances of objects), classes, attributes and relations as well as restrictions, rules and axioms. As a result, ontologies do not only introduce a shareable and reusable knowledge representation but can also add new knowledge about the domain [33]. They are considered as the data schema of the graph. In some cases, a knowledge graph can contains multiple ontologies and inter-relations between them.

A semantic network is defined as a graph G = (E,R) composed of entity $e \in E$ coming from terminologies or ontologies, and relations $r \in R$ that can be terminology/ies or ontology/ies relations. Both E and R are finite discrete spaces. In addition, every entity $e \in E$ can have some additional meta information defined with respect

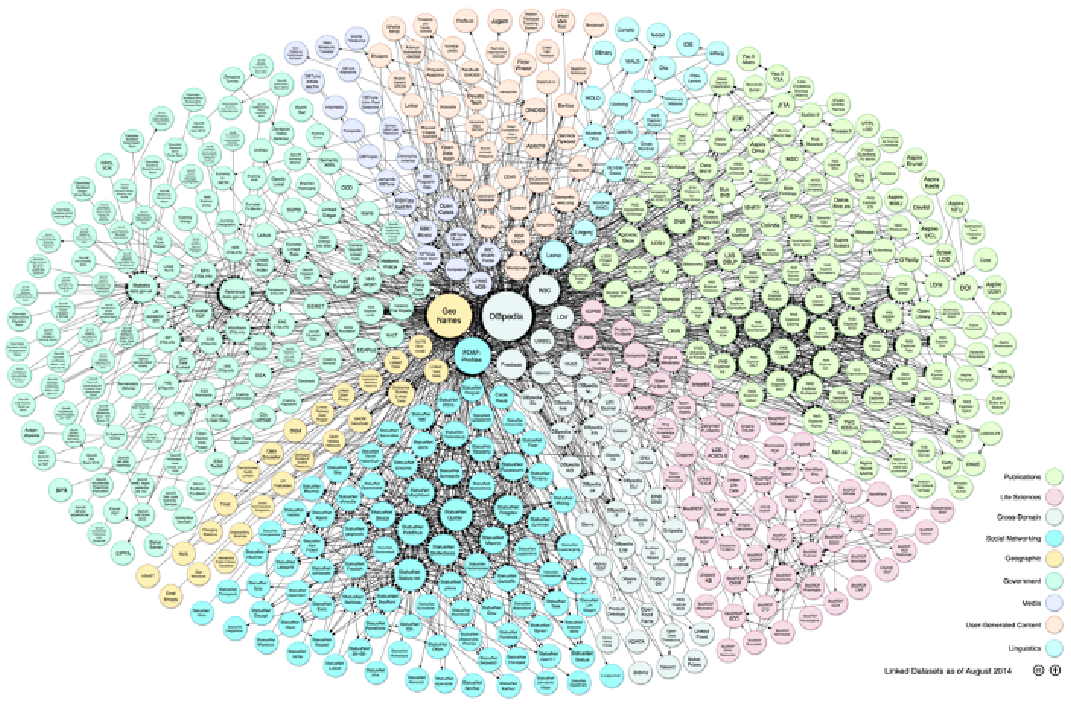


Figure 2.1: Linking Open Data cloud diagram 2014, by Max Schmatchtenberg, Cristian Bizer, Anja Jentzshc and Richard Cyganiak. http://lod-cloud.net/. The figure shows the DBPedia Knowledgw network.

to the application of the knowledge graph [34].

DBPedia [35] is an example of big knowledge graph. This project leverages the structure inherent in the info boxes of Wikipedia to create an enormous dataset of 4.58 things (link https://wiki.dbpedia.org/about) and an ontology that has encyclopedic coverage of entities such as people, places, films, books, organizations, species, diseases, etc.

2.2 Documents repositories

Thanks to the high availability of open-access articles repositories, and the continued growth of new entries and publications in the last few years, the research community has focused on text mining tools and machine learning algorithms to digest corpus and extract semantic knowledge. Due to the importance of these repositories, below is reported a brief explanation of the most commonly used documents repositories.

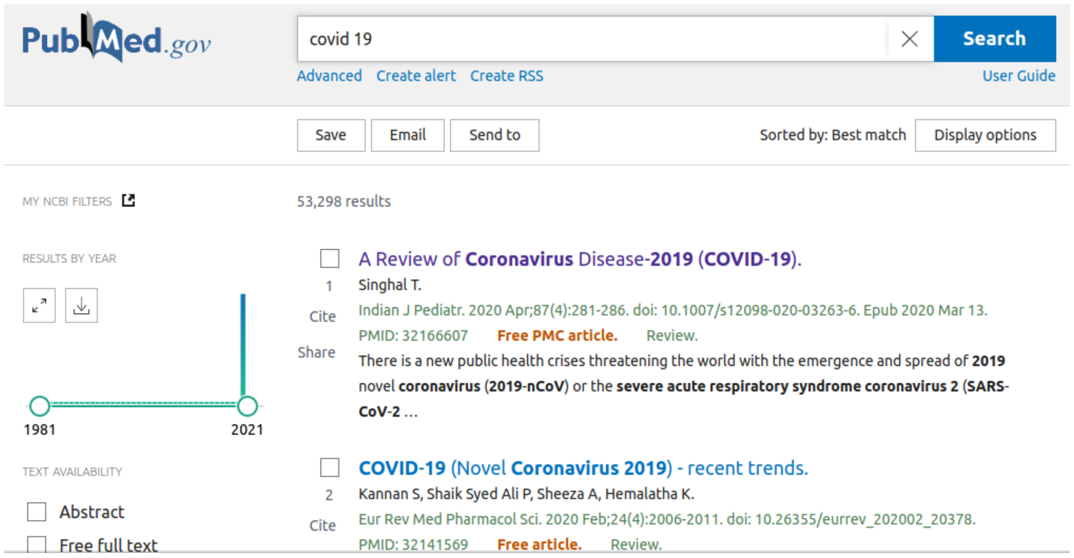
2.2.1 PubMed

PubMed [36] is a free resource supporting the search and retrieval of biomedical and life sciences literature with the aim of improving health–both globally and personally. The PubMed database contains more than 30 million citations and abstracts of biomedical literature. It does not comprise full-text journal articles. PubMed citations and abstracts include the fields of biomedicine and health, covering portions of the life sciences, behavioral sciences, chemical sciences, and bio-engineering. PubMed also provides access to additional relevant web sites and links to the other NCBI molecular biology resources. PubMed was developed since 1996, and is maintained by the National Center for Biotechnology Information (NCBI), at the U.S. National Library of Medicine (NLM), located at the National Institutes of Health (NIH). The search for based-topic publications can be carried out through two different modalities:

- User Interface over the website https://pubmed.ncbi.nlm.nih.gov/.(Fig 2.1)
- Entrez API (E-utilities) [37] allow access to all Entrez databases including PubMed. The E-utilities are a suite of eight server-side programs that accept a fixed URL syntax for search, link and retrieval operations. In BioTAGME we employ the E-utilities esearch and efetch functions during the updating of our local documents repository. The first one is used to download the PMIDs list published from mindata to maxdata. For example, to get all the new documents PMIDs from 2020/03/01 to 2020/04/01, we have to send the following GET request.

  “https://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=pubmed&

  retmode=json&retmin=2020/03/01&retmax=2020/04/01”



53,298 results

1 A Review of Coronavirus Disease-2019 (COVID-19).
Singhal T.
Indian J Pediatr. 2020 Apr;87(4):281-286. doi: 10.1007/s12098-020-03263-6. Epub 2020 Mar 13.
PMID: 32166607 Free PMC article. Review.
There is a new public health crises threatening the world with the emergence and spread of 2019 novel coronavirus (2019-nCoV) or the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2 ...

2 COVID-19 (Novel Coronavirus 2019) - recent trends.
Kannan S, Shaik Syed Ali P, Sheeza A, Hemalatha K.
Eur Rev Med Pharmacol Sci. 2020 Feb;24(4):2006-2011. doi: 10.26355/eurrev_202002_20378.
PMID: 32141569 Free article. Review.

Figure 2.2: Title and Abstract of the articles that treats COVID 19 disease

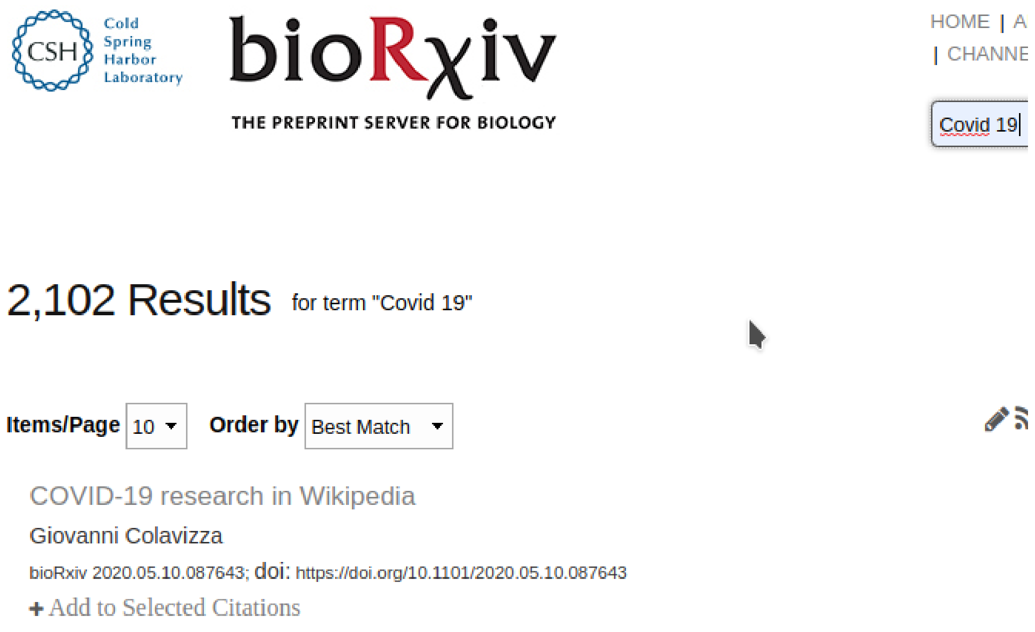
The second one is used to download title and abstract of the just downloaded PMIDs.

2.2.2 PubMed Central

PubMed Central [14] is the United States National Library of Medicine's digital archive of biomedical and life sciences journal literature which provides free access to the full text of articles. To search for free full-text articles on PubMed Central over the website "http://www.ncbi.nlm.nih.gov/pmc/", the searcher have to write topic/author/journal title. Just like PubMed, the NCBI's E-utilities can be used to download full-text by REST API.

2.2.3 bioRxiv

bioRxiv [6] is a free online archive and distribution service for unpublished pre-prints in the life sciences operated by Cold Spring Harbor Laboratory. The authors are able to to make their publications immediately available to the community (before the submitting to journals) by posting pre-prints on bioRxiv. The articles are no


Cold Spring Harbor Laboratory
bioRxiv
THE PREPRINT SERVER FOR BIOLOGY
HOME | A
| CHANNE
Covid 19

2,102 Results for term "Covid 19"

Items/Page 10 Order by Best Match

COVID-19 research in Wikipedia
Giovanni Colavizza
bioRxiv 2020.05.10.087643; doi: https://doi.org/10.1101/2020.05.10.087643
+ Add to Selected Citations

Figure 2.3: All the covid 19-based pre-print submitted to bioRxiv

reviewed before being posted online. However, these are only checked for avoiding offensive and/or non-scientific content. An article may be posted prior to, or concurrently with, submission to a journal but should not be posted if it has already been accepted for publication by a journal. In addition it is categorized as New Results, Confirmatory Results, or Contradictory Results.

- New Results describe an advance in a field.
- Confirmatory Results largely replicate and confirm previously published work
- Contradictory Results largely replicate experimental approaches used in previously published work but the results contradict and/or do not support it.

Readers may comment the articles on bioRxiv and also contact authors directly to get clarification. Pre-prints deposited in bioRxiv can be cited using their digital object identifier (DOI). Such object identifier is the same for every version of the manuscript. Therefore, if a user needs to specific a version, she has to add the version-specific URL: “https://www.biorxiv.org/content/10.1101/2019.12.11.123456v2”.

2.3 Articles annotation and Relationships extraction

Entity and relation extraction is a standard task in text mining or natural language processing (NLP) necessary to transfer unstructured textual information into a structured form (more information about these forms are reported in the next section). In this context, entities are typically short phrases representing a specific object such as 'pancreatic neoplasms'. Instead, the relations are physical or logical interactions among the entities of the considered domain (biological in this thesis work) [38].

A common idea of the computational linguistics approaches is that the semantics of a relation expressed in a piece of text can be extracted through the syntactic analysis of that text. Thus, given the sentence "A woman who was treated for thyrotoxicosis disease with methimazole drug developed agranulocytosis disease.", the aim of an annotator is to extract the entities methimazole (drug) and agranulocytosis (disease) as well as the relationship between these. The syntactic structure of a sentence is often represented as a parse tree capturing the interaction of phrasal constituents within the formalism of Dependency Grammar.

Various approaches for relation extraction have been applied to the biomedical domain:

- The simplest approach is the detection of **co-occurrences** of entities from sentences or abstracts relying on clustering procedures. The central hypothesis is that similar entities tend to occur together in similar contexts. Generally, the type and direction of the relation cannot be determined [39].

- **Pattern-based extraction** approaches to infer lexico-syntactic patterns. The main drawback is that this techniques does not face the high variability of how a relation can be expressed in natural language. It generally extracts single-word terms rather than well-formed and compound concepts [40].

- **Machine Learning** [41] classifier for associating specific category to an input text without using any advanced Natural Language Processing technique on the sentences. Since this approach relies on a standard supervised classification strategy, several algorithms like Naive Bayes Classifiers [42], Decision Trees [43] and Support Vector Machines [44], are usually employed. For example, a vector-based approach is used to transform a span of text and candidate relations into a numerical vectors used during the classification procedure. Such models suffer a main problems: The errors generated in the entities extraction step may propagate to the step of relation classification. For instance, if a drug or disease entity mention is incorrectly recognized, than the extraction of its related relationships will be incorrect.

- **Deep learning** [45] with neural networks has received increasing research attention in the text mining and NLP areas. Compared with other approaches, deep neural networks adopt low-dimensional dense embedding [46] to denote features such as words or part-of-speech (POS) tags [47], which can effectively settle the feature sparsity problem. In addition, deep neural networks demand less feature engineering, since they can learn features from training data automatically. For relation classification, deep neural networks (RNNs) [48] or convolutional neural networks (CNNs) [49] are used to learn relation representations along the words between two target entities.

Relation extraction algorithms can also be classified by the way the extraction rules are obtained, they can be: **i**. manually defined or **ii**. learned from large annotated training corpora.

Below we describe some frameworks and tools designed to entities and relationships extraction.

2.3.1 PubAnnotation

PubAnnotation [13] is an ecosystem based on agile approaches to text mining and annotation procedures. It is composed of the following three components:

- An annotation model composed of three sub-components: i. Storage, ii. Accessors. and iii. Processors

- A dictionary-based annotator.

- A storage component facilitates regression testing.

To facilitate the communication among the components, two models have been created: **Passive communication** model allows processors or accessors to get annotation stored in the annotation server, or to push new and revised ones to a server. **Active communication** model enables an annotation storage for initializing communication to "actively" obtain new or update annotations.

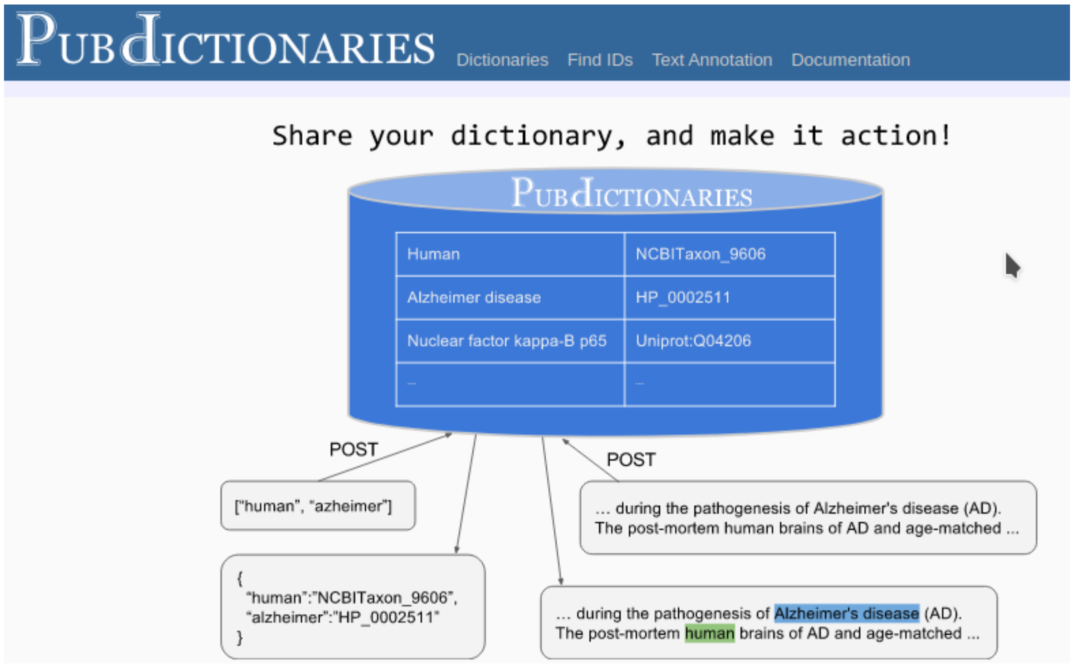
Based on these two communication models, several components have been developed within the ecosystem: PubDictionaries, PubAnnotation and TextAE

- **PubDictionaries** (http://pubdictionaries.org) is a processor to enable customized dictionary-based annotation. It provides the following functionalities: i) A repository of dictionaries, ii)An interface to test dictionary-based annotation, iii) A set of programmable RESTful APIs for scalable annotation, iv)An interface to enable edition of dictionaries. In addition, a user can easily add new entries to or delete from a dictionary. Note that, a dictionary means a list of couples composed of labels and identifiers.

  $$dictionary_i = (l_1, i_1), ..., (l_k, i_k) \tag{2.1}$$

  where labels mean term expressions that have to appear in natural language text as refer to specific entities, and identifiers are structured information sources (e.g. databases or ontologies.

- **PubAnnotation** (http://pubannotation.org) enables an user to annotate a biological text through the dictionaries previously explained. Thus, at the beginning it is necessary to select the PubDictionary processors (dictionaries), and



Human	NCBITaxon_9606
Alzheimer disease	HP_0002511
Nuclear factor kappa-B p65	Uniprot:Q04206
...	...

["human", "azheimer"]

```
{
 "human":"NCBITaxon_9606",
 "alzheimer":"HP_0002511"
}
```

... during the pathogenesis of Alzheimer's disease (AD). The post-mortem human brains of AD and age-matched ...

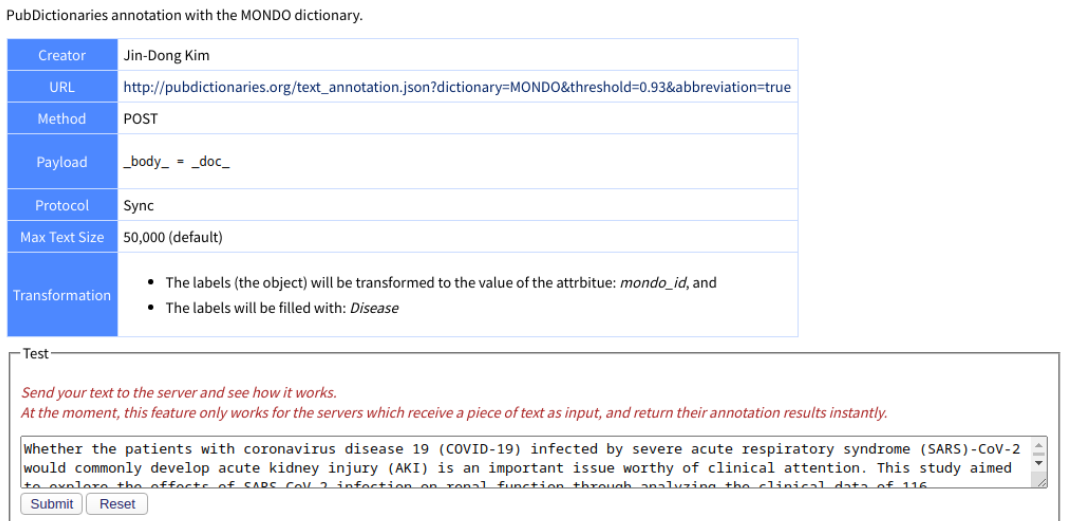
... during the pathogenesis of Alzheimer's disease (AD). The post-mortem human brains of AD and age-matched ...

Figure 2.4: PubDictionary user interface that allows to upload dictionaries, modify terms and concept of the dictionary, edit dictionaries, etc

the text to annotate. For example, if the selected text treats COVID 19 disease, the PubDictionary processors to select are: "MONDO_0100096", "COVID 19 samples", etc. Next, the user sends the request and obtains a list of annotations extracted by the text. Finally, the user may evaluate the quality of annotations in two different ways: using a benchmark dataset registered to PubAnnotation as a project and compared with any other project, or manual inspection.

If the quality is not satisfactory, an improvement may be implemented either by revising the dictionary, or by revising the annotation. Therefore, if there are missing or noisy entries within the dictionaries which have been identified during the evaluation, the user may add o remove them, respectively. In this case, the improvement may be replicated to future (dictionary-based) annotations.

Since PubAnnotation works as a public repository of annotations, as well as a platform for annotation development, the annotations developed on PubAn-


PubDictionaries annotation with the MONDO dictionary.

Creator	Jin-Dong Kim
URL	http://pubdictionaries.org/text_annotation.json?dictionary=MONDO&threshold=0.93&abbreviation=true
Method	POST
Payload	_body_ = _doc_
Protocol	Sync
Max Text Size	50,000 (default)
Transformation	• The labels (the object) will be transformed to the value of the attrbitue: *mondo_id*, and • The labels will be filled with: *Disease*

Test

Send your text to the server and see how it works.
At the moment, this feature only works for the servers which receive a piece of text as input, and return their annotation results instantly.

```
Whether the patients with coronavirus disease 19 (COVID-19) infected by severe acute respiratory syndrome (SARS)-CoV-2
would commonly develop acute kidney injury (AKI) is an important issue worthy of clinical attention. This study aimed
```

Figure 2.5: An example of text annotation using the “mondo” dictionary, and send a text that treats COVID 19 disease

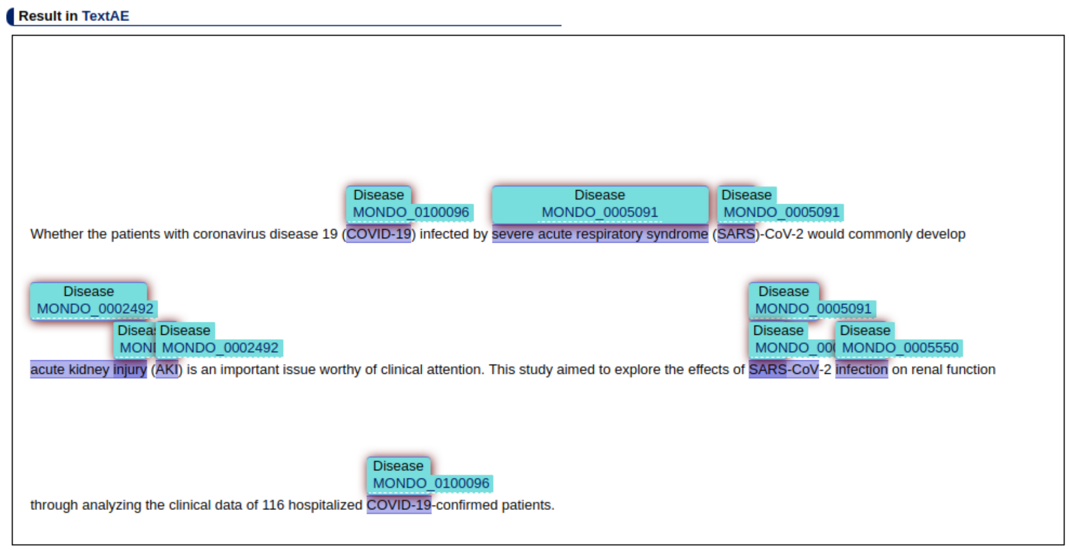
notation can be shared.

2.3.2 PubTatorCentral

PubCentral [14] provides automated concept annotations of several biomedical concept types (genes/proteins, genetic variants, diseases, chemicals, and species) across all PubMed article abstracts, and PMC full-text articles. To ensure high quality, the framework uses tools such as GeneTUKit [50] for gene mention, GenNorm [51] for gene normalization, SR4GN [52] for species, DNorm for diseases [53], tmVar [54] for mutations, and a dictionary-based lookup approach [55] for chemicals. SR4GN was also used for associating recognized species with their corresponding gene/protein mentions so that we were able to perform cross-species gene normalization in PubTator.

Five search options are currently available in PubTator:

- **PubMed** return results identical to PubMed search results
- **Gene** return all articles relevant to a specific gene or gene product
- **Chemical** return all articles relevant to a specific chemical


Result in TextAE

Whether the patients with coronavirus disease 19 (COVID-19) infected by severe acute respiratory syndrome (SARS)-CoV-2 would commonly develop acute kidney injury (AKI) is an important issue worthy of clinical attention. This study aimed to explore the effects of SARS-CoV-2 infection on renal function through analyzing the clinical data of 116 hospitalized COVID-19-confirmed patients.

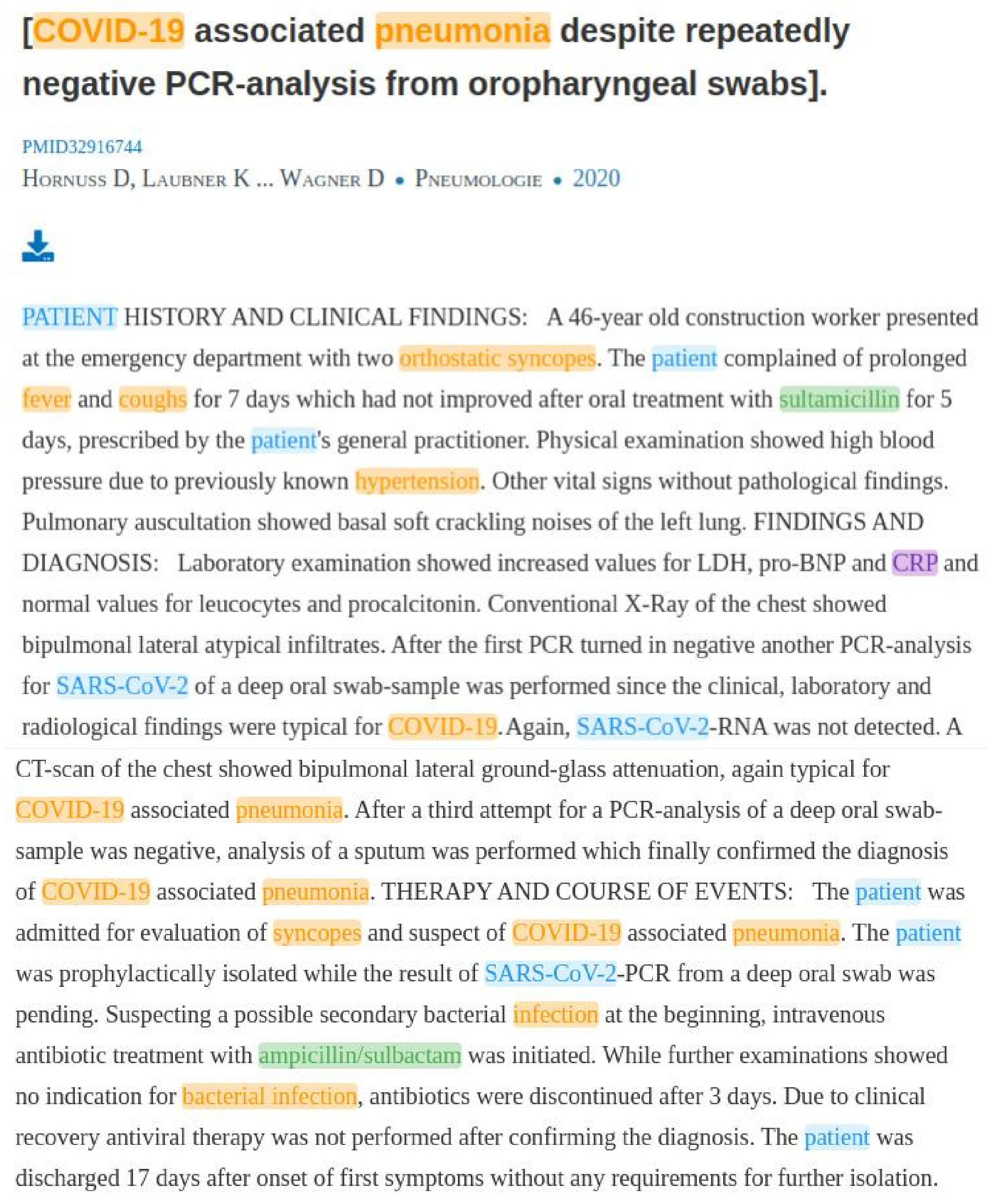
Figure 2.6: List of annotations extracted from the text submitted in Figure 2.5 and shown through the TextAE editor

- **Disease** return all articles relevant to a specific disease or syndrome
- **PMID** List: return articles in the PubMed Identifier (PMID)

PubTator supports three annotation tasks: document triage, entity annotation and relationship annotation. In document triage, biocurators are used to select articles. As a pre-step for full curation, users can identify articles in two simple mechanisms: First, a user can select articles from the search results. Second, a user can indicate whether an article is curatable at the top of the annotation page.

PubTator can be used to provide automated pre-annotations for five common types (listed above) marked with different colours. A user can modify and remove an existing annotation as well as insert a new one.

Finally, PubTator can be used for annotating relationships between entities. This allows curators to specify the kind of relations they desire to capture from literature, which can be either between the same kind of entities, such as protein–protein interactions, or between different kinds, such as gene–disease relations.



[COVID-19 associated pneumonia despite repeatedly negative PCR-analysis from oropharyngeal swabs].

PMID32916744

Hornuss D, Laubner K ... Wagner D • Pneumologie • 2020

PATIENT HISTORY AND CLINICAL FINDINGS: A 46-year old construction worker presented at the emergency department with two orthostatic syncopes. The patient complained of prolonged fever and coughs for 7 days which had not improved after oral treatment with sultamicillin for 5 days, prescribed by the patient's general practitioner. Physical examination showed high blood pressure due to previously known hypertension. Other vital signs without pathological findings. Pulmonary auscultation showed basal soft crackling noises of the left lung. FINDINGS AND DIAGNOSIS: Laboratory examination showed increased values for LDH, pro-BNP and CRP and normal values for leucocytes and procalcitonin. Conventional X-Ray of the chest showed bipulmonal lateral atypical infiltrates. After the first PCR turned in negative another PCR-analysis for SARS-CoV-2 of a deep oral swab-sample was performed since the clinical, laboratory and radiological findings were typical for COVID-19.Again, SARS-CoV-2-RNA was not detected. A CT-scan of the chest showed bipulmonal lateral ground-glass attenuation, again typical for COVID-19 associated pneumonia. After a third attempt for a PCR-analysis of a deep oral swab-sample was negative, analysis of a sputum was performed which finally confirmed the diagnosis of COVID-19 associated pneumonia. THERAPY AND COURSE OF EVENTS: The patient was admitted for evaluation of syncopes and suspect of COVID-19 associated pneumonia. The patient was prophylactically isolated while the result of SARS-CoV-2-PCR from a deep oral swab was pending. Suspecting a possible secondary bacterial infection at the beginning, intravenous antibiotic treatment with ampicillin/sulbactam was initiated. While further examinations showed no indication for bacterial infection, antibiotics were discontinued after 3 days. Due to clinical recovery antiviral therapy was not performed after confirming the diagnosis. The patient was discharged 17 days after onset of first symptoms without any requirements for further isolation.

Figure 2.7: Text Annotation through PubTator. The annotation terms are marked with different colors.

2.3.3 SemRep

SemRep [19] is a natural language processing system designed to recover semantic propositions from biomedical text using underspecified syntactic analysis and structured domain knowledge from the UMLS [56]. It extracts relationships from biomedical sentences in PubMed titles and abstracts by mapping textual content to an ontology which represents the meaning.

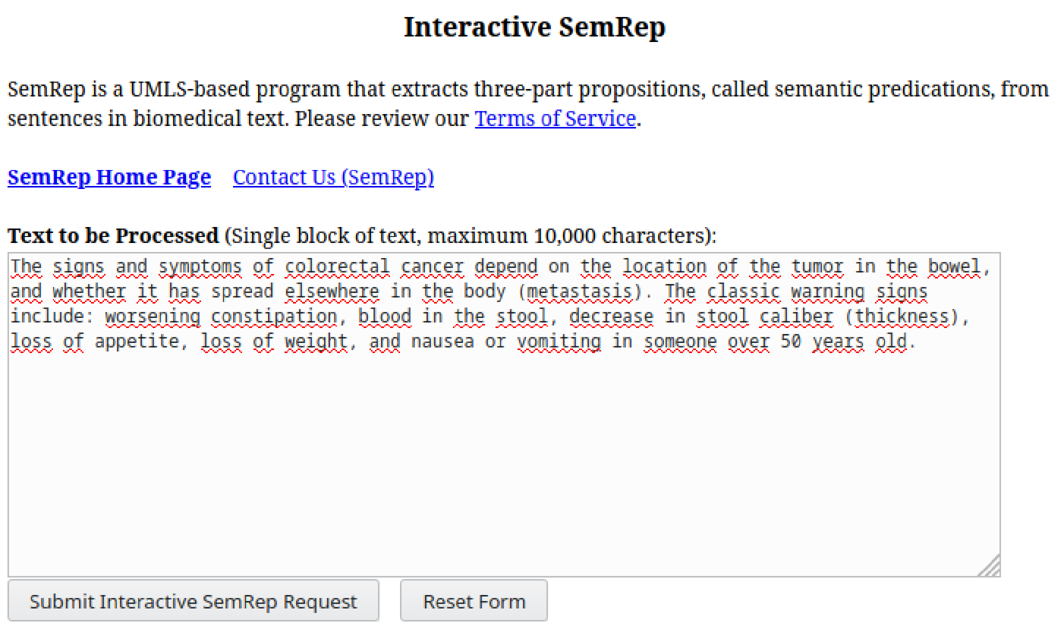
After the selection and tokenization of the text is completed, it is submitted to an under-specified parser that relies on the syntactic information in the SPECIALIST Lexicon [57]. Part-of-speech ambiguities are resolved with the Xerox Part-of-Speech Tagger [58].

Basically, simple noun phrases are identified because these contain useful information about terms relationships. Prepositional phrases are treated as noun phrases whose first element is a preposition. Instead, other syntactic categories, including verbs, auxiliaries, and conjunctions are expressed through their part-of-speech label and put into a separate phrase. In order to show the returned annotations in a compact way, the semantic types are abbreviate. Therefore, "Diseases" or "Sndromes" are referred as "dsyn", "OrganicChemical" with (orch), etc.

The domain knowledge is obtained through MetaMap [59], a knowledge-based application that uses the SPECIALIST Lexicon and several rules to determine the best mapping between the text of a noun phrase and a concept. Thus, interpretation of semantic propositions depends on the underspecified syntactic analysis enriched with domain knowledge.

The rules adopted allow to map syntactic indicators such as verbs, prepositions, and nominalization, to predicates in the Semantic Network. For example, exist a rule that links the 'nominalization treatment' with the predicate TREATS. Further syntactic constraints on argument identification are controlled by statements expressed in a dependency grammar.

SemRep also addresses noun phrase coordination [60] by taking advantage of se-


Interactive SemRep

SemRep is a UMLS-based program that extracts three-part propositions, called semantic predications, from sentences in biomedical text. Please review our Terms of Service.

SemRep Home Page Contact Us (SemRep)

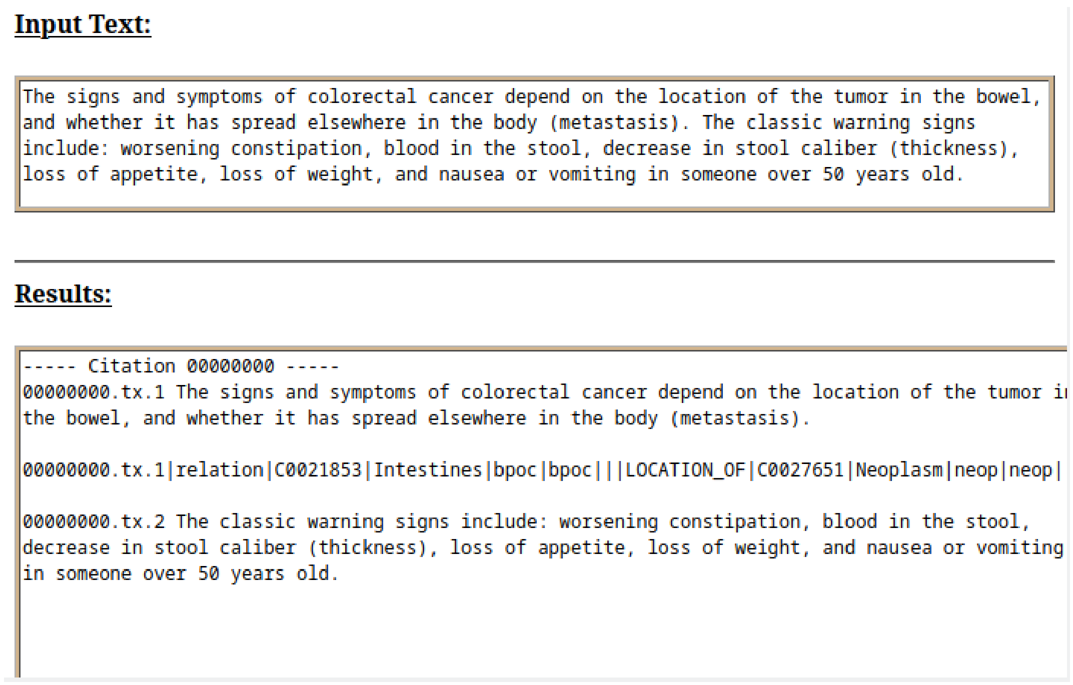
Text to be Processed (Single block of text, maximum 10,000 characters):

```
The signs and symptoms of colorectal cancer depend on the location of the tumor in the bowel,
and whether it has spread elsewhere in the body (metastasis). The classic warning signs
include: worsening constipation, blood in the stool, decrease in stool caliber (thickness),
loss of appetite, loss of weight, and nausea or vomiting in someone over 50 years old.
```

Figure 2.8: Example of text submitting in SemRep User Interface

mantic types. Thus, on the basis of the under-specified syntax enhanced with domain knowledge, it is necessary to determine whether each coordinator is conjoining noun phrases or something other than noun phrases. For a coordinator marked as conjoining noun phrases, the semantic type of the noun phrase immediately to the right of that coordinator is examined. Then, the noun phrase immediately to the left of the coordinator, and noun phrases occurring to the left of that noun phrase (separated by another coordinator or a comma) are examined to understand whether they are semantically consonant.

Terms that share same meaning are classed under the same concept. Instead, each concept is assigned a semantic type, such as "Disease or Syndrome" or "Gene or Genome", so that similar concepts are clustered under the same semantic type. Once all text sentences are evaluated, the framework returns several tuples per each sentence that contain a subject, an object, and the relation that links them [61].


Input Text:

```
The signs and symptoms of colorectal cancer depend on the location of the tumor in the bowel,
and whether it has spread elsewhere in the body (metastasis). The classic warning signs
include: worsening constipation, blood in the stool, decrease in stool caliber (thickness),
loss of appetite, loss of weight, and nausea or vomiting in someone over 50 years old.
```

Results:

```
----- Citation 00000000 -----
00000000.tx.1 The signs and symptoms of colorectal cancer depend on the location of the tumor i
the bowel, and whether it has spread elsewhere in the body (metastasis).

00000000.tx.1|relation|C0021853|Intestines|bpoc|bpoc|||LOCATION_OF|C0027651|Neoplasm|neop|neop|

00000000.tx.2 The classic warning signs include: worsening constipation, blood in the stool,
decrease in stool caliber (thickness), loss of appetite, loss of weight, and nausea or vomiting
in someone over 50 years old.
```

Figure 2.9: SemRep results extracted from the Input text. Only a single relationship has been obtained: $< subject : Intestines, type\_rel : LOCATION\_OF, object : Neoplasm >$

2.3.4 STRING

The STRING database [17] aims to collect and integrate functional interactions between the expressed proteins for a large number of organisms. The associations in STRING include direct (physical) interactions, as well as indirect (functional) ones. Regarding interaction predictions, these are derived from several sources:

- systematic co-expression analysis.
- detection of shared selective signals across genomes.
- automated text-mining of the scientific literature.
- computational transfer of interaction knowledge between organisms based on gene orthology.

For each protein–protein association stored in STRING, a score is provided. These scores indicate the estimated likelihood that a given interaction is biologically meaningful, specific and reproducible, given the supporting evidence. This last is divided into one or more 'evidence channel' (delineated by edges of different colors in the user interface) on the basis of the origin and type of the evidence. An interaction is good when the score is high, but at the same time, there is also more than one evidence channel contributing to the score. The “evidence channels” in STRING are seven:

- Experiments channel: the evidence comes from actual experiments in the lab extracted by databases organized in the IMEx consortium [62], plus BioGRID [63].
- Database channel: the evidence is asserted by a human expert curator. This information is imported from pathway databases.
- Textmining channel: the evidence comes from abstract or even sentence into fulltext articles.
- Coexpression channel: the evidence is extracted by gene expression data. Pairs of proteins that are consistently similar in their expression patterns receive an high association score.
- Neighborhood channel: the evidence is inferred considering genes and other's genome neighborhood (genome-based prediction).
- Fusion channel: Pairs of proteins are given an association score when there is at least one organism where their respective orthologs have fused into a single, protein-coding gene.
- Co-occurrence channel: In this channel, STRING evaluates the phylogenetic distribution of orthologs of all proteins in a given organism [64]. If two proteins show a high similarity in this distribution, i.e. if their orthologs tend to be observed as “present” or “absent” in the same subsets of organisms, then an association score is assigned

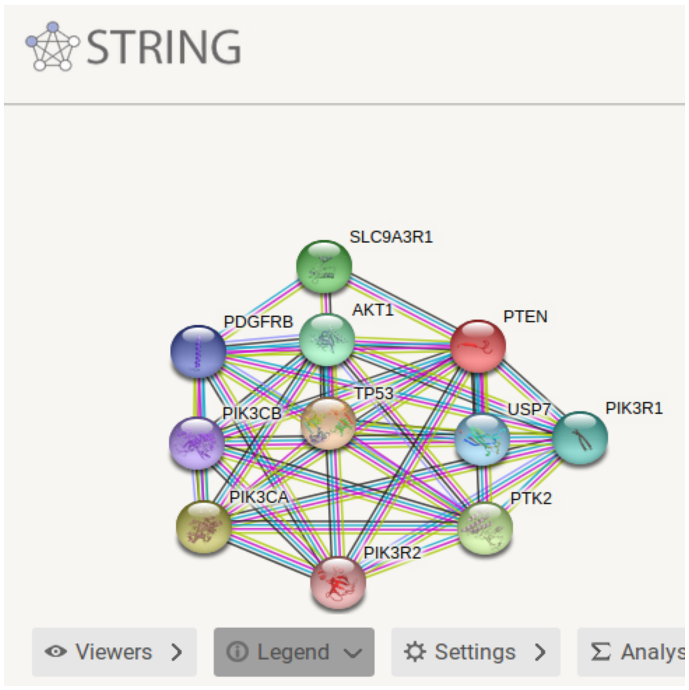
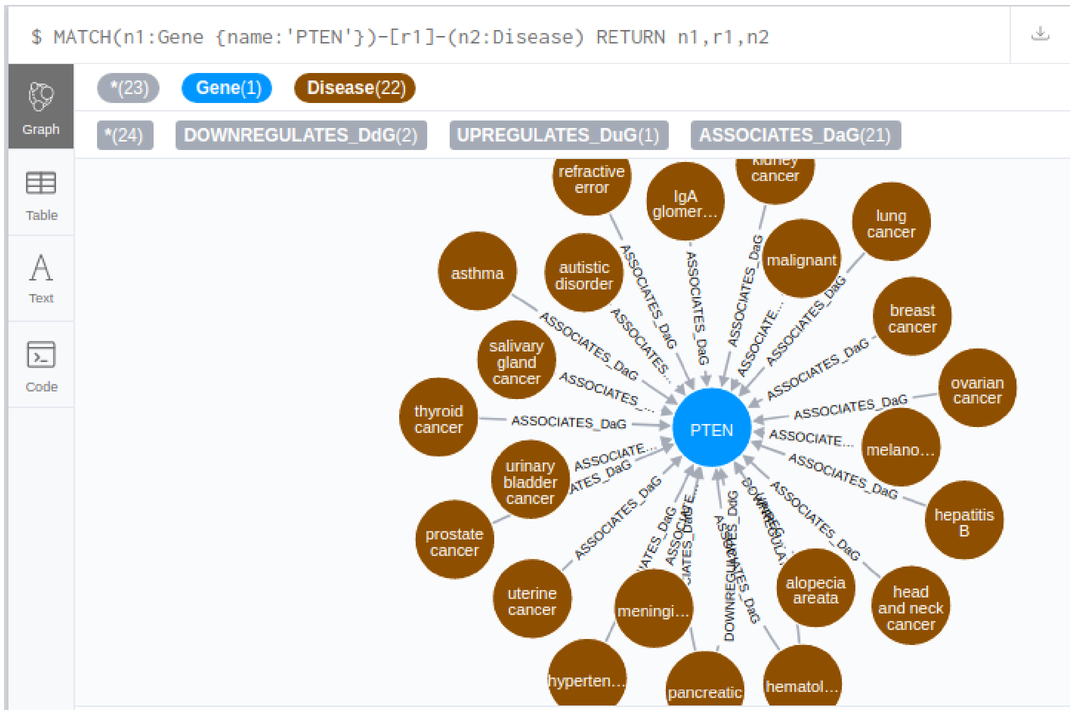


Figure 2.10: PTEN echo network in STRING. There are several edges between two proteins: one for each “evidence channel”

2.3.5 Hetionet

Hetionet [18] is a heterogeneous network of biomedical knowledge constructed from a collection of 29 publicly available databases and millions of publications. It was originally created as part of Project Rephetio, a study that utilizes the benefits of heterogeneous network to predict new uses for existing drugs. In the last few years, it has been modified for working over a broader variety of purposes: i) Drug Repurposing: used heterogeneous edge prediction to predict which disease is treated by each available compounds. ii) Prioritizing Disease-Associated Genes: used hetnet edge prediction for disease-associated genes, based on a predecessor network to Hetionet. The network contains many types of node that are: Gene, Disease, Compound, etc. In addition the relationships have been grouped into several categories such as “up/down regulates”, “inhibit”, etc. It is available over the website “https://het.io/”, and provides two search modes:

- Connectivity Search: is used to find the most important paths connecting any



```
$ MATCH(n1:Gene {name:'PTEN'})-[r1]-(n2:Disease) RETURN n1,r1,n2
```

Figure 2.11: PTEN-Drugs relationships in Hetionet shown through Neo4j.

two nodes in Hetionet, even when they are different types or not known to be related.

- Neo4j [65] Browser: is a graph database that supports the Cypher language for querying and visualizing heterogeneous network.

2.3.6 Reactome

Reactome [66] is a free, open-data, and peer-reviewed knowledge bases of biomolecular pathways [67] that contains a detailed representation of cellular processes interconnecting terms to form a graph of biological knowledge. Such network can be seen as a directed graph, which consists of nodes and directed edges connecting ordered pairs of nodes. Reactom stores pathway data in its natural form so that it does not require any transformation of data into a flat format. This approach reduces the complexity of the database and guarantees a easier access to the data. Compared with other biological pathways databases, Reactom adopts neo4j [65] as graph database to improve the data traversal performance and knowledge discovery. Reactome is available over the website:"https://reactome.org".

2.3.7 kindred

Kindred is a Python package builds upon the Stanford CoreNLP framework [68] and the scikit-learn machine learning library [69] for relation extraction using a vector-based approach. This approach transforms a span of text and candidate relation into a numerical vector that can be used in traditional machine learning classification. The algorithm used is Support vector machine (SVM) [44]. Relation candidates are created by finding every possible pair of entities within each sentence and stored with a class number for use in the multiclass classifier. The types of relations and how many classes are required for the multiclass classifier are based on the training data provided to Kindred. Kindred integrates PubAnnotation [13] and PubTator [14] to allow easy access to training data and text.

2.4 Annotations format

Once the entities and relationships have been infer from the documents, it is necessary to transform these in a format suitable for:

- classification and prediction methods.
- visualization tools.

Based on that, several representation formats have been proposed. The most important are: Biological expression language (BEL) [61] and Resource Description Framework (RDF) [12]

2.4.1 Biological Expression Language (BEL)

The Biological Expression Language (BEL) [61] is a language used to unambiguously represent casual biological entities and their inter-relationships reported in the scientific literature. The objective is to obtain knowledge with sufficient accuracy and semantics for building knowledge bases suitable for querying, powerful visualisation, and inference capabilities necessary to support today's life sciences research. Biological Expression Language can also represent a rich set of relationships, including correlative relationships, genomic relationships [70] and other relationships that can describe concepts such as biomarkers, or even define a hierarchical arrangement of concepts to form a taxonomy.

It is comprised of a set of function and relationships types usually used in conjunction with well known vocabularies such as HGNC human gene symbols [23], Gene Ontology [25], Disease Ontology [21], ChEBI [71], and MeSH. BEL also provides the necessary metadata to make the data findable, accessible, inter-operable and reusable, as well as guarantees a strong interoperability using standardized terminologies and external data sources for all biological entities.

The entities and relationships among them are represent through a triplet:

$$a(CHEBI : corticosteroid) - |bp(NCI : TissueDamage) \tag{2.2}$$

The topic treats in this example is related to the " amount of corticosteroid". Since, BEL's functional notation allows to represent sophisticated concepts as single terms, or combination of simpler ones, the abundance concept of this example has been abbreviated as "a". The biological process as "bp". Instead the relationships effect with "|-" to indicate a decrease of the target entity when the first one increase, or with "->" to indicate an increase of the target entity when also the first one increase. In addition, the entities can be prefixed with a "namespace" and a colon ":" to specify the source where the information has been extracted.

In this example, the "corticosteroid" information has been extracted from CHEBI,

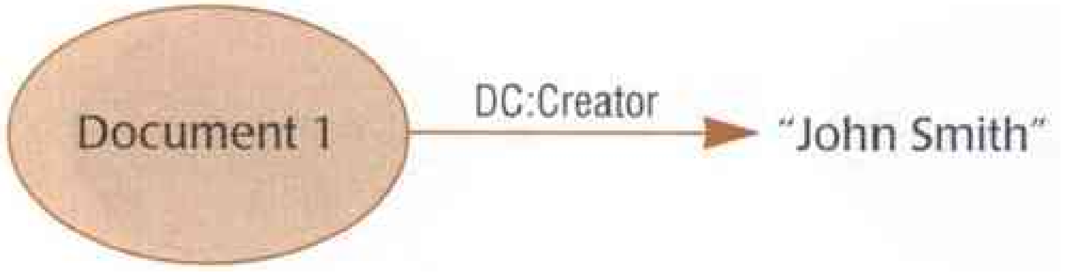


Figure 2.12: Writer-Document RDF representation []

while the “Tissue Damage” one from NCI (National Cancer Institute) vocabulary. In summary, the relationship in (2.2) equation claims that the presence of some amount of corticosteroid causes there to be less tissue damage.

2.4.2 Resource Description Framework

RDF [12] provides a model for describing resources characterized by attributes and properties. A resource is any object uniquely identifiable by an Uniform Resource Identifier (URI). The properties associated with resources are identified by a types, and property types have corresponding values. These property types express the relationships of values associated with resources. The values may be atomic (text strings, numbers, etc.) or other resources. At the core of RDF is a xml-based syntax model [72] for representing resources and their corresponding descriptions.

Due to several disambiguation caused by the use of same property type to mean very different concepts, RDF avoid this uniquely identifying property types through the XML name-space mechanism. XML name-spaces provide a method for unambiguously identifying the semantics and conventions of property types by uniquely identifying the governing authority of the vocabulary. For example, the property type "author" is defined as the "person or organization responsible for the creation of the intellectual content". An XML name space is used to unambiguously identify the Schema for the vocabulary by pointing to the definitive resource that defines the corresponding semantics.

In figure 2.12 is represents a resource “Document 1” with the semantics of property type “Creator” unambiguously defined in the context of DC vocabulary (Dublin Core). The value of this property type is John Smith.

The corresponding syntactic way of expressing this statement using XML namespaces is:

```
<?xml:namespace ns = "http://www.w3.org/RDF/RDF/" prefix ="RDF" ?>
<?xml:namespace ns = "http://purl.oclc.org/DC/" prefix = "DC" ?>
<RDF:RDF>
<RDF:Description RDF:HREF = "http://uri-of-Document-1">
<DC:Creator>John Smith<DC:Creator>
</RDF:Description>
</RDF:RDF>
```

The element <RDF:RDF> is a simple wrapper that marks the boundaries in an XML document where the content is explicitly intended to be mappable into an RDF data model instance. The element <RDF:Description> is correspondingly used to denote or instantiate a resource with the corresponding URI http://uri-of-Document-1. And the element <DC:Creator> in the context of the <RDF:Description> represents a property type DC:Creator and a value of “John Smith.”

Chapter 3

Algorithms and Frameworks

In this chapter, we introduce the frameworks and algorithms that have been used through this thesis to implement the BioTAGME core, annotate the text coming from scientific papers (full text or document title and abstract), and to build the knowledge graph which will be searchable through our implemented User Interface. Therefore, we will start providing an overview of the DT-Hybrid method [73] and Tagme [20] tool used to predict relationships among biological elements, and for annotating scientific text, respectively. We then give a description of SPARK [24] and Neo4j [65] frameworks employed for implementing the whole BioTAGME core, and for querying the obtained knowledge graph, respectively.

3.1 DT-Hybrid

The domain tuned-hybrid (DT-Hyrid) [73] extends the NBI algorithm proposed in Zhou et al. (2007) by adding application domain knowledge. This technique provides a complete and functional framework for in silico prediction of drug and target relationships.

DT-Hybrid employs the recommendation method based on the bipartite network projection technique that implements the concept of resources transfer within the network. Therefore, given a set of structures (i.e. biological compounds, molecules)

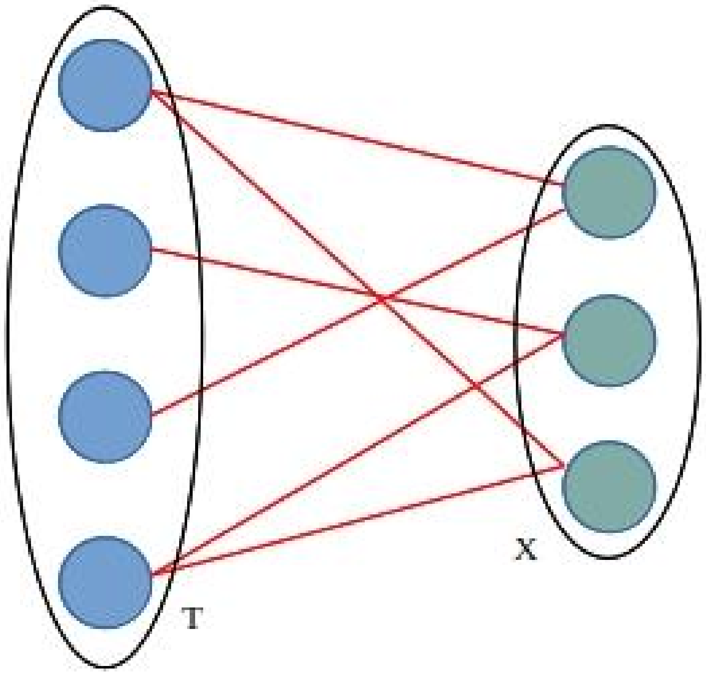


Figure 3.1: Bipartite Graph

$X = \{x_1, x_2, ..., x_m\}$, and a set of targets (i.e. genes, proteins) $T = \{t_1, t_2, ..., tn\}$; the X-T network of interactions can be described as a bipartite graph $G(X, T, E)$ where a link between $x_i \in X$ and and $t_j \in T$ is in the graph iff the structure x_i is associated with the target t_j. The network can be represented by an adjacency matrix A, where $a_{ij} = 1$ if x_i is connected to t_j; zero otherwise. Once the bipartite graph is built, a two-phase resource transfer is associated with one of its projections (Fig 3.1): at the beginning, the resource is transferred from nodes belonging to T to those in X, and subsequently the resource is transferred back to the T nodes.

This process allows to calculate each element of the weight matrix W in the projection as follows:

$$w_{i,j} = \frac{1}{\Gamma(i,j)} \sum_{l=1}^{m} \frac{a_{i,l} a_{j,l}}{k(x_l)} \tag{3.1}$$

Where:

- $k(x_l)$ is the degree of the x_l node in the bipartite network.
- $\Gamma(i,j) = \frac{1}{k(t_i)^{\alpha} k(t_j)^{\alpha-1}}$ where $k(t_i)$ and $k(t_j)$ are the degrees of the t_i and t_j, respectively.

We denote the sum of the equation(1) by $\mathrm{D}_{i,j}(S)$ symbol. Given the weight matrix W and the adjacency matrix A of the bipartite network, the recommendation matrix “R” is computed by the product between A and W. Finally, each column i of the R matrix is sorted in a score-based descending order so that the structures s_i having a higher interaction with the target t_j are in the first rows of the matrix column i. DT-Hybrid maintains only the top L columns values. Where L is a user-defined parameter.

3.2 Custom DT-Hybrid

In this thesis, we implemented a custom DT-Hybrid algorithm that considers: the documents as structures, the annotation terms (keywords) as target, and elements of the matrix R as the confidence score in the association of term t_i to term t_j. In addition, we have considered only the W matrix because we are just interested in terms relationships and not terms-documents ones. Since DT-Hybrid presents a structure and implementation incompatible with the Spark distributed concept, we re-implemented the whole algorithm using Scala and Spark SQL languages. In that regard, the matrices are replaced with Spark DataFrame, and the matrix product with join operations.

Given the “Documents_Terms” DataFrame containing all the annotation terms for each document s_i, the weight one "W" is computed as follows: at the beginning a self-join on the column “document_id” is applied to join the “Documents_Terms” to itself for generating the “Documents_Term$_i$_Term$_j$" DataFrame. This DataFrame contains all the associations Term$_i$-Term$_j$ located in the same document.

Then, two GROUP BY operations are executed on the “Documents_Terms”: the

Documents_Term$_i$_Term$_j$		
document_id	term$_i$	term$_j$
s_0	t_0	t_1
s_0	t_0	t_3
s_0	t_0	t_7
s_0	t_3	t_7
...	...	...
s_n	t_3	t_6

DataFrame 3.1

first one groups all the rows having the same "document_id" to generate the "Documents_recipDeg" DataFrame containing the reciprocal of the documents degree. Where the document$_i$ degree means the number of annotation terms linked by the document$_i$. Instead, the second one groups all the rows having the same "term_id" to generate the "Terms_recipDeg" DataFrame containing the reciprocal of the terms degree. Where the term$_j$ degree means the number of documents linked by the term$_j$.

Documents_recipDeg	
document_id	recip_deg
s_0	$\frac{1}{deg(s_0)}$
s_1	$\frac{1}{deg(s_1)}$
...	...
s_n	$\frac{1}{deg(s_n)}$

DataFrame 3.2

Terms_recipDeg	
term_id	recip_deg
t_0	$\frac{1}{deg(t_0)}$
t_1	$\frac{1}{deg(t_1)}$
...	...
t_m	$\frac{1}{deg(t_m)}$

DataFrame 3.3

After that, a document_id-based join is applied to join the "Docuents_Term$_i$_Term$_j$" DataFrame with the "Documents_inverseDeg" one. All the rows having the same term$_i$-term$_j$ value are then group to compute, through the Spark built-in "sum" function applied on the column "recip_deg", the contribute that term$_i$-term$_j$ associations received from all the documents (D$_{i,j}$(S)). This intermediate DataFrame is called "Documents_contribution".

Finally two other "JOIN" operations are executed: The first is a term$_i$-based join between the "Documents_contribution" DataFrame with the "Term_inverseDeg" one. Instead, the second one is similar to the previous, but the join is carried out on the term$_j$ column. Below is the Documents_contribution DataFrame after the "JOIN" operations.

Documents_contriution				
term$_i$	term$_j$	D$_{i,j}$(S)	recip_t_i_deg	recip_t_j_deg
t_0	t_5	D$_{0,5}$(S)	$\frac{1}{deg(t_0)}$	$\frac{1}{deg(t_5)}$
t_0	t_{12}	D$_{0,12}$(S)	$\frac{1}{deg(t_0)}$	$\frac{1}{deg(t_{12})}$
...	...	...	...	...
t_k	t_r	D$_{k,r}$(S)	$\frac{1}{deg(t_k)}$	$\frac{1}{deg(t_r)}$

DataFrame 3.4

Performing the product among the columns D$_{i,j}$(S), recip_t_i_deg, and recip_t_j_deg , we obtained all the $w_{i,j}$ terms of the "W" DataFrame. This DataFrame will be used during the Prediction procedure.

W		
term$_i$	term$_j$	w$_{i,j}$
t_0	t_5	w$_{0,5}$
t_0	t_{12}	w$_{0,12}$
...	...	...
t_k	t_r	w$_{k,r}$

DataFrame 3.5

3.3 Tagme framework

Tagme [20] annotates, on-the-fly and with high precision/recall, short text with pertinent hyperlinks to Wikipedia articles. Tagme uses Wikipedia anchor texts as spots and the pages linked to them in Wikipedia as their possible meanings. The authors employ Wikipedia as corpus because of ever-expanding pages number. In addition, it enriches texts with explanatory links to provide structured knowledge about any unstructured fragment of text. So any task that is currently addressed with bags of words-indexing could use these links to draw a vast network of concepts and their inter-relations. The anchor for a Wikipedia page p is the text used in another Wikipedia page to point to p. All the anchors composed by one character or just numbers have been discarded because these could be unsuitable for annotation and probably misleading for disambiguation.

To solve ambiguity and polysemy, Tagme tries to disambiguate each anchor “a” in “AT” (set of all anchors occurring in the input text “T”) by computing a score for each possible meaning “pa” of “a” through a new notation of “collective agreement” between “pa” and the possible meanings of all other anchors detected in “T”. Where “pa” in “Pg(a)” (set of all Wikipedia pages linked by “a”). Therefore, for each other anchor “b” in $AT - \{a\}$ is computed its vote to the annotation “a” -> “pa”. Since “b” could have several meanings, Tagme compute the vote as the average relatedness between each meaning “pb” of the anchor “b” and the meaning “pa” that should be associated to the anchor “a”. However, not all possible meanings of “b” have the same significance, so Tagme weight the contribution of “pb” by means of its commonness “Pr(pb|b)”. The formula used to compute the vote is:

$$\sum_{pb \in Pg(b)} rel(pb, pa) \bullet Pr(pb|b)|Pg(b)| \tag{3.2}$$

When “b” is unambiguous, the vote $b(pa) = rel(pb, pa)$ because the product between “Pr(pb|b)” and “|Pg(b)|” is equal to one. But if “b” is polysemous, only the meanings “pb” related to “pa” will affect the vote “b(pa)”. The total score rel$_a$(pa) for the “goodness” of the annotation “a->pa” is computed as the sum of the votes given to it by all other anchors “b” detected in the input text.

To disambiguate “a”, the best annotation “a->pa” is selected through two ranking algorithms combination: Disambiguation by Classifier (DC) and Disambiguation by Threshold (DT). DC uses a classifier with the score “rel$_a$(pa)”, and the commonness “$Pr(pa|a)$” as features to compute a “probability of correct disambiguation” for all meanings $pa \in Pg(a)$. Among all these meanings, DC selects the “pa” reporting the highest classification score. Instead, DT recognizes a roughness in the value of

"rel_a(pa)" among all $pa \in Pg(a)$, so it computes the top-best meanings p' in Pg(a) according to their "rel_a(p')", and then annotates "a" with the means that obtains the highest commonness among them.

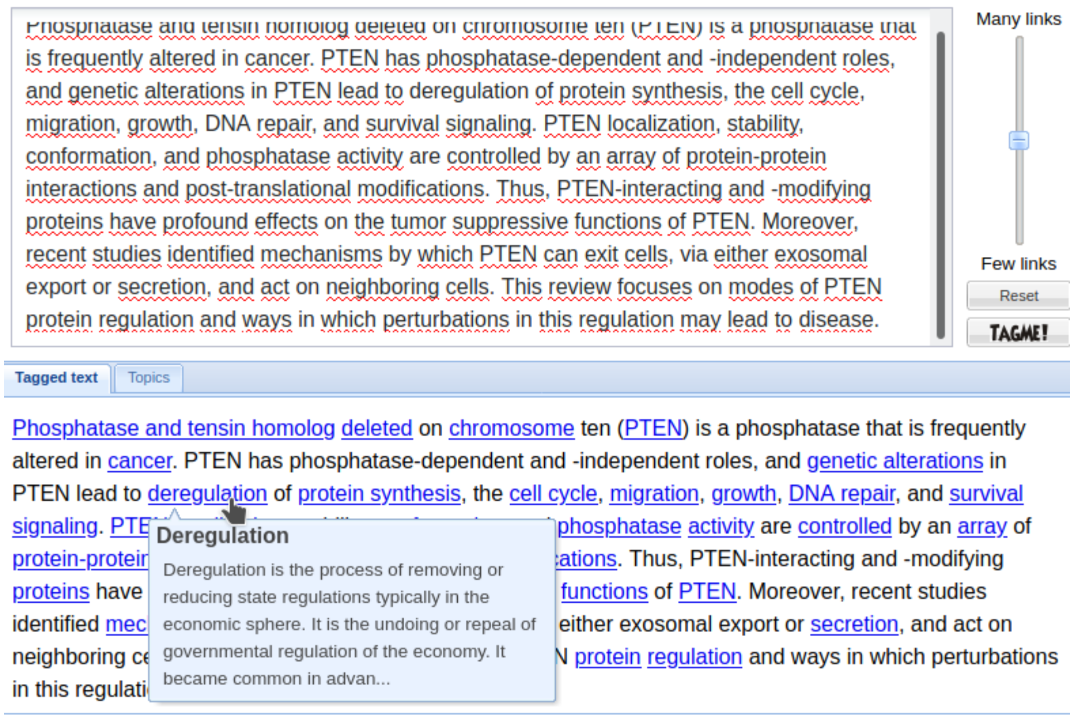
The set "M(AT)" of candidates produced by the Disambiguation Phase has to be pruned in order to remove improper anchors "a". The "bad anchors" are detected via a novel scoring function based on only two features: the link probability "lp(a)" of the anchor "a" and the coherence of its candidate annotation "a->pa" with respect to the candidate annotations of the other anchors in "M(AT)". Where "lp(a)" is defined as the ratio between the number of times the text "a" occurs as an anchor in Wikipedia ("link(a)") and the number of times the text a occurs in Wikipedia as an anchor or not ("freq(a)"). Instead, the coherence is the average relatedness between the candidate meaning "pa" for "a" and the candidate meaning "pb" for all other anchors "b" in "T". The objective is to keep all anchors whose link probability is high or whose assigned sense (page) is coherent with the senses (pages) assigned to the other anchors in M(AT). At the end of pruned procedure, we obtain the best anchors set to the Text T submit from the user.

3.4 Custom Tagme (OntoTagme)

One of the major problem encountered during the BioTAGME annotation procedure was a high anchor-meaning association error due to the excessive generality of the Wikipedia corpus used by Tagme.

To mitigate this Tagme negativity in a biological context, I, my research team, and the student Stefano Borzì have implemented a new custom Tagme version, called OntoTAGME, which uses biological and biomedical ontologies suitable to perform annotations on text coming from scientific papers. The basic idea was to change the Wikipedia corpus with a more specific one generated from the following databases: GeneOntology [], Drug-Bank [],DisGeNET [], and Obofoundry [].

In order to ensure high Tagme performance with the new corpus, we developed a



Phosphatase and tensin homolog deleted on chromosome ten (PTEN) is a phosphatase that is frequently altered in cancer. PTEN has phosphatase-dependent and -independent roles, and genetic alterations in PTEN lead to deregulation of protein synthesis, the cell cycle, migration, growth, DNA repair, and survival signaling. PTEN localization, stability, conformation, and phosphatase activity are controlled by an array of protein-protein interactions and post-translational modifications. Thus, PTEN-interacting and -modifying proteins have profound effects on the tumor suppressive functions of PTEN. Moreover, recent studies identified mechanisms by which PTEN can exit cells, via either exosomal export or secretion, and act on neighboring cells. This review focuses on modes of PTEN protein regulation and ways in which perturbations in this regulation may lead to disease.

Phosphatase and tensin homolog deleted on chromosome ten (PTEN) is a phosphatase that is frequently altered in cancer. PTEN has phosphatase-dependent and -independent roles, and genetic alterations in PTEN lead to deregulation of protein synthesis, the cell cycle, migration, growth, DNA repair, and survival signaling. PTE phosphatase activity are controlled by an array of protein-protein ations. Thus, PTEN-interacting and -modifying proteins have functions of PTEN. Moreover, recent studies identified mec either exosomal export or secretion, and act on neighboring ce N protein regulation and ways in which perturbations in this regulati

Deregulation

Deregulation is the process of removing or reducing state regulations typically in the economic sphere. It is the undoing or repeal of governmental regulation of the economy. It became common in advan...

Figure 3.2: Error encountered during the Tagme annotation

two-step procedure that converts a generic ontology or database in a Wikipedia-like database. First, each biological element within the downloaded database or ontology is transformed in a XML file containing the unique ID of the biological element in the downloaded source, its name (title), type (category), and other details such as abstract and body of the page. Then, for each page is generated and associated a tuple containing the new ID assigned by our customized version, and the basic information of the page: $< Tagme\_ID, title, category, ... >$. These tuples are stored within the "wiki-latest-page" SQL table.

Since an element “j” could have several linked pages “LPs” (i.e.DOID:0002116 is a DOID:10124) or redirect pages “RPs” (i.e. GO:0006200 replace by GO:0016887), we generated a tuple $< uniqueIDj, uniqueIDk >$ for each element “k” belonging to “LPs” and a tuple$< uniqueIDj, uniqueIDi >$ for each element “i” belonging to “RPs”. These tuples are stored in the tables “wiki-latest-pagelinks” and “wiki-latest-redirect”, respectively.

Next, the SQL tables and XML pages are converted in nodes and edges which can be manipulated by OntoTAGME. Note that, the core of OntoTAGME is the same of Tagme one. We only modified the corpus and the data structure used by Tagme to generate the network. OntoTAGME did not use in the actual version of BioTAGME. However, it has been tested in another project, called NetME, that will be explained in the following sections.

3.5 SPARK framework

Apache Spark [24] is an analytic engine for large-scale data processing. It provides high-level APIs in Java, Scala, Python and R, and an optimized engine that supports general execution graphs. It also supports a rich set of higher-level tools including Spark SQL for SQL and structured data processing, MLlib for machine learning, GraphX for graph processing, and Structured Streaming for incremental computation and stream processing.

SPARK application consists of a driver program that runs the user's main function in a parallel mode on a cluster. The SPARK component that establishes both the driver-machines of the cluster communication, and the resources allocation across applications is the "SparkContext". Such component can connect to several types of cluster managers (either Spark's own standalone cluster manager, Mesos [74] or YARN [75]). Once connected, SPARK acquires executors on nodes in the cluster, which are processes that run computations and store data for your application. Next, it sends our application code (defined by JAR or Python files passed to SparkContext) to the executors. Finally, SparkContext sends tasks to the executors to run.

The main aspects to consider about this architecture are []:

- Each application gets its own executor processes, which stay up for the duration of the whole application and run tasks in multiple threads. This has the benefit of isolating applications from each other, on both the scheduling side (each driver schedules its own tasks) and executor side (tasks from different

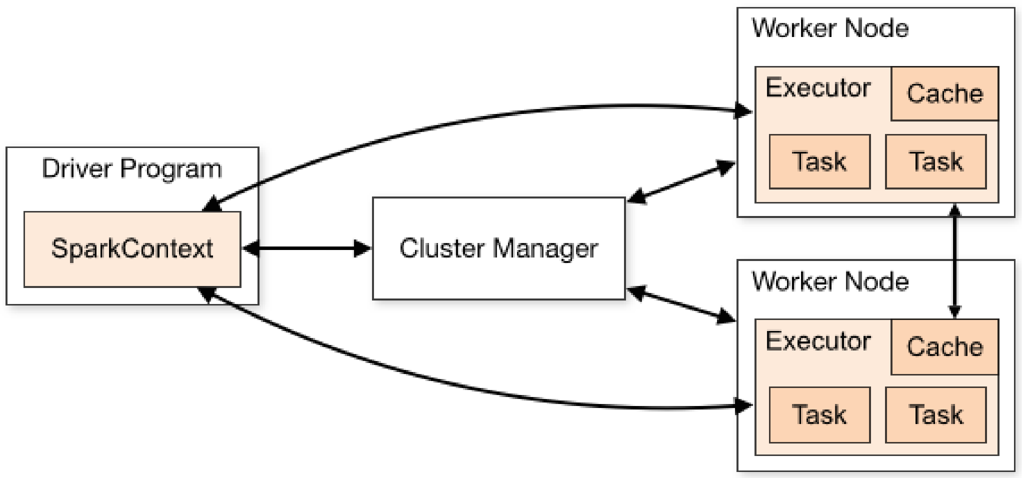


Figure 3.3: SPARK architecture

applications run in different JVMs). However, it also means that data cannot be shared across different Spark applications (instances of SparkContext) without writing it to an external storage system.

- SPARK is agnostic to the underlying cluster manager. As long as it can acquire executor processes, and these communicate with each other, it is relatively easy to run it even on a cluster manager that also supports other applications (e.g. Mesos/YARN).
- The driver program must listen for and accept incoming connections from its executors throughout its lifetime. As such, the driver program must be network addressable from the worker nodes.
- Because the driver schedules tasks on the cluster, it should be run close to the worker nodes, preferably on the same local area network. If you'd like to send requests to the cluster remotely, it's better to open an RPC to the driver and have it submit operations from nearby than to run a driver far away from the worker nodes.

Regarding the data structures, SPARK provides:

- **Resilient distributed dataset (RDD)** is a collection of elements partitioned across the nodes of the cluster that can be operated on in parallel. RDDs are created by a file in the Hadoop file system (or any other Hadoop-supported file system), or an existing Scala collection in the driver program, and transforming it. Users may also ask SPARK to persist an RDD in memory, allowing it to be reused efficiently across parallel operations. Finally, RDDs automatically recover from node failures.

- **shared variables** can be used in parallel operations. By default, when SPARK runs a function in parallel as a set of tasks on different nodes, it ships a copy of each variable used in the function to each task. Sometimes, a variable needs to be shared across tasks, or between tasks and the driver program. SPARK supports two types of shared variables: broadcast variables, which can be used to cache a value in memory on all nodes, and accumulators, which are variables that are only "added" to, such as counters and sums.

- **Dataset** is a distributed collection of data. Dataset is a new interface added in SPARK 1.6 that provides the benefits of RDDs (strong typing, ability to use powerful lambda functions) with the benefits of Spark SQL's optimized execution engine. A Dataset can be constructed from JVM objects and then manipulated using functional transformations (map, flatMap, filter, etc.). The Dataset API is available in Scala and Java. Python does not have the support for the Dataset API. But due to Python's dynamic nature, many of the benefits of the Dataset API are already available. The case for R is similar.

- **DataFrame** is a Dataset organized into named columns. It is conceptually equivalent to a table in a relational database or a data frame in R/Python, but with richer optimizations under the hood. DataFrames can be constructed from a wide array of sources such as: structured data files, tables in Hive, external databases, or existing RDDs. The DataFrame API is available in Scala, Java, Python, and R. In Scala and Java, a DataFrame is represented by a Dataset of Rows. In the Scala API, DataFrame is simply a type alias of Dataset[Row]. While, in Java API, users need to use Dataset<Row> to represent a DataFrame.

Thank to the SPARK potentialities (computing distribution on a cluster of machines), and the widely collection of data structures, we decided to employ this framework to implement all the BioTAGME core parts related to data cleaning, text annotation, and network construction. More information about these components will be explained in the next chapters.

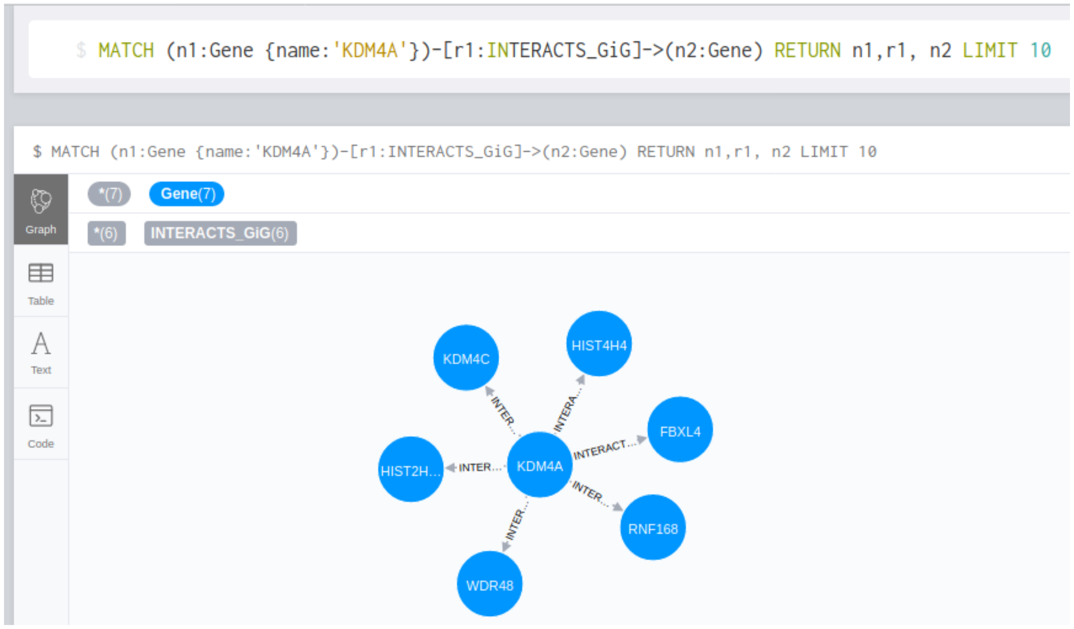
3.6 Neo4j

Neo4j [65] is an open-source, NoSQL, native graph database that provides an ACID-compliant transactional back-end for applications. A graph database [76], also referred as a semantic database, is a software application designed to store, query and modify network graphs. A network graph is a visual construct that consists of nodes and edges. Each node represents an entity (such as a person) and each edge represents a connection or relationship between two nodes.

Traditionally classified as a type of NoSQL database, the graph database records are referred as triples $< subject, predicate, object >$ in order to use special indexes that allow to improve the performance of the system as well as to reduce the queries running time.

Neo4j is referred to as a native graph database because it efficiently implements the property graph model down to the storage level. This means that the data is stored exactly as you whiteboard it, and the database uses pointers to navigate and traverse the graph. In contrast to graph processing or in-memory libraries, Neo4j also provides full database characteristics, including ACID transaction compliance, cluster support, and runtime failover - making it suitable to use graphs for data in production scenarios.

Some of the following particular features make Neo4j very popular:



```
$ MATCH (n1:Gene {name:'KDM4A'})-[r1:INTERACTS_GiG]->(n2:Gene) RETURN n1,r1, n2 LIMIT 10
```

```
$ MATCH (n1:Gene {name:'KDM4A'})-[r1:INTERACTS_GiG]->(n2:Gene) RETURN n1,r1, n2 LIMIT 10
```

Figure 3.4: In this example is reported the sub-graph extracted from Hetionet using Neo4j and the cypher language. The query (written in cyper format on the top of the picture) return the first ten interactions among the gene KDM4A and other genes within the hetionet network.

- **Cypher** [77], a declarative query language similar to SQL, used to look for motifs, clusters, and particular configuration within the graph database.
- **Constant time traversals** in big graphs for both depth and breadth due to efficient representation of nodes and relationships. Enables scale-up to billions of nodes on moderate hardware.
- **Flexible** property graph schema that can adapt over time, making it possible to materialize and add new relationships later to shortcut and speed up the domain data when the business needs change
- **Drivers** for popular programming languages, including Java, JavaScript, .NET, Python, and many more.

In this thesis, Neo4j has been employed to store the BioTAGME network. Such network may be processed by the users though our User Interface.

Chapter 4

BioTAGME

BioTAGME is a framework backed by two different pipelines for building a biological knowledge graph by applying several machines learning algorithms (DT-Hybrid and TAGME relatedness computing) over the whole PubMed titles and abstracts database.

The first pipeline is built on top of the SPARK analytic engine and Hadoop Distributed File System (HDFS) to guarantee large-scale data processing through scalable cluster managers, i.e Apache Meson, YARN, etc. This pipeline collects into DataFrames the data from major data sources i.e. DrugBank, HGNC, Ensembl, DisGeNET, Pathway Commons, and PubMed in order to process them though the Spark SQL language, and then returns two csv files (Nodes.csv, Edges.csv) containing nodes and relationships of the graph as soon as the appropriate elaborations have been completed. DataFrame and SQL language provide a common way to access a variety of data file, including Hive, Avro, Parquet, CSV, TSV and JSON [SparkReference]. The major functionalities provided by the first pipeline are: (i) Download and import, (ii) SQL to JSON parser, (iii) Integrating databases, (iv) Annotation, (v) Prediction, (vi) Network generation, and (vii) Updating. Such functionalities will be describe in the next sections.

Instead, the second pipeline (It forms the User Interface) is built on top of the Laravel framework and is backed by the following components: (j) MySQL for stor-

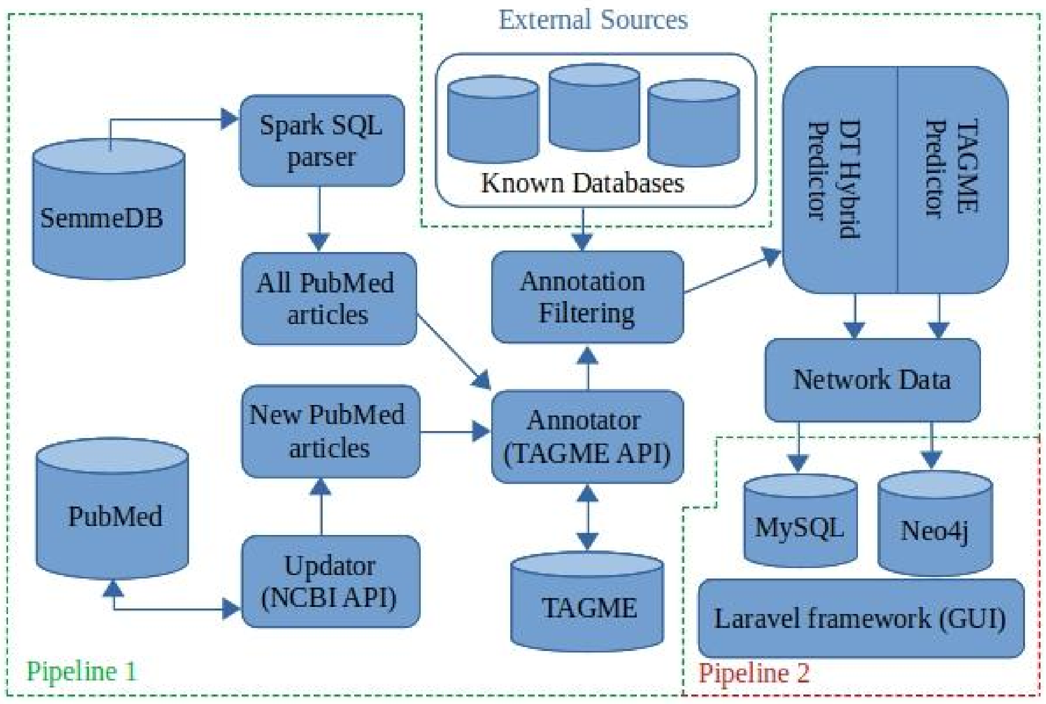


Figure 4.1: BioTAGME architecture

ing information related to biological elements such as names, aliases, BioTAGME ids and wikipedia pages associated during the annotation procedure. This will allow the user to make a based name or alias element search by the user interface. (jj) Neo4j for drawing the echo network of each biological element or for computing the shortest path between two user-specified biological ones. Data processing is done in PHP and bash to achieve high performances. In addition, all the GUI modules have been realized in react-native. The mainly functionalities of the second pipeline are: (i) Network import, (ii) Searching.

In this chapter, we first describe the cluster configuration has been used to run the pipeline one and two, and the external sources employed to extract the literature evidence. Then we explain all the main components of the pipeline one including external libraries or APIs that allow to build the Network. Finally, we describe all the User Interface components of the pipeline two needed for querying the knowledge graph.

4.1 Cluster configuration

Just like we widely discussed, our project aims to build a knowledge graph by annotating all the PubMed [36] documents' title and abstract, and applying the combination of DT-Hybrid algorithm [73] with relatedness computing method of Tagme [20] as relationships prediction technique. The huge amount of data to elaborate (Big data context), however, really requires:

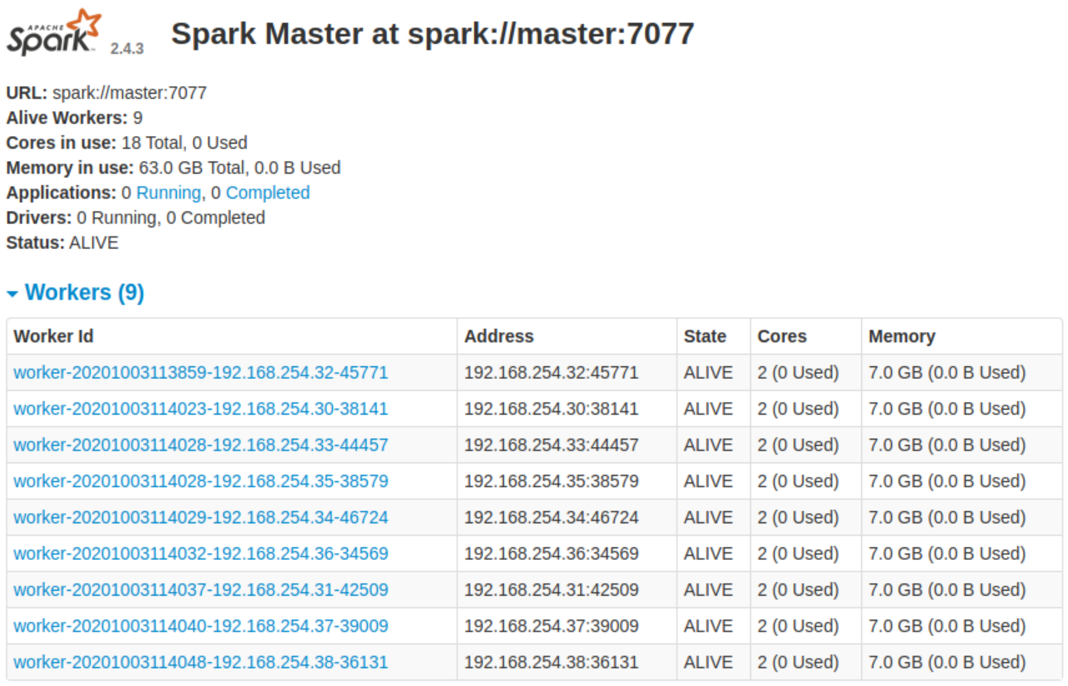
- a cost-effective computing power to ensure greater performance, and faster data elaboration.
- a huge amounts of RAM memory to avoid the data SWAP from RAM to SWAP partition due to the excessive size of the data.

To overcome these issues, we deployed the pipeline one on a SPARK cluster [24] made available by the Catania section of the National Institute for Nuclear Physics (INFN). This cluster is composed of:

- one master having 8 GiB of RAM, 2 Virtual CPUs of 2GHz, one disk of 500 GB.
- nine slaves having 8 GiB of RAM, 2 Virtual CPUs of 2GHz, one disj of 500 GB.

Note that, the cluster machines that we used are undersized compared with the SPARK specifications due to the limited resources available of the INFN. Therefore, if you want to execute the pipeline one on a SPARK cluster in an efficient way, we suggest you to use a cluster composed of several machines having the following specifications:

- 8-16 cores per machine at least
- a 10 Gigabit or higher network is the best way to make the applications faster


2.4.3 **Spark Master at spark://master:7077**

URL: spark://master:7077
Alive Workers: 9
Cores in use: 18 Total, 0 Used
Memory in use: 63.0 GB Total, 0.0 B Used
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (9)

Worker Id	Address	State	Cores	Memory
worker-20201003113859-192.168.254.32-45771	192.168.254.32:45771	ALIVE	2 (0 Used)	7.0 GB (0.0 B Used)
worker-20201003114023-192.168.254.30-38141	192.168.254.30:38141	ALIVE	2 (0 Used)	7.0 GB (0.0 B Used)
worker-20201003114028-192.168.254.33-44457	192.168.254.33:44457	ALIVE	2 (0 Used)	7.0 GB (0.0 B Used)
worker-20201003114028-192.168.254.35-38579	192.168.254.35:38579	ALIVE	2 (0 Used)	7.0 GB (0.0 B Used)
worker-20201003114029-192.168.254.34-46724	192.168.254.34:46724	ALIVE	2 (0 Used)	7.0 GB (0.0 B Used)
worker-20201003114032-192.168.254.36-34569	192.168.254.36:34569	ALIVE	2 (0 Used)	7.0 GB (0.0 B Used)
worker-20201003114037-192.168.254.31-42509	192.168.254.31:42509	ALIVE	2 (0 Used)	7.0 GB (0.0 B Used)
worker-20201003114040-192.168.254.37-39009	192.168.254.37:39009	ALIVE	2 (0 Used)	7.0 GB (0.0 B Used)
worker-20201003114048-192.168.254.38-36131	192.168.254.38:36131	ALIVE	2 (0 Used)	7.0 GB (0.0 B Used)

Figure 4.2: Cluster configuration where the pipeline one is deployed

- 4-8 disks per node configured without RAID (just as separate mount points)
- from 8 GiB to hundreds of gigabytes of memory per machine. In all cases, we recommend allocating only at most 75% of the memory for Spark; leave the rest for the operating system and buffer cache.

4.2 External Sources

Below, we provide a brief overview of the main biological elements databases that have been integrate in our pipeline one for both extracting the Literature evidence and mitigating the disambiguation error.

4.2.1 DisGeNET

DisGeNET [26] is a discovery platform containing one of the largest publicly available collections of genes and variants associated to human diseases. It integrates data from expert curated repositories, GWAS catalogues, animal models and the scientific literature. DisGeNET data are homogeneously annotated with controlled vocabularies and community-driven ontologies. Additionally, several original metrics are provided to assist the prioritization of genotype–phenotype relationships.

The current version of DisGeNET (v7.0) contains 1,134,942 gene-disease associations (GDAs), between 21,671 genes and 30,170 diseases, disorders, traits, and clinical or abnormal human phenotypes, and 369,554 variant-disease associations (VDAs), between 194,515 variants and 14,155 diseases, traits, and phenotypes.

DisGeNET is a versatile platform that can be used for different research purposes including the investigation of the molecular underpinnings of human diseases, the analysis of the properties of disease genes, the generation of hypothesis on drug therapeutic action and drug adverse effects, the validation of computationally predicted disease genes and the evaluation of text-mining methods performance.

4.2.2 Disease Ontology

Disease Ontology (DO) [21] represents a comprehensive knowledge base of 8043 inherited, developmental and acquired human diseases. In addition, it integrates disease and medical vocabularies through extensive cross mapping and integration of MeSH, ICD, NCI's thesaurus, SNOMED CT and OMIM disease-specific terms and identifiers. The DO is utilized for disease annotation by major biomedical databases (e.g. Array Express, NIF, IEDB), as a standard representation of human disease in biomedical ontologies.

4.2.3 DiseaseEnhancer

DiseaseEnhancer [78] provides a comprehensive map of manually curated disease-associated enhancers, which includes 1059 disease-associated enhancers in 167 human diseases, involving 896 unique enhancer-gene interactions. It also manually collected their dysregulated target genes and mechanistic-related information, such as associated variant types and affected transcription factor bindings. Additional genome data were also integrated into DiseaseEnhancer to help characterize disease-associated enhancers.

4.2.4 DrugBank

DrugBank combines detailed drug data with comprehensive drug target and drug action information. Such database facilitates in silico drug target discovery, drug design, drug docking or screening, drug metabolism prediction, drug interaction prediction and general pharmaceutical education. In addition, drug-drug and food-drug interactions, and experimental ADME have been added to each element of the database. DrugBank offers a suite of products powered by Platform and has customers located around the world crossing multiple industries including precision medicine, electronic health records, drug development and regulatory agencies. DrugBank also provides DrugBank Online as a free-to-access resource for academic research and is used by millions of pharmacists, pharmacologists, health professionals and pharmaceutical researchers every year. It is available at “http://www.drugbank.ca”. [22]

4.2.5 PharmGKB

PharmGKB [79] is a tool designed to displays pharmacogenomic relationships, genotype, molecular, and clinical knowledge integrated into pathway representations. Detailed relationship data from the literature is manually curated and described using controlled vocabularies. The files containing drugs/chemicals information in tabular format are available at the website “www.pharmgkb.org”. Such website provides a diverse array of PGx information, from annotations of the primary literature to guidelines for adjusting drug treatment based on genetic information. It is freely available and accessible to everyone from researchers to clinicians to everyday citizens.

4.2.6 HUGO Gene Nomenclature Committee

HUGO Gene Nomenclature Committee (HGNC) [23] assigns unique symbols and names for human genes in order to remove ambiguity with other databases. Part of these symbols have been defined for protein-coding genes, instead the rest to small and non-coding RNAs, and pseudo-genes.

HGNC have three main types of placeholder symbols: C#orf#s for predicted genes designated by the chromosome of origin, the letters 'orf' for open reading frame and an iterative number. KIAA#s are approved for genes identified by the Kazusa cDNA sequencing project when no other information is known about the gene. Finally, the FAM# root is used to group together a set of genes that are related based on sequence similarity, but cannot be described by function or conserved domains.

The file contains all the HGNC database in tsv format is available at the following website:"www.genenames.org".

4.2.7 Ensembl

Ensembl [80] is a tool that supports research in comparative genomics, evolution, sequence variation and transcriptional regulation. It annotates genes, computes multiple alignments, predicts regulatory function and collects disease data. Ensembl tools include BLAST, BLAT, BioMart and the Variant Effect Predictor (VEP) for all supported species. Ensembl is available as an interactive Web site, a set of flat files, and as a complete, portable open source software system for handling genomes. All data are provided without restriction, and code is freely available. Ensembl's aims are to continue to "widen" the biological integration to include other model organisms relevant to understanding human biology as they become available; to "deepen" the integration to provide an ever more seamless linkage between equivalent components in different species; and to provide further classification of functional elements in the genome that have been previously elusive.

4.2.8 LNCipedia

LNCipedia [81] is a highly consistent database composed of 21.488 annotated human long non-coding RNA (lncRNA) transcripts obtained from different source. For each of these transcripts several data i.e secondary structure, protein coding potential and microRNA binding, and available literature evidences have been associated in order to provide as much information as possible to the researcher.

Transcripts from different sources are added to the database avoiding redundant transcript collection. After each import, the chromosomal positions of the newly added exons are converted to either hg19 or hg38 using the liftOver tool. Only conversions with perfect remapping of all bases are considered. In the filtering step, the following transcript are removed from the database: i. transcripts that do not map to the hg38 reference genome, ii. transcripts not localized on the reference chromosomes, iii. transcripts with less than 200 nucleotides, iv. transcripts with exons that overlap coding sequence as annotated in the Ensembl 92 database. As soon as the filtering procedure is completed, all transcripts are clustered into genes based on exon overlap. These genes are then given a LNCipedia name.

4.2.9 miRcode

miRcode [82] has been developed to identifies putative microRNA-Target sites in lncRNAs of interest, or predicted specific microRNAs target across the GENCODE long non-coding transcriptome. GENCODE is used because contains many well known RNA genes and microRNAs as well as represents a comprehensive description of polyA+ transcriptome. Although all of GENCODE is accessible by miRcode, only lncRNA genes that produce predicted non-coding transcripts with a mature length higher than 200 nt are considered during the process. In addition, 6-mers are not considered because they are marginally effective. To assess evolutionary conservation, a 46-way Multiz vertebrate genomic multiple alignment is remapped onto transcripts, and site conservation levels are determined based on site presence in primates, non-primate mammals and non-mammal vertebrates. miRcode additionally covers other GENCODE gene classes, including 12,549 pseudogenes and 19,999 coding genes both in typical (3'-UTR) and atypical (5'-UTR and CDS) positions.

4.2.10 miRBase

miRBase [83] is a database based on comprehensive microRNA sequence data, annotation and predicted gene targets. miRBase performs three main roles that are: i) the miRBase Registry acts as an independent arbiter for the miRNA names assignment ii) the miRBase Sequences is the primary online repository for miRNA sequence data and annotation. iii) miRBase Targets is a comprehensive database of predicted miRNA target genes.

4.2.11 miRTarBase

miRTarBase [84] is an update database containing more than 3500 miRNA-Target (MTI) interactions obtained after a data mining procedure over several PubMed full-texts that potentially describe such interactions. All the MITs have been experimentally confirmed by reporter assays, western blot, microarray experiments, pSILAC or qRT–PCR. In addition, to examine the functions of the target genes involved in human MITs, a gene ontology (GO) and KEGG pathways enrichment annotation was carried out using the DAVID gene annotation scheme.

4.2.12 miRCancer

miRCancer [85] is a database composed of miRNA-cancer association extracted by a rules-based text mining system over the PubMed titles and abstracts. Regarding the cancer name recognition, the system compares the text with a dictionary compiled from the international classification of diseases for ontology (ICD-0). Instead, the miRNA name recognition is based on standardized names nomenclatures.

4.2.13 Reactome

Reactome [66] is a tool for visualisation, interpretation and analysis of pathway knowledge using neo4j as graph database. The Reactome data has been designed to generalize the reaction concept. This allows to include transformations of entities such as chemical transformations of classical biochemistry, etc. Entities include nucleic acids, small molecules, proteins (with or without post-translational modifications) and macromolecular complexes.

4.2.14 PathBank

PathBank [86] is a database that aims to provide a pathway for every protein. Each pathway includes information of the organs, organelles, subcellular compartments, cofactors, molecular locations, chemical structures and protein quaternary structures. In addition, references and detailed descriptions are included to provide an overview of the pathway. PathBank is basically designed to support pathway discovery in transcriptomics, proteomics, metabolomics and systems biology.

4.2.15 UniProt

UniProt [87] is a collection of database contains sequences and functional information available for proteins. It consist of four mainly components: (i) The UniProt Knowledge-base (UniProtKB) is an expertly curated database over integrated protein information with cross-references to multiple sources. It combines UniProtKB/Swiss-Prot and UniProtKB/TrEMBL. UniProtKB/Swiss-Prot contains over 550.000 sequences and experimental information extracted from the literature. UniProtKB/TrEMBL provides a further 60 million sequences derived from high throughput sequencing of DNA. Such sequence are annotated by rule-based annotation systems. (ii) The UniProt Archive (UniParc) is a sequence repository reflecting the history of all protein sequences. (iii) UniProt Reference Clusters (UniRef) merge closely related sequences based on sequence identity in order to reduce the searching time. (iv) The

UniProt Metagenomic and Environmental Sequences (UniMES) database is a repository developed for the metagenomic and environmental data. UniProt resources can be accessed via the website at http://www.uniprot.org/.

4.2.16 BRENDA

BRENDA [88] is a relational database contains functional and molecular information of enzymes, including data on the occurrence, catalyzed reaction, kinetics, substrates/products, inhibitors, co-factors, activators, structure and stability. All the enzymes have been classified according to the EC system of the IUBMB Enzyme Nomenclature Committee, and the entries are cross-referenced to other databases, i.e. organism classification, protein sequence, protein structure and literature references. In addition, BRENDA provides an academic web access at http://www.brenda.uni-koeln.de for research of enzyme mechanisms, metabolic pathways, the evolution of metabolism and, furthermore, for medicinal diagnostics and pharmaceutical research.

4.2.17 RefSeq

The Reference Sequence (RefSeq) [89] project at the National Center for Biotechnology Information (NCBI) provides annotation for over 95.000 prokaryotic genomes that meet standards for sequence quality, completeness, and freedom from contamination. Genomes are annotated by a single Prokaryotic Genome Annotation Pipeline (PGAP) to provide users with a resource that is as consistent and accurate as possible. Antimicrobial resistance proteins have been reannotated comprehensively, improved structural annotation of insertion sequence transposases and selenoproteins is provided, curated complex domain architectures have given upgraded names to millions of multidomain proteins, and we introduce a new kind of annotation rule, called BlastRules. Continual curation of supporting evidence, and propagation of improved names onto RefSeq proteins ensures that the functional annotation of

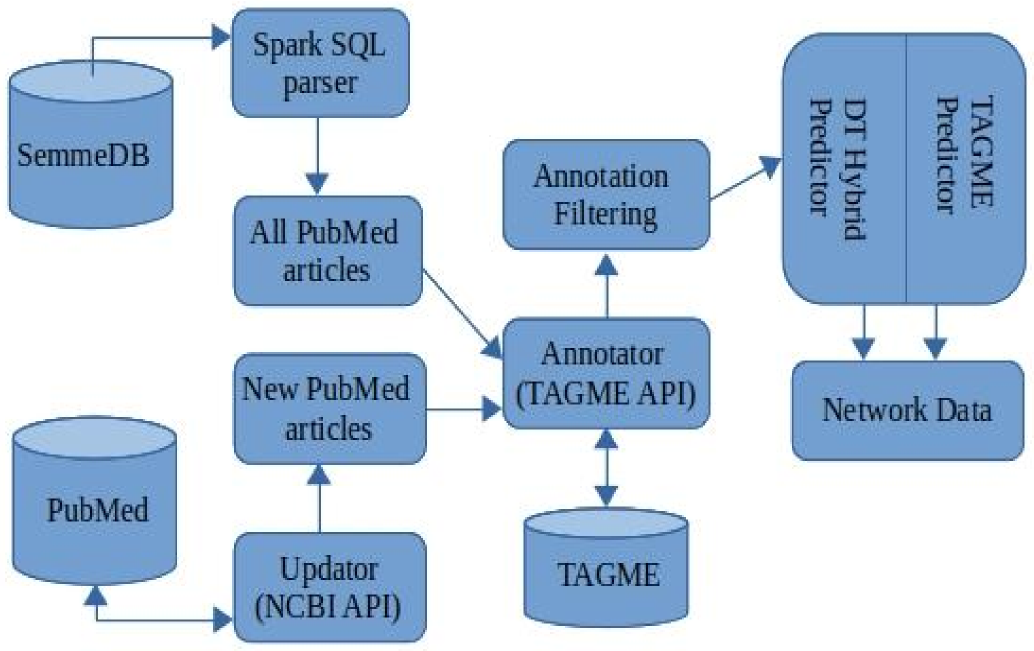


Figure 4.3: Pipeline one structure

genomes is kept current. RefSeq is found at “https://www.ncbi.nlm.nih.gov/refseq/”.

4.3 Pipeline one

In this section, we describe all the components and functionalities of the pipeline one (Fig. 4.3) necessary to transform all the unstructured documents’ title and abstract of PubMed in a Knowedge Graph.

4.3.1 Download and Import module

This module enables the user to import, into Hadoop FileSystem (HDFS), all the external databases through a bash file, called “importing.bash”, provided with Bio-TAGME package. Such file consists of three main sections:

- Pubmed section downloads all the articles title and abstract for the annotation procedure. Since the whole Pubmed database (about 30 millions of articles) requires much time to be downloaded through the NCBI Entrez APIs (only three queries per second are allowed) [37], we opted for downloading the SemmedDB database.

SemMedDB is a repository of semantic predictions extracted from the PubMed citations by SemRep, and consists of several MySQL tables. The SENTENCE will be the selected one because it contains all the sentences of PubMed titles and abstracts.

- Literature databases section downloads all the literature databases (listed in the "External databases" section) necessary:
  - for filtering all noisy annotation terms caused by disambiguation and high generality of the Wikipedia corpus employed by Tagme during the Annotation phase.
  - for building the literature edges to our knowledge network. These edges allows to evaluate the quality of BioTAGME prediction.

  Note that, some databases, such as DrugBank, PharmGKB, Brenda, require a free registration or authorization to be downloaded. Therefore, this procedure is left to the user.

- Import section transfers the downloaded databases from the local file system to the Hadoop one (HDFS).

4.3.2 SQL to JSON parser module

Although SemmedDB guarantees lesser downloading time than NCBI Entrez APIs, it presents two main problems:

- Title and abstract of each PMID are divided in sentences. There are, therefore, several contiguous SENTENCE table rows per title or abstract instead of just one.

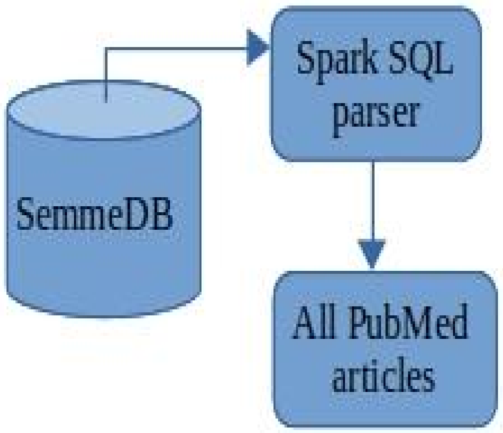


Figure 4.4: SQL to JSON parser

- The SENTENCE table is in SQL format, but Spark engine does not have native libraries for handling that.

To solve these issues, we implemented a new Spark module, named SQL2Json parser, that fulfils the following tasks and converts the SQL file in a JSON one (Fig. 4.4). First, the SENTENCE.sql file is loaded in a single-column Spark DataFrame, named “SENTENCE_df”, using the Spark built-in “read_txt” function. Since the file contains both table header and data, we start to extract the header by applying our implemented “get_table_header” function. This function employs the SQL Window concept and the SQL “CREATE TABLE” keyword to separate the header rows from those of the data. The header rows are then transform in a vector element by the SPARK collect function.

Once the header has been extracted, the comma-separated content of the remaining records are split into vectors, and then elaborated so that the i_{th} elements of the just generated vectors are associated with the appropriate index of the header one.

In order to make the “SENTENCE_df” DataFrame more appropriate for JOIN and GROUP BY operations, we need to transform the array column into separate columns. That is obtained by applying to the “SENTENCE_df” DataFrame the

Spark built-in "*select*" function with the header vector as argument. The features of the new DataFrame are: PMID, TYPE, SENTENCE_ID, SENTENCE. Where TYPE is equal to "ti" for title sentence, or "ab" to abstract one.

SENTENCE_df	
ID	ROW
0	$[pmid_0, type_0, sentence\_id_0, sentence_0]$
1	$[pmid_1, type_1, sentence\_id_1, sentence_1]$
...	...
n	$[pmid_n, type_n, sentence\_id_n, sentence_n]$

DataFrame 4.1

PMIDs_Data				
ID	PMID	TYPE	SENTENCE_ID	SENTENCE
0	$pmid_0$	$type_0$	$sentence\_id_0$	$sentence_0$
1	$pmid_1$	$type_1$	$sentence\_id_1$	$sentence_1$
...	...	...	...	...
n	$pmid_n$	$type_n$	$sentence\_id_n$	$sentence_n$

DataFrame 4.2

The final step of this module is to aggregate on the same row the Document$_i$ title and abstract making the following two consecutive "GROUP BY":

- The first one, combined with the spark built-in collect_list and concat_ws functions, groups the SENTENCE column content of the rows having the same "pmid" and "type" value in a new textual column that will contain the whole Document$_i$ title or abstract.

- The second one, combined with the spark built-in joinMap function, groups rows having same pmid so that Document$_i$ title and abstract will be in the same row.

The resulting DataFrame, named Titles_Abstracts, is stored into (HDFS) in JSON format and used during the Annotation procedure.

Titles_Abstracts		
PMID	TITLE	ABSTRACT
$pmid_0$	$title_0$	$abstract_0$
$pmid_1$	$title_1$	$abstract_1$
...	...	...
$pmid_n$	$title_n$	$abstract_n$

DataFrame 4.3

4.3.3 Integrating databases module

As previously mentioned, several literature databases are integrated by our pipeline in order to inject the literature evidence in our knowledge graph. However, there are various aspects and problems related to databases integration to consider:

- Different databases often use different words for the same thing (synonyms). Whereas DisGenNET for example uses "Colorectal cancer, hereditary non-polyposis, type 1", DiseaseOntology (DO) uses "Lynch syndrome 1" to refer the same disease.
- Equivalent attributes have different names in different databases. Whereas one database might for example use the attribute name "mirna_nr", another data-base might use "id" for the same attribute.
- Different format of the downloaded databases i.e JSON, XML, TXT, CSV, TAB, OBO, GTF, FASTQ, SQL, etc.

Starting from these remarks, we implemented our integrating module that executes the following tasks. First, all the databases are loaded into Spark DataFrame through:

- The built-in Spark read.csv, read.txt, or read.jsom functions for CSV, TAB, TXT and Json format, respectively.
- Our Spark define functions for OBO, GFT, SQL, and FASTQ format.
- The Databricks spark-xml for xml format.

Once all DataFrame have been generated, they are elaborated in order to extract two additional DataFrame per biological element type (it can be genes, proteins, or drugs, etc):

- The Indexing DataFrame that contains the assigned BioTAGME id, the official name, and the synonyms for each biological element. This DataFrame will be used to remove all noisy annotation terms. An example is reported in DataFrame 4.6, which is obtained through all Gene Databases employed in BioTAGME.

- The Relationships Dataframe contains many logical or physical relationships among biological elements extracted by the "external_reference" column of each considered database. For every relationship, we also reported the first and second "biological element" type (under the column "Type_i_Type_j") in order to speed up the searching and filtering operation. Such DataFrame will be used to create the literature edges for our knowledge network.

This module can be considered as an intermediate layer that transforms all external databases in new ones having same schema and attributes.

Gene_indexing		
BioTAGME_ID	NAME	ALIASES
bio_id_1	$gene\_name_1$	$[alias_{1,1},...,alias_{1,m}]$
bio_id_2	$gene\_name_2$	$[alias_{2,1},...,alias_{2,m}]$
...	...	...
bio_id_n	$gene\_name_n$	$[alias_{n,1},...,alias_{n,m}]$

DataFrame 4.4

4.3.4 Annotation module

This module (Fig. 4.5) runs all the procedures to transform all the PubMed documents title and abstract in a list of annotation terms. At beginning, the Titles_Abstracts DataFrame (DataFrame.4.3) is loaded through the Spark built-in read.json function.

Gene_relationships		
BioTAGME_ID$_i$	BioTAGME_ID$_j$	Type$_i$_Type$_j$
$bio\_id_{1,1}$	$bio\_id_{1,2}$	$gene - gene$
$bio\_id_{2,1}$	$bio\_id_{2,2}$	$gene - lncRNA$
...	...	...
$bio\_id_{n,1}$	$bio\_id_{n,2}$	$gene - disease$

DataFrame 4.5

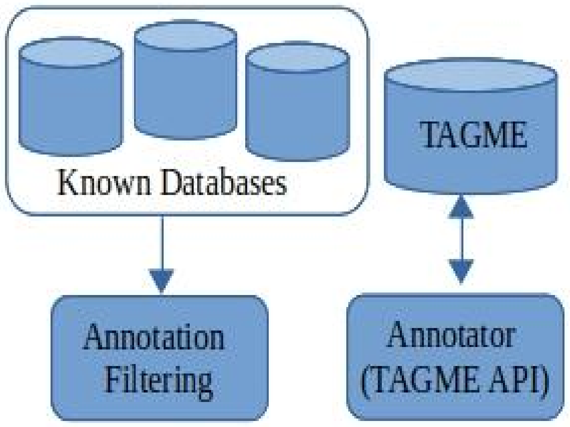


Figure 4.5: Annotation module components

Then, for each document "t$_i$", a tuple (TI_AB, TAGME parameters map)$_i$ is generated and sent to the "TAGME" tool through a HTTP POST request. Where TI_AB is the union of Document$_i$ Title and Abstract.

TAGME tool removes all stop-words and punctuation symbols from the TI_AB text at first. Next, a list of "anchors" to Wikipedia pages is extracted and returned in response to the request. This response is in a JSON format and contains the following information for each anchor: spot, wikipedia page title, wikipedia page categories, wikipedia page id, etc. These received JSONs constitute the Title_Abstract ANNOTATIONS column (DataFrame 4.6).

To complete the Annotation procedure, our pipeline builds the Annotation DataFrame applying the Spark "select" and "explode" function over the Titles_Abstracts one. The "explode" function transform all the vectors under the ANNOTATIONS column in several rows (one per anchor). Instead, the "select" one transforms the JSON

Titles_Abstracts			
PMID	TITLE	ABSTRACT	ANNOTATIONS
$pmid_0$	$title_0$	$abstr_0$	$[anch_{1,1},...,anch_{1,m}]$
$pmid_1$	$title_1$	$abstr_1$	$[anch_{2,1},...,anch_{2,m}]$
...	...	...	...
$pmid_n$	$title_n$	$abstr_n$	$[anch_{n,1},...,anch_{n,m}]$

DataFrame 4.6

fields of each anchor in columns (DataFrame 4.7).

Annotation				
PMID	SPOT	Wiki_Id	Wiki_title	Wiki_Categs
$pmid_i$	$spot_0$	pID_0	$pTitle_0$	$[c_{0,0},...,c_{0,t}]$
$pmid_i$	$spot_1$	pID_1	$pTitle_1$	$[c_{1,0},...,c_{1,k}]$
...	...	...	...	...
$pmid_i$	$spot_m$	pID_m	$pTitle_m$	$[c_{m,0},...,c_{m,s}]$
...	...	...	...	...

DataFrame 4.7

A main problem encountered during the annotation procedure is a high disambiguation error due to the generality of the Wikipedia corpus used by TAGME. To solve that, our pipeline filters the "noisy annotation terms" making a join among the Annotation DataFrame and all the biological_elements(Genes, Diseases, etc)_indexing ones generated by the "Integrate databases module". The final result is a DataFrame containing only biological elements-documents associations that will be used to predict the relationships amoung the elements.

4.3.5 Prediction module

The aim of our methodology is to predict one or more potential relationships among biological elements on the basis of the BioTAGME score value. This score is defined as the product between the DT-Hybrid (Alaimo ed al, 2013) [73] score and the TAGME relatedness [20]. Much higher is the score value much higher is the meaningful of the predicted relationship. To compute the BioTAGME score, we build the "Documents_Term$_i$_Term$j$ (DataFrame 4.8) at first. Such DataFrame contains all the associations Term$_i$-Term$_j$ that occur in the same document. Then, a group by operation on the columns Term$_i$ and Term$_j$ is performed to compute the frequency "f$_{i,j}$" (number of pmids that contains the Term$_i$ and Term$_j$ association). Such frequency will be used to weigh the DT-Hybrid score and TAGME relatedness. Below is the "Documents_Term$_i$_Term$j$ after the "GROUP BY" operation.

Documents_Term$_i$_Term$_j$		
Term$_i$	Term$_j$	f$_{i,j}$
t_0	t_5	$f_{0,5}$
t_0	t_{12}	$f_{0,12}$
...	...	...
t_3	t_6	$f_{3,6}$

DataFrame 4.8

After that, the DT-Hybrid and TAGME predictions are computed. Custom DT-Hybrid produces a DataFrame, named "DT_Hybrid_df", containing the confidence "s$_{i,j}$" in the association of the element Term$_j$ to Term$_i$ (more information are described in "Custom DT-Hybrid algorithm section"). Instead, to get the TAGME predictions, our pipeline groups the couples of terms in several chuncks composed of 100 elements at most, and then send these to TAGME server through http POST requests. The server computes the relatedness "r$_{i,j}$" value for each received couples (Term$_i$, Term$j$) and returns the valuess back. The relatedness value is in the range [0,1] and expresses how much two terms are semantically related to each other. Zero value means nothing relationships between them. All the obtained relatednesses are collected within the "TAGME_Relatedness" DataFrame.

DT_Hybrid_df		
Term$_i$	Term$j$	s$_{i,j}$
t_0	t_1	$s_{0,1}$
t_0	t_2	$s_{0,2}$
...	...	...
t_x	t_y	$s_{x,y}$

DataFrame 4.9

TAGME_Relatedness		
Term$_i$	Term$j$	r$_{i,j}$
t_0	t_1	$r_{0,1}$
t_0	t_2	$r_{0,2}$
...	...	...
t_x	t_y	$r_{x,y}$

DataFrame 4.10

To complete the BioTAGME score computing, Documents_Term$_i$_Term$_j$, DT_Hybrid_df and TAGME_Relatedness DataFrame are joined together on the columns Term$_i$, Term$_j$. And then, the “BioTAGME score” is computed through the following expression:

$$BioTG_{i,j} = r_{i,j} * s_{i,j} * f_{i,j} \tag{4.1}$$

4.3.6 Network Construction

As soon as the documents have been annotated, and the prediction procedure has been completed, the last procedure of the pipeline is to build the knowledge graph containing logical or physical relationships among biological elements.

For every association Termi-Termj obtained during the prediction procedure, we create three different edges:

- The literature one which represents a biological evidence resulted of laboratory experiments, biological and biophysical processes, etc. Such interactions have been extracted from the relationships DataDrame built through the integration databases module.
- The STRING one which represents the predicted protein-protein associations stored in STRING database. We report this information because our system integrate homo-sapiens protein-protein network of STRING.
- the BioTAGME one that represent the BioTAGME prediction. Both BioTAGME edge and STRING one are marked with the corresponding score value in order to indicate the likelihood of the interaction.

More information about the plotting of the network, motif search and shortest path computing are reported in the pipeline 2 section.

4.3.7 Updating procedure

BioTAGME pipeline uses the annotations of Pubmed documents' title and abstract to predict relationships among biological components obtained during the Annotation procedure. Since many new documents are submitted to Pubmed database every day, it is necessary for both Annotations DataFrame and Documents one to be periodically update.

To achieve this purpose, our pipeline carried out the following tasks: First, it downloads all the PMIDs within an established data range [mindate, maxdate] through a NCBI esearch POST request. "Mindate" usually refers to the last updating date, and it is saved into the "last_update_date.txt" file. Instead, "maxdate" usually is set to the actual one. Maxdate may be configured through "biotagme_config.xml" file located under the project src directory.

Once the PMIDs list has been obtained, the updating module tries to download the title and abstract of these PMIDs using the NCBI efetch API. Thus, at beginnig the list is partitioned into chunks of "retmax" size, and then several chank-based NCBI efetch post requests are generated and sent to Pubmed server in order to obtain the data required. Note that NCBI doesn't report a maximum value for the retmax parameter (numer of PMIDs that can be processed by a single request), especially when a POST request is used. However, we suggest to keep this value under 10000 for the following considerations:

- Since the NCBI recommends to send three requests per second at most, our pipeline just employes the SPARK cluster driver to carry out the updating task to avoid that the application would be banned (all the cluster nodes have different private IP address, but same public one). Moreover, the driver available RAM memory could be limited resulting so in possible out of memory error. So, we need to maintains the chunk size to a low value.

- If the number of articles per request is huge, then the submitted job is considered as a heavy job. Therefore, NCBI recommends to execute these jobs

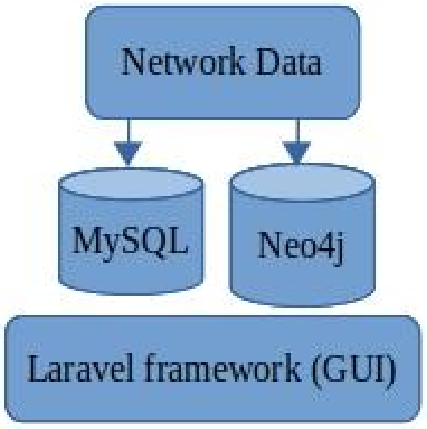


Figure 4.6: Pipeline two structure

during the weekends.

As soon as the Documents title and abstract have been download, the annotation, prediction and network construction procedure will be execute in order to update the edges and nodes of the knowledge graph.

4.4 Pipeline two

The second pipeline (Fig 4.6) has been implemented for importing the knowledge network into Neo4j graph database through the appropriate file (generated by the first pipeline), and for running queries (using our User Interface) on the graph in order to get either the echo network of a biological element, or to compute the shortest path between two ones. This pipeline employs the Laravel model-view-controller, and the React native framework to implement the logic and the web pages components, respectively. In this section we will describe all the main modules of the pipeline two.

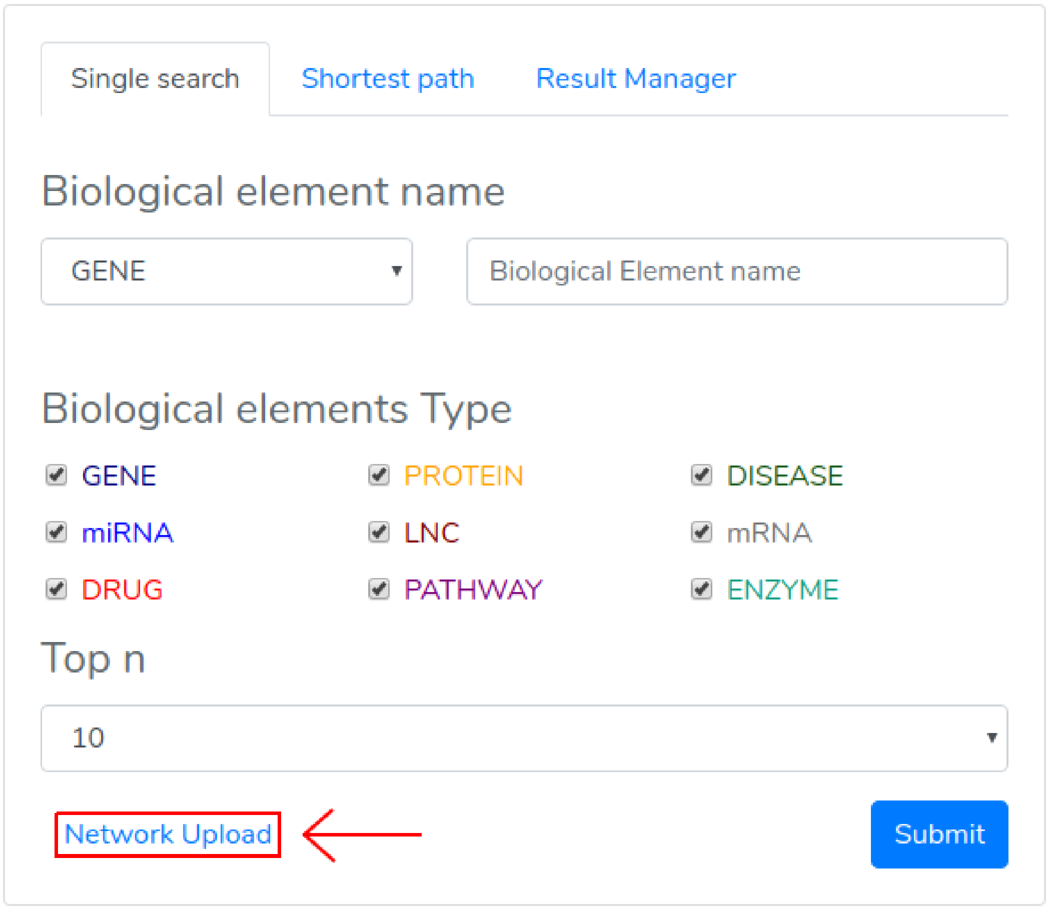


Figure 4.7: Biological element search panel

4.4.1 Network import module

Through the “biological element search” panel (Fig. 4.7), a user may access to the uploading section by clicking on the “network files upload” link.

Such section includes three consecutive phases:

- The fist one is the “authentication phase” to ensure that only authorized users may execute the import procedure. The user have to specify email and password (see Fig. 4.8) provided by the application manager.
- If the authentication is successfully completed, the “files selection phase” is enabled (see Fig. 4.9(a)). During this phase, the user selects the “nodes.csv”

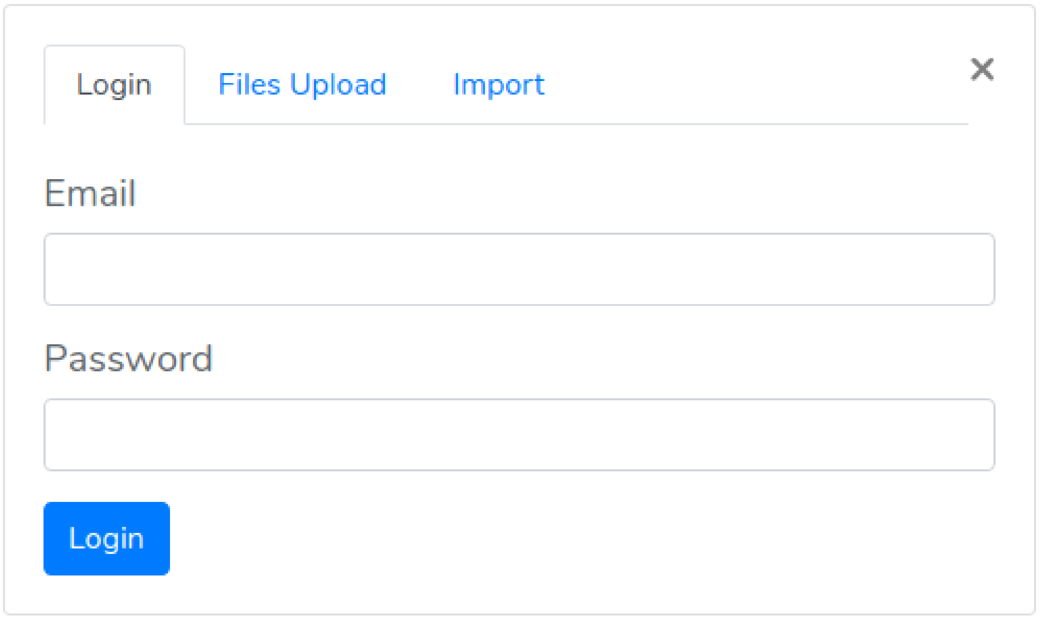


Figure 4.8: Authentication Panel

and “edges.csv” files containing network components, or the “BioIDs _Wiki-IDs.xcv”, “Name_Aliases.csv”, “Wiki_Titles.csv” files about biological elements aliases, and Wikipedia pages information respectively. It is not possible to upload the five files in the same transaction due to their large size. The “Name_Aliases.csv” file will enables the user to make query based on biological element aliases/name. Instead the“Wiki_Titles.csv” allow the pipeline to create links toward Wikipedia pages and PubMed articles. Once all the necessary files have been selected, the user send them to the BioTagME server clicking on submit button (see Fig. 4.9(b)).

Since the size of the files is large (GB), we used the “Pion” library [90] in order to split the file into small chunks (clint side), and re-assemble them as soon as these are correctly received (server side).

- Upon all the file have been successfully received, the “import phase” is enabled. It shows a summary (Fig. 4.10) about the uploaded files so that the user may check any mistakes in the files selection. If everything is correct,then she

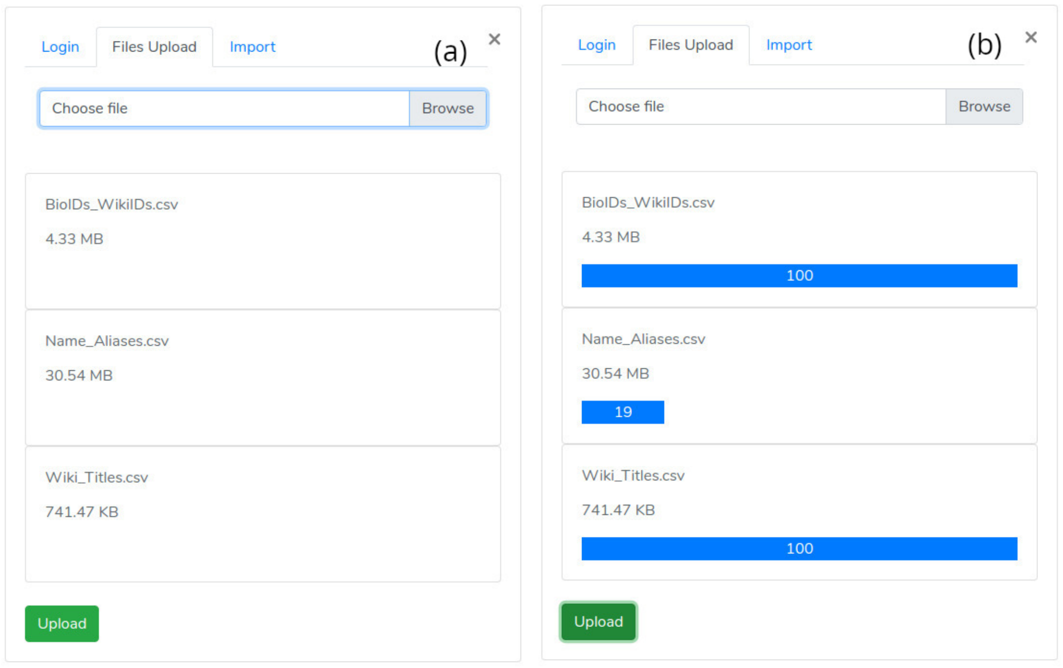


Figure 4.9: File selector panel

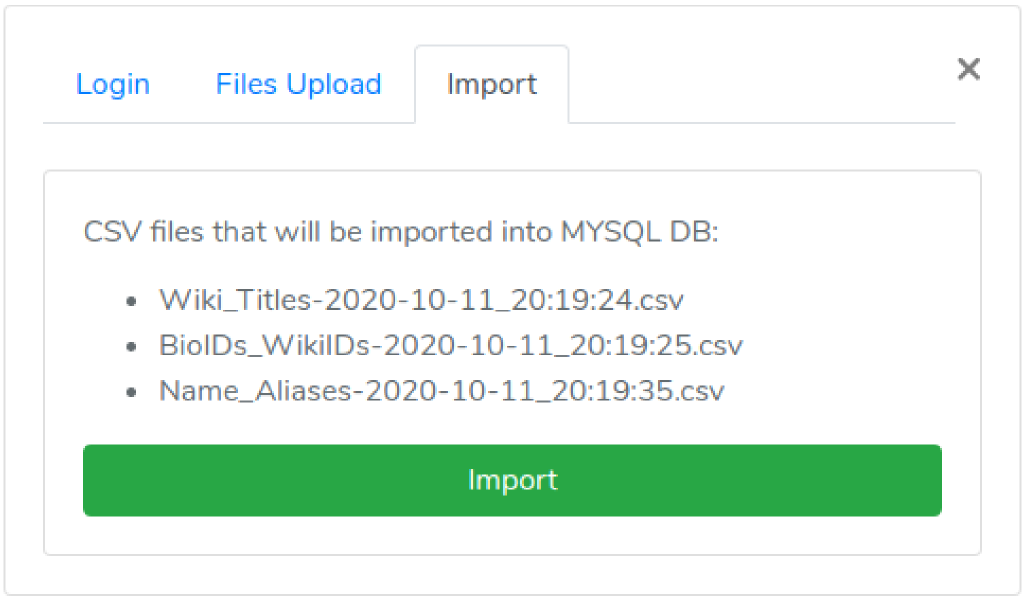
may trigger the execution of the following two Laravel jobs by clicking on the import button:

- mysql_import one transforms the “BioIDs_WikiIDs.xcv”, “Name _Aliases.csv”, “Wiki_Titles.csv” csv file in mysql tables.
- neo4j_impot one convert the “nodes.csv” and “edges.csv header” in a compatible neo4j one at beginning, and then imports the network in no4j using neo4j_admin tool.

If there are not errors, the user will receive a successfully notification as soon as the processes are completed.

4.4.2 Searching module

Once the network has been imported within the neo4j graph database, a user may execute several queries, through our User Interface “GUI” (Fig.4.11), to get the desired information. Such GUI is composed of the following two panels: Searching



CSV files that will be imported into MYSQL DB:

- Wiki_Titles-2020-10-11_20:19:24.csv
- BioIDs_WikiIDs-2020-10-11_20:19:25.csv
- Name_Aliases-2020-10-11_20:19:35.csv

Figure 4.10: Import panel

panel (Fig.4.11.a) and Graph one (Fig.4.11.b).

Searching panel

The searching panel is used for setting the query parameters on the basis of the selected menu: i. Echo network or ii. shortest path.

- When the **Echo network** is selected, a user may search the **Echo Network** of a **biological element** "be$_i$". Therefore, she needs to provide the type and name of the biological element to analyze (Fig.4.12.a, red rectangle), and the type of the other elements (Fig.4.12.a, orange rectangle) to include within the echo-network. To avoid an excessive size of the network, a maximum nodes number have to be configured (from 10 to 50 with a step size of 10) through the "Top n" section (Fig.4.12.a, green rectangle). Upon all the required parameters have been filled in, the searching process is triggered by clicking on Submit button. This process transform the specified form parameters in a "cypher query" which will carry out the following two actions: The first one looks

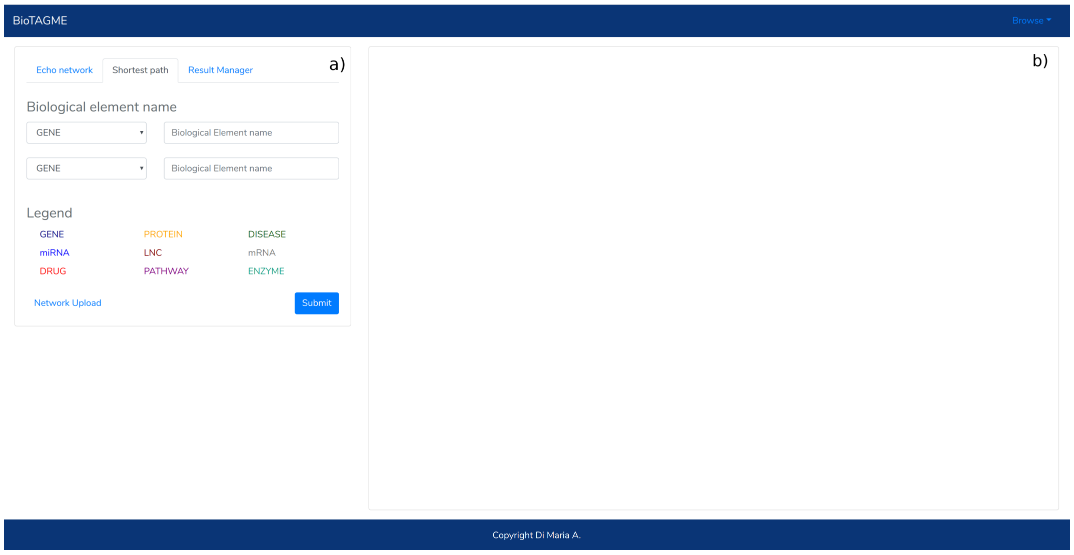


Figure 4.11: BioTAGME Homepage

for the “Top n” nodes having one or more links from/to “be$_i$”. Instead, the second one return all the edges among the “Top n” nodes previously selected. This query is sent to neo4j service by a post request in order to get the desired information in a JSON form.

- Instead, when the **shortest panel** is selected (Fig.4.12.b), a user looks for the shortest path between two biological elements. Therefore, first she needs to provide the type and name of the source “*el_src*” and destination “*el_dst*” elements (Fig.4.12.b, red rectangle), and then may run the appropriate process by clicking on the submit button. As the process starts, all the form parameters are transformed in a cypher query which is mainly based on a neo4j procedure for the shortest path computing. This query will be sent to Neo4j server in order to get the edges and intermediate nodes connecting “*el_src*” with “*el_dst*”.

Graph panel

The **graph panel** (Fig.4.11.b) plots either the “echo-network” or the “shortest path” by employing the CytoscapeJS library [91]. All the edges are interactive. Thus, if

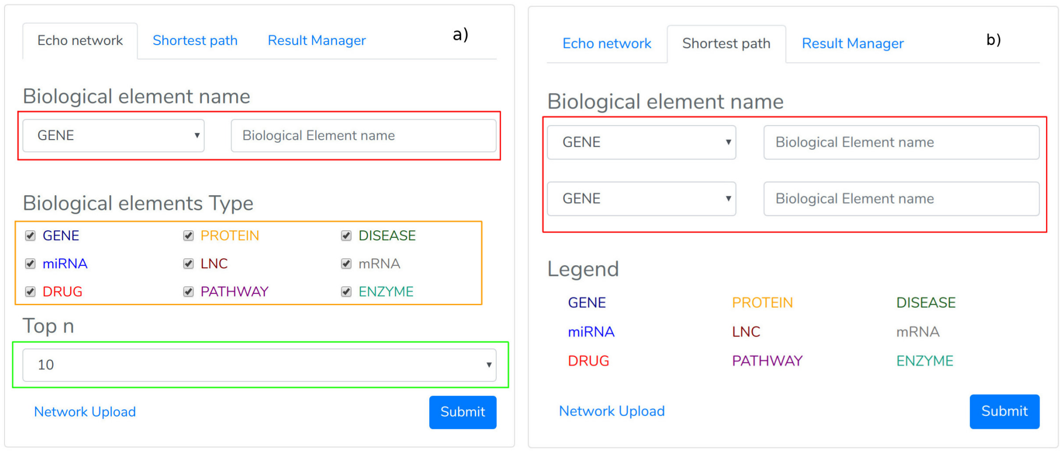


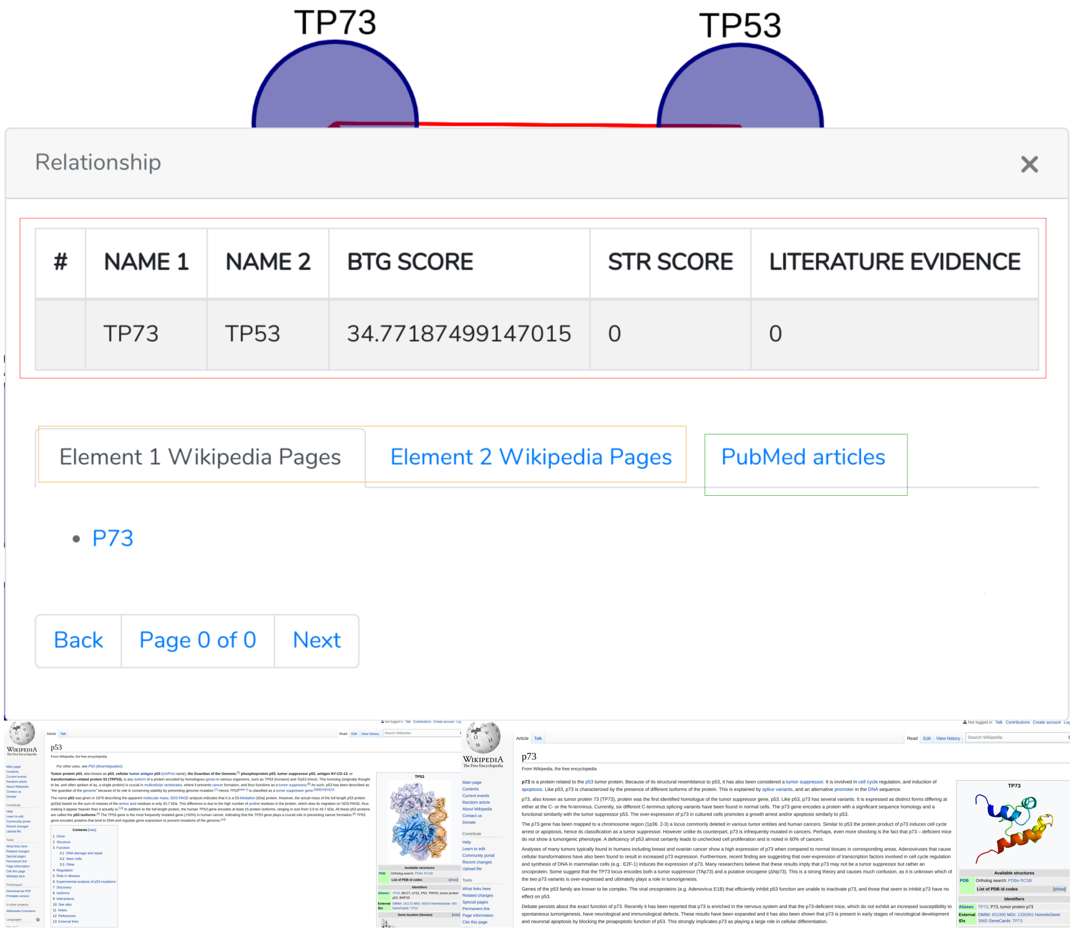
Figure 4.12: BioTAGME Homepage

a user clicks on one of that, then a new windows (Fig.4.13) will be shown. Such a window is composed of the following data:

- A table (Fig.4.13, red rectangle) containing the name of the source and destination nodes as well as the BioTagME and STRING scores. In addition, the last column of the table report the literature evidence (1 if the association is in one of the literature databases at least, 0 otherwise).

- A navigation panel with three different options. The first two (Fig.4.13, orange rectangle) show several links among Wikipedia pages and source or destination node, respectively. Instead, the last one (Fig.4.13, green rectangle) shows several links toward the PubMed articles containing information about the selected association.

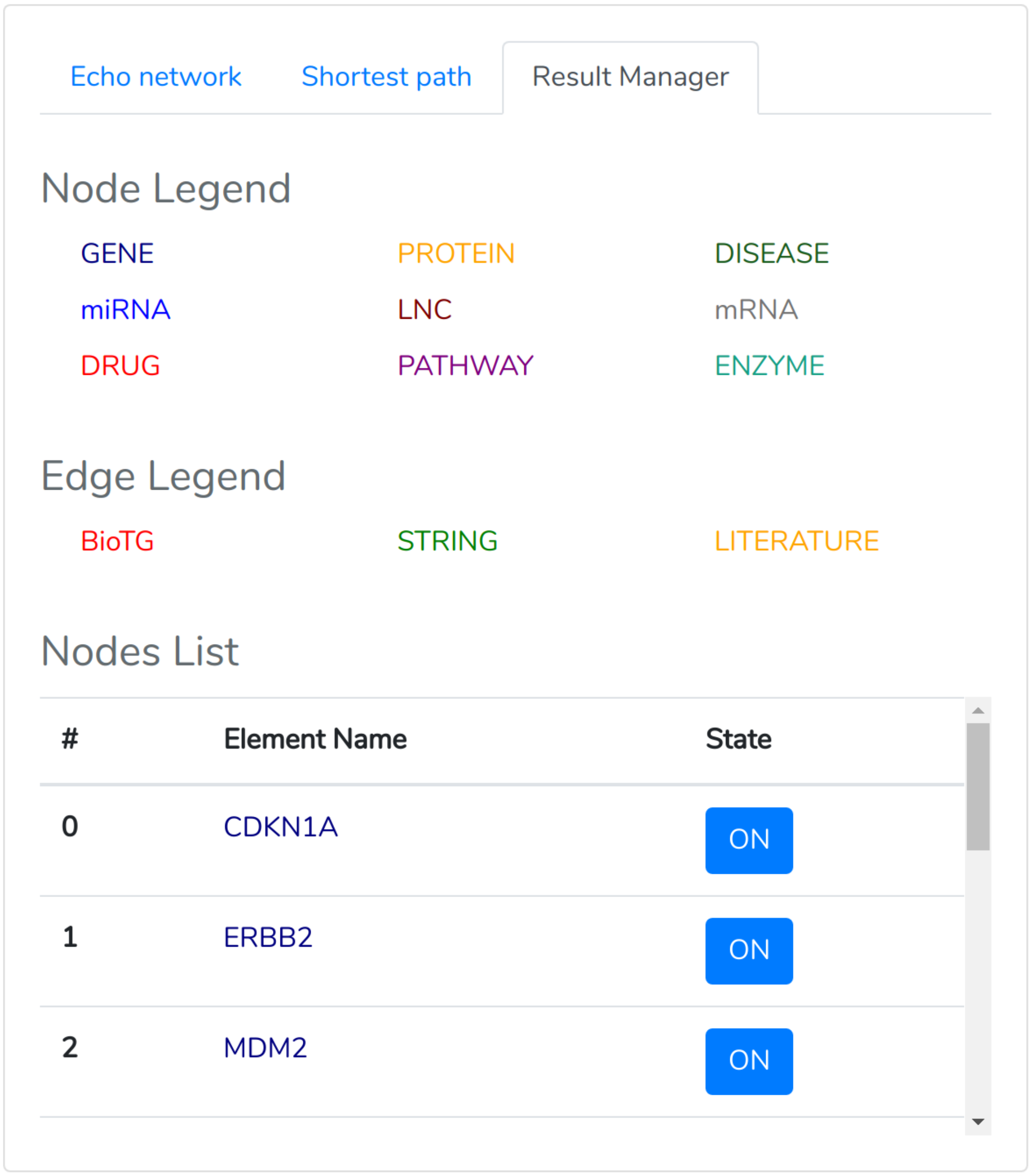
Once the network or the shortest path has been plotted, the **result menu** of the panel (Fig.4.11.a) is appropriately configured. Such a menu reports the node and edge legend, and a table listing all the plotted node names (Fig.4.14). Through this table, the user may remove or add back one or more nodes by clicking on the corresponding button.



#	NAME 1	NAME 2	BTG SCORE	STR SCORE	LITERATURE EVIDENCE
	TP73	TP53	34.77187499147015	0	0

- P73

Figure 4.13: Edge window



Node Legend

GENE	PROTEIN	DISEASE
miRNA	LNC	mRNA
DRUG	PATHWAY	ENZYME

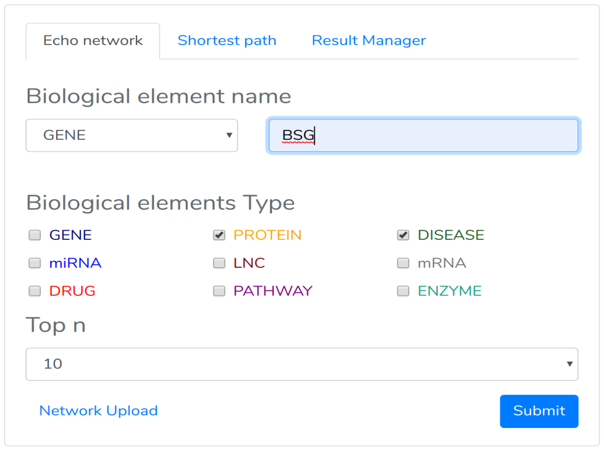
Edge Legend

BioTG	STRING	LITERATURE

Nodes List

#	Element Name	State
0	CDKN1A	ON
1	ERBB2	ON
2	MDM2	ON

Figure 4.14: Result menu



Biological element name

GENE

BSG

Biological elements Type

- [ ] GENE
- [x] PROTEIN
- [x] DISEASE
- [ ] miRNA
- [ ] LNC
- [ ] mRNA
- [ ] DRUG
- [ ] PATHWAY
- [ ] ENZYME

Top n

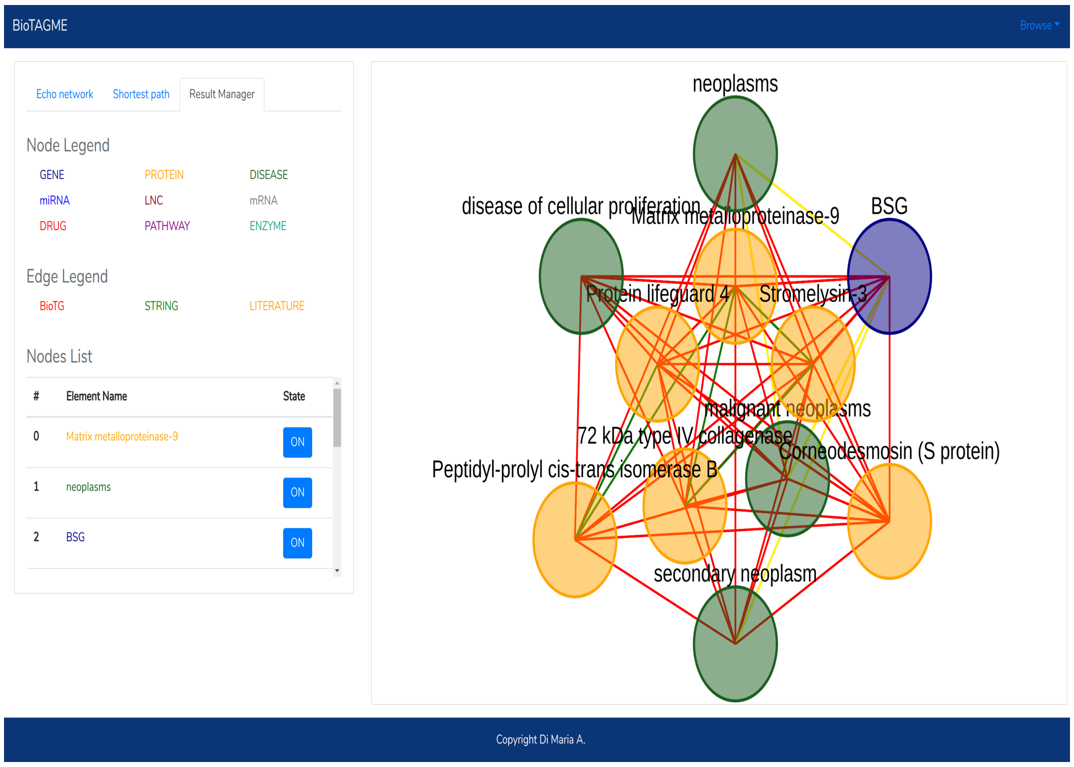
10

Figure 4.15: BSG parameters configuration

4.4.3 Case study

In the proposed case study, we built the echo network of the gene "BSG" considering only diseases and proteins as other element of the network. In addition, we also set the "Top n' value to ten in order to get (from Neo4j) a network having ten nodes at most. All the configuration parameters have been reported in Fig.4.15. How it may be noticed from the Fig.4.16, there are several kind of edges.

- The yellow edges represent biological elements interaction already available in literature, but our framework is not able to infer.
- The yellow edges with red border are similar to the previous one, but in this case our framework is able to predict such interactions, too.
- The green edges instead represent a protein-protein association in STRING. To understand if our framework is able to infer those, we need to click on the edges and check if the BioTG score is not zero.
- Finally, the red edges represent prediction obtained in BioTAGME, but neither found in STRING nor in the actual literature databases employed in BioTAGME.



#	Element Name	State
0	Matrix metalloproteinase-9	ON
1	neoplasms	ON
2	BSG	ON

Figure 4.16: BSG echo network

Chapter 5

NetME

In the previous chapters, we have explained BioTAGME and its functionalities for building a knowledge network starting from all documents' title and abstract in PubMed [36]. The inferred edges among the nodes only represent a possible physical or logical interaction, but at the moment our framework is not able to infer the interaction type due to the following aspects:

- Title and Abstract contains limited information.
- BioTAGME employs two prediction algorithms only for computing the robustness of the interaction. Therefore, we need to include one or more "Sentiment Analysis" techniques [92] in order to infer the type.

Since working with all PubMed Central full-text, and traditional "Sentiment Analysis" methods requires much time and computing resource, we have decided to test such methods in a small subset of full-text. This yielded our second application called NetME (submitted and accepted to the COMPLEX NETWORK 2020, December 1-3, 2020). Thanks to the encouraging results obtained in NetME experiments, we will apply the same NetME methodology in a future BioTAGME second version in order to infer the type of each relationship. It will require a lot of time because we will use all full-text in PubMed as well as titles and abstracts.

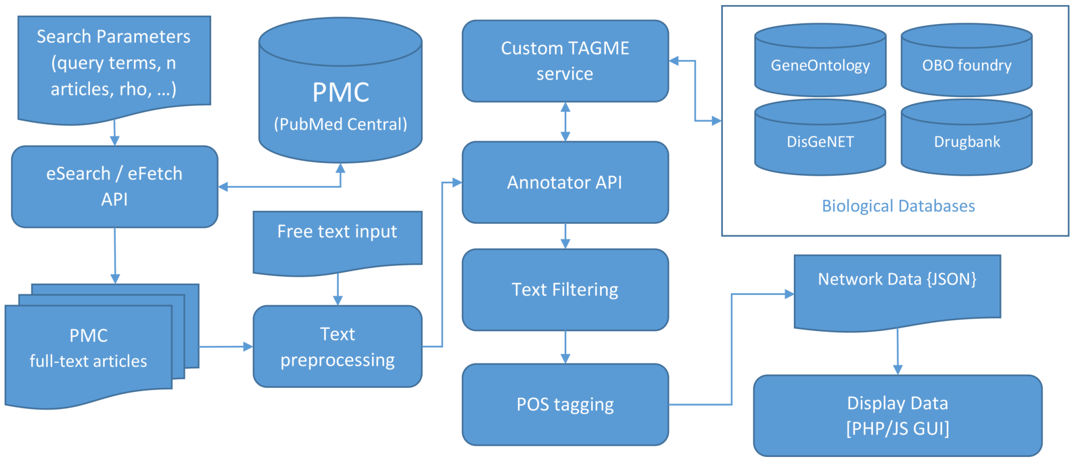


Figure 5.1: NetME pipeline

NetME is a web-based app (publicly available at http://netme.alaimos.com/netme/) which is capable to extract knowledge from a collection of full-text documents. The tool orchestrate two different entities which enable the inference of biological networks:

- The customized TagME version (OntoTagme) which is designed to to extract *biological entities* from a collection of full-text articles obtained through a query to PubMed. These biological entities will represent the nodes of the network;
- A software module developed upon the NLTK library [28] resource to derives relationships (edges).

The final network will contain both directed and undirected edges according to the predictions made by the model. NetME is the first tool which allows to interactively synthesize biological knowledge graphs on-the-fly starting from a query on PubMed.

5.1 NetME methodology

As soon as the user provide a list of searching keywords, the tool send a request to PubMed (using the NCBI Entrez API [37]) in order to obtain the top n PubMed Central full-texts containing the desired keywords. Next, all the nodes of the graph are extracted through our implemented OntoTagme version. Once the nodes have been deduced, the textual analysis is triggered to infer edges between them. The analysis is performed through a methodology developed on top of the Python library NLTK [28]. Solving a syntax analysis problem for a sentence means looking for predefined syntactic forms which branch out from the single words. The main syntactic form is the sentence (S) which contains noun phrases (NP) or verb phrases (VP) that in turn are formed by further elementary syntactic forms such as nouns (N), verbs (V), determiners (DET), etc. Two approaches are available for the syntactic analysis of sentences: top-down and bottom-up.

- In the top-down approach the syntactic form of a sentence is determined by starting from the whole sentence. This allows to predict several phrase constituents that can be recursively analyzed until the smallest units are reached. However, this approach could lead infinite loops.

- Insead, in the bottom-up parsing approach the objects represented by the smallest units of text together with their syntactic class are recognized at first. Then, it tries to find and combine sequences of words and phrases until the whole sentence is reached. This approach does not guarantee that a valid parsing will be found for the input even if it exists. Moreover, no checking is performed to establish if the language derived entity is globally consistent with the grammar.

Our approach is based on a left-corner parser, which integrates both the bottom-up and the top-down approaches. First, a left-corner parser pre-processes the context-free grammar to build a two-columns table. The first one holds a non-terminal category (basically are Sentences), instead, the second one holds the collection of possible left corners of not-terminal category (basically they are VP, NP, etc). Next, our approach parses each phrase higher syntactic forms, and filters the results starting from the smallest text units. This approach should overcome the shortcomings of both parsing methodologies. The procedure will yield all PartOf-Speech (POS) of a sentence (S) that are nouns (N) linked through verbal relations (V).

Finally, the network is built compared the annotation terms (network nodes) obtained by the OntoTAGME algorithm with the the elements extracted by the left-corner parser. If there is a correspondence between two entities, then an edge of the graph will connect such entities. More precisely, through a PubMed search, we obtain a list of n full-text papers $[p_1, p_2, \cdots, p_n]$, that are divided in grammatical periods. Therefore, given an article p_i we derive a set of m sentences $s_{i,1}, s_{i,2}, \cdots, s_{i,m_i}$. Then, for each sentence, we employ OntoTagme computing a set of spots $n_1, ..., n_z$. These spots will be the nodes in our final network. To define the edges in such a network, we tokenize all sentences and perform a syntactic analysis. This analysis will yield, for each sentence $s_{i,j}$ of a paper i, a set of labelled tokens $lt_{i,j,1}$, $lt_{i,j,2} \cdots, lt_{i,j,k_i}$ where each token is a pair $(token, POS)$. Irrelevant POS are therefore removed (stop-words, URLs, etc.), keeping only useful verb forms and spots extracted through annotation procedure.

In our final network, each edge $e = (a, b)$ is weighted as:

$$w(e) = \Sigma_{k=1}^{z} \tfrac{1}{e_k}/z \tag{5.1}$$

where z is the number of sentences (in all the papers) in which an edge between (a, b) is reported and e_k the number of edges between a and b in the *k-th* sentence. Indeed, nodes a and b may be connected more the once in the same sentence.

5.2 NetME User Interface

NETME consists of a front-end developed in PHP and Javascript. Furthermore, the Network rendering is performed with the CytoscapeJS library [91]. The major functionalities made available through the User Interface are essentially two:

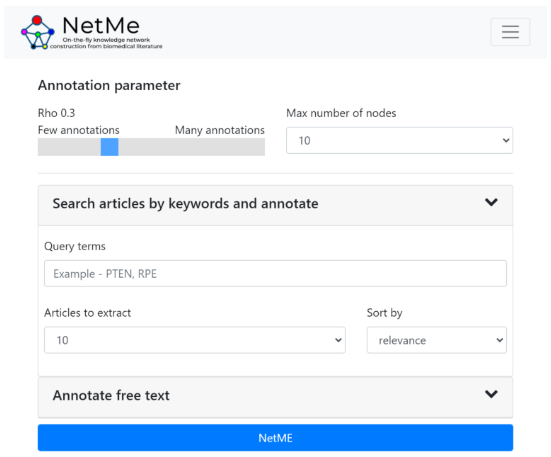
- **Query-based network annotation** enables a user to provide a list of keywords which are employed to run a query on PubMed (Fig.5.2a). The top resulting papers are retrieved and the network inference procedure is performed. Several parameters can be defined by the user: (i) OntoTagme ρ to increase or reduce the number of spots, (ii) the maximum number of network nodes, (iii) the number of top article to retrieve from PubMed, and (iv) the criteria used to sort papers (relevance or date).

- **User-provided free-text network annotation** enables a user to provided a text.

Once the user starts the analysis, the directed network is generated, and all inference details are provided in three tables containing (i) the list of extracted papers, (ii) the list of annotations, and (iii) the list of edges together with their weight (Fig.5.2b).

5.3 Case study

In the proposed case study, we built a network based on the "SRC" (Proto-oncogene tyrosine-protein kinase) concept. We choose to build a network of 200 nodes from the Top-20 PubMed articles. ρ has been set to 0.3 which are suitable to perform an interactive test. We made a comparison considering HetioNet as competitor, and SemRep as groundtruth. For each node and edge in SemRep results, we checked whether NETME was able to infer it. The same analysis has been done using HetioNet. In this analysis edge labels were ignored. To evaluate the quality of our prediction the "precision" and "recall" have been computed (). Our analysis, although preliminary, clearly shows that NETME has a comparable precision with HetioNet in terms of inferred nodes and an higher recall. Concerning inferred edges, HetioNet precision is slightly higher although NETME recall is much higher.


NetMe
On-the-fly knowledge network construction from biomedical literature

Annotation parameter

Rho 0.3

Few annotations — Many annotations

Max number of nodes: 10

Search articles by keywords and annotate

Query terms: Example - PTEN, RPE

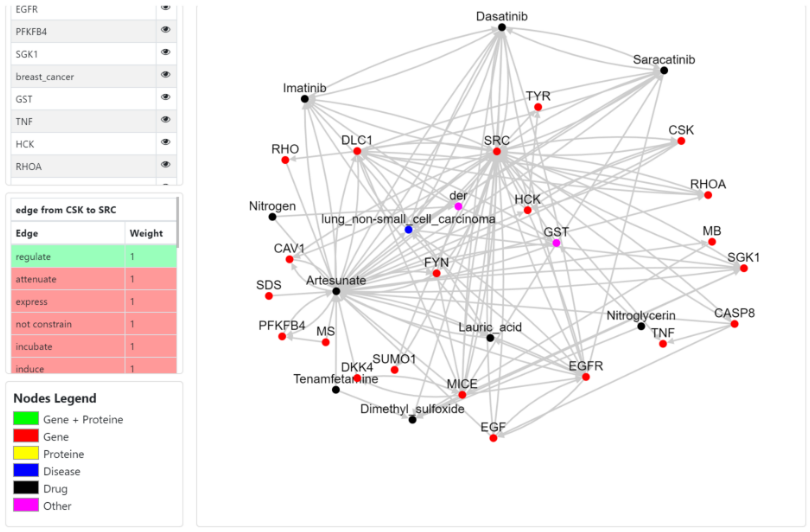
Articles to extract: 10

Sort by: relevance

Annotate free text

NetME

(a)



EGFR
PFKFB4
SGK1
breast_cancer
GST
TNF
HCK
RHOA

edge from CSK to SRC

Edge	Weight
regulate	1
attenuate	1
express	1
not constrain	1
incubate	1
induce	1

Nodes Legend

🟩	Gene + Proteine
🟥	Gene
🟨	Proteine
🟦	Disease
⬛	Drug
🟪	Other

(b)

Figure 5.2: NetME web interface in (a), generated network in (b)

	NETME	HetioNet	SemRep (Ground truth)
Detected nodes	200	112	189
Valid nodes (TP)	123	61	189
Wrong nodes (FP)	77	51	–
Missing nodes (FN)	66	128	–
Precision	61.5%	54.5%	–
Recall	65.1%	32.3%	–
Detected edges	495	56	292
Valid edges (TP)	178	34	292
Wrong edges (FP)	317	22	–
Missing edges (FN)	114	258	–
Precision	56.2%	60.7%	–
Recall	61%	11.6%	–

DataFrame 5.1: Netme performance results

Chapter 6

Conclusions

In this thesis we introduced BioTAGME and NetME, two frameworks for the inference of knowledge graphs from scientific documents gathered through PubMed. The differences between the two approaches are:

- The actual BioTAGME version works on all PubMed documents title and abstract. Instead, NetME works on a collection of full-texts either submitted by the user or extracted from PubMed Central through a user-specified query.

- BioTAGME builds a graph on all the documents titles and abstract in PubMed up to the generation date of such graph. Therefore, it is necessary to periodically update the graph in order to add new nodes and predicted edges resulting from the analysis and elaboration of new documents. NetME instead downloads (in real time) the top n (from 10 to 200) full-texts based on the user-specified context (i.e breast cancer disease, etc), and then generates the graph starting from a partial knowledge content of the just downloaded full-texts. Therefore, the BioTAGME knowledge graph is static instead the NetME one is dynamic and content dependent.

- BioTAGME employees the traditional TagME version, based on Wikipedia corpus, to transform a document in a list of keywords. MetME instead employees a custom TagME version based on Gene and Disease ontologies. Such a custom version has been implemented to mitigate the disambiguation error.

- BioTAGME is able to predict a possible relationships between two biological elements combining the bipartite graph methodology with the relatedness among Wikipedia pages. This allows to estimate the strength of the relationships, but not the type. For example, if we consider two genes A and B, and A up-regulates B, our methodology is able to predict the interaction between A and B, but not the up-regulate concept. Instead, NetME is based on sentimental analysis methodologies able to infer both the strength and type of a relationship.

Future works will include:

- The improvement of BioTAGME prediction by employing all PubMed central full-texts, and sentimental analysis methodologies.
- The improvement of NetME performance and predictions through the extension of the ontology used in OntoTagme, the temporary storage of networks for fast computation of updates, and the implementation of word2vect and/or doc2vect to predict possible missing relationships.

Bibliography

[1] Albert-László Barabási, Natali Gulbahce, and Joseph Loscalzo. Network medicine: a network-based approach to human disease. *Nature Reviews Genetics*, 12(1):56–68, December 2010.

[2] X. Ma and L. Gao. Biological network analysis: insights into structure and functions. *Briefings in Functional Genomics*, 11(6):434–442, November 2012.

[3] R. Resnick, D. Halliday, and K. S. Krane. *Fisica*, volume 1. Casa Editrice Ambrosiana, Milano, 2003.

[4] Beck J. Report from the field: PubMed central, an XML-based archive of life sciences journal articles. In *Proceedings of the International Symposium on XML for the Long Haul: Issues in the Long-term Preservation of XML*. Mulberry Technologies, Inc.

[5] Ginsparg P. *arXiv*. Available at "`https://arxiv.org`".

[6] *bioRxiv*. Available at "`https://www.biorxiv.org`".

[7] Lambrix P., Tan H., Jakoniene V., and Strömbäck L. Biological ontologies. In *Semantic Web*, pages 85–99. Springer US.

[8] Cohen A. M. A survey of current work in biomedical text mining. *Briefings in Bioinformatics*, 6(1):57–71, January 2005.

[9] Krallinger M., Erhardt R. A., and Valencia A. Text-mining approaches in molecular biology and biomedicine. *Drug Discovery Today*, 10(6):439–445, March 2005.

[10] Nicholson D. N. and Greene C. S. Constructing knowledge graphs and their biomedical applications. *Computational and Structural Biotechnology Journal*, 18:1414–1428, 2020.

[11] Slater T. Recent advances in modeling languages for pathway maps and computable biological networks. *Drug Discovery Today*, 19(2):193–198, February 2014.

[12] McBride B. The resource description framework (RDF) and its vocabulary description language RDFS. In *Handbook on Ontologies*, pages 51–65. Springer Berlin Heidelberg, 2004.

[13] Kim J., Wang Y., Fujiwara T., Okuda S., Callahan T. J., and K. B. Cohen. Open agile text mining for bioinformatics: the PubAnnotation ecosystem. *Bioinformatics*, 35(21):4372–4380, April 2019.

[14] Wei C., Allot A., Leaman R., and Lu Z. PubTator central: automated concept annotation for biomedical full text articles. *Nucleic Acids Research*, 47(W1):W587–W593, May 2019.

[15] Jyoti Soni, Ujma Ansari, Dipesh Sharma, and Sunita Soni. Predictive data mining for medical diagnosis: An overview of heart disease prediction. *International Journal of Computer Applications*, 17(8):43–48, March 2011.

[16] Asghar M. Z., Khan A., Ahmad S., Qasim M., and Khan I. A. Lexicon-enhanced sentiment analysis framework using rule-based classification scheme. *PLOS ONE*, 12(2):e0171649, February 2017.

[17] Szklarczyk D., Morris J. H., Cook H., Kuhn M., Wyder S., Simonovic M., Santos A., Doncheva N. T., Roth A., Bork P., Jensen L. J., and von Mering C.

The STRING database in 2017: quality-controlled protein–protein association networks, made broadly accessible. *Nucleic Acids Research*, 45(D1):D362–D368, October 2016.

[18] Himmelstein D. S. and Baranzini S. E. Heterogeneous network edge prediction: A data integration approach to prioritize disease-associated genes. *PLOS Computational Biology*, 11(7):e1004259, July 2015.

[19] Rindflesch T. C. and Fiszman M. The interaction of domain knowledge and linguistic structure in natural language processing: interpreting hypernymic propositions in biomedical text. *Journal of Biomedical Informatics*, 36(6):462–477, December 2003.

[20] Ferragina P. and Scaiella U. TAGME. In *Proceedings of the 19th ACM international conference on Information and knowledge management - CIKM '10*. ACM Press, 2010.

[21] Schriml L. M., Mitraka E., Munro J., Tauber B., Schor M., Nickle L., Felix V., Jeng L., Bearer C., Lichenstein R., Bisordi K., Campion N., Hyman B., Kurland D., Oates C. P., Kibbey S., Sreekumar P., Le C., Giglio M., and Greene C. Human Disease Ontology 2018 update: classification, content and workflow expansion. *Nucleic Acids Research*, 47(D1):D955–D962, 11 2018.

[22] Wishart D. S., Feunang Y. D., Guo A. C., Lo E. J., Marcu A., Grant J. R., Sajed T., Johnson D., Li C., Sayeeda Z., Assempour N., Iynkkaran I., Liu Y., Maciejewski A., Gale N., Wilson A., Chin L., Cummings R., Le D., Pon A., Knox C., and Wilson M. DrugBank 5.0: a major update to the DrugBank database for 2018. *Nucleic Acids Research*, 46(D1):D1074–D1082, November 2017.

[23] Gray K. A., Seal R. L., Tweedie S., Wright M. W., and Bruford E. A. A review of the new HGNC gene family resource. *Human Genomics*, 10(1), February 2016.

[24] Zaharia M., Xin R. S., Wendell P., Das T., Armbrust M., Dave A., Meng X., Rosen J., Venkataraman S., Franklin M. J., Ghodsi A., Gonzalez J., Shenker S., and Stoica I. Apache spark. *Communications of the ACM*, 59(11):56–65, October 2016.

[25] Gene Ontology Consortium. The gene ontology (GO) database and informatics resource. *Nucleic Acids Research*, 32(90001):258D–261, January 2004.

[26] Piñero j., Ramírez-Anguita J. M., Saüch-Pitarch J., Ronzano F., Centeno E., Sanz F., and Furlong L. I. The DisGeNET knowledge platform for disease genomics: 2019 update. *Nucleic Acids Research*, 48(D1):D845–D855, 11 2019.

[27] Smith B., Ashburner M., Rosse C., Bard J., Bug W., Ceusters W., Goldberg L. J., Eilbeck K., Ireland A., Mungall C. J., Leontis N., Rocca-Serra P., Ruttenberg A., Sansone S., Scheuermann R. H., Shah N., Whetzel P. L, and Lewis S. The OBO foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, 25(11):1251–1255, November 2007.

[28] Loper E. and Bird S. NLTK. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics* -. Association for Computational Linguistics, 2002.

[29] Drieger P. Semantic network analysis as a method for visual text analytics. *Procedia - Social and Behavioral Sciences*, 79:4–17, June 2013.

[30] Huo C., Wu H., Xiao J., Meng D., Zou S., Wang M., Qi P., Tian H., and Hu Y. Genomic and bioinformatic characterization of mouse mast cells (p815) upon different influenza a virus (h1n1, h5n1, and h7n2) infections. *Frontiers in Genetics*, 10, June 2019.

[31] Chen H., Ding L., Wu Z., Yu T., Dhanapalan L., and Chen J. Y. Semantic web for integrated network analysis in biomedicine. *Briefings in Bioinformatics*, 10(2):177–192, March 2009.

[32] Jens D. and Marc J. Semantic knowledge graph embeddings for biomedical research: Data integration using linked open data, 09 2019.

[33] Xiaoke M. and Lin G. Biological network analysis: insights into structure and functions. *Briefings in Functional Genomics*, 11(6):434–442, 11 2012.

[34] David Nettleton. Data representation. In *Commercial Data Mining*, pages 49–66. Elsevier, 2014.

[35] Sören A., Christian B., Georgi K., Jens L., Richard C., and Zachary I. Dbpedia: A nucleus for a web of open data. In *Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference*, ISWC'07/ASWC'07, page 722–735, Berlin, Heidelberg, 2007. Springer-Verlag.

[36] Williamson P. O. and Minter C. I. J. Exploring PubMed as a reliable resource for scholarly communications services. *Journal of the Medical Library Association*, 107(1), January 2019.

[37] Sayers E. *Entrez Programming Utilities Help*. Available at `https://www.ncbi.nlm.nih.gov/books/NBK25501/`.

[38] Li F., Zhang M., Fu G., and Ji D. A neural joint model for entity and relation extraction from biomedical text. *BMC Bioinformatics*, 18(1), March 2017.

[39] Lancia F. Word co-occurrence and similarity in meaning. 01 2007.

[40] Qi L., Meng J., Xikun Z., Meng Q., Timothy H., Jing G., and Jiawei H. Truepie: Discovering reliable patterns in pattern-based information extraction.

In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 1675–1684, New York, NY, USA, 2018. Association for Computing Machinery.

[41] Sidey-Gibbons J. A. M. and Sidey-Gibbons C. J. Machine learning in medicine: a practical introduction. *BMC Medical Research Methodology*, 19(1), March 2019.

[42] Wang Q., Garrity G. M., Tiedje J. M., and Cole J. R. Naive bayesian classifier for rapid assignment of rRNA sequences into the new bacterial taxonomy. *Applied and Environmental Microbiology*, 73(16):5261–5267, June 2007.

[43] Azar A. T. and El-Metwally S. M. Decision tree classifiers for automated medical diagnosis. *Neural Computing and Applications*, 23(7-8):2387–2403, October 2012.

[44] Tang Y., Jin B., and Zhang Y. Granular support vector machines with association rules mining for protein homology prediction. *Artificial Intelligence in Medicine*, 35(1-2):121–134, September 2005.

[45] Fei W., Peter Casalino, L., and Dhruv K. Deep Learning in Medicine—Promise, Progress, and Challenges. *JAMA Internal Medicine*, 179(3):293–294, 03 2019.

[46] S. Levine, Wang J. M., Haraux A., Popović Z., and Koltun V. Continuous character control with low-dimensional embeddings. *ACM Transactions on Graphics*, 31(4):1–10, August 2012.

[47] J. Cussens. Part-of-speech tagging using progol. In *Inductive Logic Programming*, pages 93–108. Springer Berlin Heidelberg, 1997.

[48] Havaei M., Davy A., Warde-Farley D., Biard A., Courville A., Bengio Y., Pal C., Jodoin P., and Larochelle H. Brain tumor segmentation with deep neural networks. *Medical Image Analysis*, 35:18–31, January 2017.

[49] Tajbakhsh N., Shin J. Y., Gurudu S. R., Hurst R. T., Kendall C. B., Gotway M. B., and Liang J. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE Transactions on Medical Imaging*, 35(5):1299–1312, 2016.

[50] Huang M., Liu J., and Zhu X. GeneTUKit: a software for document-level gene normalization. *Bioinformatics*, 27(7):1032–1033, February 2011.

[51] Zheng G., Wang H., Wei C., and Li Y. iGepros: an integrated gene and protein annotation server for biological nature exploration. *BMC Bioinformatics*, 12(S14), December 2011.

[52] Wei C., Kao H., and Lu Z. SR4gn: A species recognition software tool for gene normalization. *PLoS ONE*, 7(6):e38460, June 2012.

[53] Leaman R., Dogan R. I., and Lu Z. DNorm: disease name normalization with pairwise learning to rank. *Bioinformatics*, 29(22):2909–2917, August 2013.

[54] Wei C.-H., Harris B. R., Kao H.-Y., and Lu Z. tmVar: a text mining approach for extracting sequence variants in biomedical literature. *Bioinformatics*, 29(11):1433–1439, April 2013.

[55] Zhou X., Zhang X., and Hu X. MaxMatcher: Biological concept extraction using approximate dictionary lookup. In *Lecture Notes in Computer Science*, pages 1145–1149. Springer Berlin Heidelberg, 2006.

[56] Humphreys B. L., Lindberg D. A. B., Schoolman H. M., and Barnett G. O. The unified medical language system: An informatics research collaboration. *Journal of the American Medical Informatics Association*, 5(1):1–11, January 1998.

[57] Qi C. and Guang-chun G. A corpus-based lexical study on frequency and distribution of coxhead's AWL word families in medical research articles (RAs). *English for Specific Purposes*, 26(4):502–514, January 2007.

[58] Cutting D., Kupiec J., Pedersen J., and Sibun P. A practical part-of-speech tagger. In *Proceedings of the third conference on Applied natural language processing* -. Association for Computational Linguistics, 1992.

[59] Aronson A. R. and Lang F. An overview of MetaMap: historical perspective and recent advances. *Journal of the American Medical Informatics Association*, 17(3):229–236, May 2010.

[60] Rindflesch T. C. Integrating natural language processing and biomedical domain knowledge for increased information retrieval effectiveness. In *Proceedings of the 5th Annual Dual-use Technologies and Applications Conference*, pages 260–5, 1995.

[61] Hoyt C. T., Domingo-Fernández D., and Hofmann-Apitius M. BEL commons: an environment for exploration and analysis of networks encoded in biological expression language. *Database*, 2018, January 2018.

[62] Orchard S., Kerrien S., Abbani S., Aranda B., Bhate J., Bidwell S., Bridge A., Briganti L., Brinkman F. S. L., Cesareni G., Chatr-aryamontri A., Chautard E., Chen Ca., Dumousseau M., Goll J., Hancock R. E. W., Hannick L. I., Jurisica I., Khadake J., Lynn D. J., Mahadevan U., Perfetto L., Raghunath A., Ricard-Blum S., Roechert B., Salwinski L., Stümpflen V., Tyers M., Uetz P., Xenarios I., and Hermjakob H. Protein interaction data curation: the international molecular exchange (IMEx) consortium. *Nature Methods*, 9(4):345–350, March 2012.

[63] Chatr-aryamontri A., Breitkreutz B., Oughtred R., Boucher L., Heinicke S., Chen D., Stark C., Breitkreutz A., Kolas N., O'Donnell L., Reguly T., Nixon J., Ramage L., Winter A., Sellam A., Chang C., Hirschman J., Theesfeld C., Rust J., Livstone M. S., Dolinski K., and Tyers M. The BioGRID interaction database: 2015 update. *Nucleic Acids Research*, 43(D1):D470–D478, November 2014.

[64] Franceschini A., Lin J., von Mering C., and Jensen L. J. SVD-phy: improved prediction of protein functional associations through singular value decomposition of phylogenetic profiles. *Bioinformatics*, 32(7):1085–1087, November 2015.

[65] Mark Needham and Amy E. Hodler. *Graph algorithms: practical examples in Apache Spark and Neo4j*. O'Reilly, 2019.

[66] Croft D., Mundo A. F., Haw R., Milacic M., Weiser J., Wu G., Caudy M., Garapati P., Gillespie M., M. R. Kamdar, Jassal B., Jupe S., Matthews L., May B., Palatnik S., Rothfels K., Shamovsky V., Song H., Williams M., Birney E., Hermjakob H., Stein L., and D'Eustachio P. The Reactome pathway knowledgebase. *Nucleic Acids Research*, 42(D1):D472–D477, 11 2013.

[67] Yin P., Choi Harry M. T., Calvert C. R., and Pierce N. A. Programming biomolecular self-assembly pathways. *Nature*, 451(7176):318–322, January 2008.

[68] Christopher M. D., Mihai S., John B., Jenny F., and David M. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.

[69] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[70] Goddard M.E., Hayes B.J., and Meuwissen T.H.E. Using the genomic relationship matrix to predict the accuracy of genomic selection. *Journal of Animal Breeding and Genetics*, 128(6):409–421, November 2011.

[71] Hastings J., Owen G., Dekker A., Ennis M., Kale N., Muthukrishnan V., Turner S., Swainston N., Mendes P., and Steinbeck C. ChEBI in 2016: Improved

services and an expanding collection of metabolites. *Nucleic Acids Research*, 44(D1):D1214–D1219, October 2015.

[72] Ying Z. and Kostas K. Towards a portable xml-based source code representation. *Proc. XML Technologies and Software Engineering (XSE2001)*, pages 343–353, 2001.

[73] Alaimo S., Pulvirenti A., Giugno R., and Ferro A. Drug–target interaction prediction through domain-tuned network-based inference. *Bioinformatics*, 29(16):2004–2008, May 2013.

[74] Roger I. *Mesos in Action*. Manning Publications Co., USA, 1st edition, 2016.

[75] Perwej Y., Kerim B., Adrees M. S., and Sheta O. E. An empirical exploration of the yarn in big data. *International Journal of Applied Information Systems*, 12(9):19–29, December 2017.

[76] D. Dominguez-Sal, P. Urbón-Bayes, A. Giménez-Vañó, S. Gómez-Villamor, N. Martínez-Bazán, and J. L. Larriba-Pey. Survey of graph database performance on the HPC scalable graph analysis benchmark. In *Web-Age Information Management*, pages 37–48. Springer Berlin Heidelberg, 2010.

[77] Nadime Francis, Andrés Taylor, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, and Petra Selmer. Cypher. In *Proceedings of the 2018 International Conference on Management of Data - SIGMOD '18*. ACM Press, 2018.

[78] Zhang G., Shi J., Zhu S., Lan Y., Xu L., Yuan H., Liao G., Liu X., Zhang Y., Xiao Y., and Li X. DiseaseEnhancer: a resource of human disease-associated enhancer catalog. *Nucleic Acids Research*, 46(D1):D78–D84, 10 2017.

[79] Barbarino J. M., Whirl-Carrillo M., Altman R. B., and Klein T. E. PharmGKB: A worldwide resource for pharmacogenomic information. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, 10(4):e1417, February 2018.

[80] Birney E. An overview of ensembl. *Genome Research*, 14(5):925–928, May 2004.

[81] Volders P., Anckaert J., Verheggen K., Nuytens J., Martens L., Mestdagh P., and Vandesompele J. LNCipedia 5: towards a reference set of human long non-coding RNAs. *Nucleic Acids Research*, 47(D1):D135–D139, October 2018.

[82] Jeggari A., Marks D. S., and Larsson E. miRcode: a map of putative microRNA target sites in the long non-coding transcriptome. *Bioinformatics*, 28(15):2062–2063, 05 2012.

[83] Kozomara A., Birgaoanu M., and Griffiths-Jones A. miRBase: from microRNA sequences to function. *Nucleic Acids Research*, 47(D1):D155–D162, 11 2018.

[84] Hsi-Yuan H., Yang-Chi-Dung L., Jing L., Kai-Yao H., Sirjana S., Hsiao-Chin H., Yun T., Yi-Gang C., Chen-Nan J., Yuan Y., Jia-Tong X., Yue-Ming L., Xiao-Xuan C., Zhen-Yu Z., Xiao-Hang C., Yuan-Yuan P., Liang H., Jin-Jiang S., Shi-Dong C., Fei W., Yue-Yang X., Si-Yuan D., Meng-Fan L., Chih-Hung C., Nai-Wen C., Kai-Wen C., Yu-Hsiang C., Xin-Hong W., Wen-Lian H., Tzong-Yi L., Feng-Xiang W., and Hsien-Da H. miRTarBase 2020: updates to the experimentally validated microRNA–target interaction database. *Nucleic Acids Research*, October 2019.

[85] Boya X., Qin D., Hongjin H., and Di W. miRCancer: a microRNA–cancer association database constructed by text mining on literature. *Bioinformatics*, 29(5):638–644, 01 2013.

[86] Wishart D. S., Li C., Marcu A., Badran H., Pon A., Budinski Z., Patron J., Lipton D., Cao X., Oler E., Li K., Paccoud M., Hong C., Guo A. C., Chan C., Wei W., and Ramirez-Gaona M. PathBank: a comprehensive pathway database for model organisms. *Nucleic Acids Research*, 48(D1):D470–D478, 10 2019.

[87] UniProt: the universal protein knowledgebase. *Nucleic Acids Research*, 45(D1):D158–D169, November 2016.

[88] Jeske L., Placzek S., Schomburg I., Chang A., and Schomburg D. BRENDA in 2019: a European ELIXIR core data resource. *Nucleic Acids Research*, 47(D1):D542–D549, 11 2018.

[89] Haft D. H., DiCuccio M., Badretdin A., Brover V., Chetvernin V., O'Neill K., Li W., Chitsaz F., Derbyshire M. K., Gonzales N. R., Gwadz M., Lu F., Marchler G. H., Song J. S., Thanki N., Yamashita R. A., Zheng C., Thibaud-Nissen F., Geer L. Y., Marchler-Bauer A., and Pruitt K. D. RefSeq: an update on prokaryotic genome annotation and curation. *Nucleic Acids Research*, 46(D1):D851–D860, 11 2017.

[90] Jordi Boggiano. Packagist the php package repository.

[91] Franz M., Lopes C. T., Huck G., Dong Y., Sumer O., and Bader G. D. Cytoscape.js: a graph theory library for visualisation and analysis. *Bioinformatics*, page btv557, September 2015.

[92] Anna Stavrianou, Periklis Andritsos, and Nicolas Nicoloyannis. Overview and semantic issues of text mining. *ACM SIGMOD Record*, 36(3):23–34, September 2007.