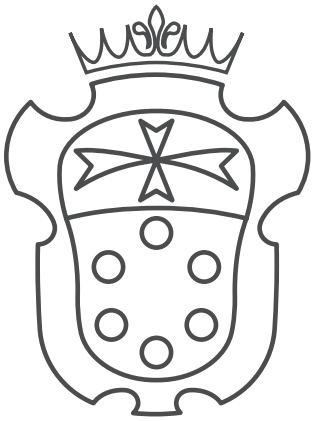


SCUOLA
NORMALE
SUPERIORE

Classe di Scienze
Data Science Ph.D.
XXXVI Cycle

Building a Biomedical Knowledge Graph from PubMed Central articles

Candidate
Lorenzo Bellomo

Supervisors
Prof. Paolo Ferragina
Prof. Dino Pedreschi

Academic Year 2022/2023

ABSTRACT

Natural Language Processing, Knowledge Graphs, and Artificial Intelligence have deeply impacted multiple disciplines, with biomedicine standing out due to its significant societal impact and overall research effort. Crucially, the vastness of the biomedical field, evident from the overwhelming publication rates on repositories like PubMed, presents challenges in manually accessing and synthesizing up-to-date research. Due to this growing issue, researchers are increasingly leveraging AI-driven computational tools for efficient navigation through various biomedical knowledge repositories.

In this thesis, we focus on the study, design, implementation, and evaluation of platforms aimed at constructing comprehensive and accurate biomedical Knowledge Graphs (KG). The initial effort culminated in the creation of BioTagME, a large biomedical KG containing nodes representing bio-entities, and edges representing the strength of their relationships, sourced from ontologies and PubMed abstracts. While BioTagME showcased promising results, it highlighted pitfalls and limitations related to the task of entity retrieval, and (lack of) labeling for their edges.

We then introduced OntoTagME, a tailor-made biomedical entity linker, whose goal was to surpass some of BioTagME's limitations. OntoTagME performs high-quality filtering of biomedical entities from Wikidata and then merges its annotations with another state-of-the-art tool: PubTator. This integration obtains a boost in F_1-metrics over those individual tools when evaluated on two ground-truth datasets of genes and diseases.

We finally deployed OntoTagME with state-of-the-art edge extraction and labeling techniques to design a novel, sophisticated, publicly-available platform offering on-the-fly biomedical network construction capabilities, named NetME. Notably, NetME enriches its KG with a Retrieval Augmented Generation (RAG) tool that allows users to generate succinct, contextually relevant, linguistically well-formed summaries about the entities modeled in that graph, thus extending the explainability capabilities of the knowledge distilled by our tool. Through several rigorous evaluations on gold standard datasets, NetME demonstrated superior performance against known biomedical KGs on the task of gene-disease association discovery.

We believe that this work represents a step forward in the design of AI-driven computational tools for the efficient and easy navigation of the biomedical literature.

PUBLICATIONS AND AWARDS

PUBLISHED WORK

Some of the results presented in this thesis have appeared in:

- Alessandro Muscolino, Antonio Di Maria, Rosaria Valentina Rapicavoli, Salvatore Alaimo, Lorenzo Bellomo, Fabrizio Billeci, Stefano Borzì, Paolo Ferragina, Alfredo Ferro, and Alfredo Pulvirenti. "NETME: on-the-fly knowledge network construction from biomedical literature." In: *Applied Network Science*, 7.1: 1–24, Springer, 2022. DOI: https://doi.org/10.1007/s41109-021-00435-x.
- Antonio Di Maria, Salvatore Alaimo, Lorenzo Bellomo, Fabrizio Billeci, Paolo Ferragina, Alfredo Ferro, and Alfredo Pulvirenti. "BioTAGME: A Comprehensive Platform for Biological Knowledge Network Analysis." In: *Frontiers in Genetics*, 13, 2022. DOI: ttps://doi.org/10.3389/fgene.2022.855739.

AWARDS

The NetME tool has been awarded within the *2022 Google Cloud Research Innovators program*[1]. The project, with PI Prof. Paolo Ferragina, was presented as a collaboration among researchers from the University of Pisa, Scuola Normale Superiore, and the University of Catania.

SUBMITTED

The following manuscript has been submitted for publication in the Bioinformatics journal.

- Antonio Di Maria, Lorenzo Bellomo, Fabrizio Billeci, Alfio Cardillo, Salvatore Alaimo, Paolo Ferragina, Alfredo Ferro, and Alfredo Pulvirenti. "NetMe 2.0: A web-based platform for extracting and modeling knowledge from biomedical literature as a labeled graph." Submitted to *Bioinformatics*.

1 https://cloud.google.com/edu/researchers/innovators?hl=en

ACKNOWLEDGMENTS

Ci tengo a ringraziare i miei supervisor per i preziosi consigli, l'attenzione, e la pazienza.

Voglio ringraziare di cuore anche tutta la mia famiglia, che immagino si aspetti una dedica divertente ma, come prevedibile, non la avrà. In particolare voglio salutare i nonni, che credo che ci tengano.

Un sentito ringraziamento va anche ai Pasava... scusate i Seppias... no ok i Sushi Socks stavolta sono sicuro (anche se ammetto che c'è molta intersezione con il gruppo ringraziato appena in precedenza).

Un saluto anche ai miei amici di Nulans, che sicuramente apriranno la tesi cercando questa pagina ed ignorando tutte le altre (e ancora, con il senno di poi, questo punto si applica anche a tutti gli altri gruppi precedentemente citati).

Ci tengo anche a ringraziare i miei amici di ISC, ed in particolare Devid che penso sia una delle massimo 5 persone che leggerà questa tesi nella sua interezza.

Ed infine voglio ringraziare Fede e Ghibli per tutti questi mille bellissimi anni insieme (e anche soprattuto per la pazienza, ce ne vuole).

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

GDA	Gene-Disease Association
PM	PubMed
PMC	PubMed Central
NER	Named Entity Recognition
GUI	Graphical User Interface
PoS	Part of Speech
KG	Knowledge Graph
SOTA	state-of-the-art
EL	Entity Linking
DAG	Directed Acyclic Graph
NLP	Natural Language Processing
NCBI	National Center for Biotechnology Information
KB	Knowledge Base
HDFS	Hadoop Distributed File System
DB	DataBase
NN	Neural Network
GUI	Graphical User Interface
API	Application Programming Interface
ML	Machine Learning
RAG	Retrieval Augmented Generation
LLM	Large Language Model
GPT	Generative Pre-Trained Transformer

1 INTRODUCTION

The importance of Natural Language Processing, Knowledge Graphs, and Artificial Intelligence is rapidly growing both in research and industry. The adoption of these techniques is widespread in both general-purpose applications (e.g., news recommender systems, and search engines) and specialized tools.

One particularly interesting field where the use of these techniques is blowing up is BioMedicine because, nowadays, obtaining up-to-date information on biomedical advancements and discoveries is extremely hard for those researchers. This is mainly due to the wide amount of work that is published daily on scientific repositories such as PubMed (PM), the most well-known abstract repository in all of Medicine, and PubMed Central (PMC), the largest open-access full-text repository in the biomedical field, both of which are hosted by National Center for Biotechnology Information (NCBI). To get an idea of their scale: over 1.5 million papers are published every year [1] on PM, amassing to a total of more than 36 million abstracts. This corresponds to a daily rate of roughly 4 000 new papers per day.

Given these figures, staying up-to-date with state-of-the-art scientific results is almost impossible if approached manually. This is the reason why more and more researchers are starting to use computational tools, mainly driven by AI, to obtain help in navigating and synthesizing this massive amount of digital information. For instance, Jim Weatherall, vice president of data science and AI at AstraZeneca, has stated [2] that getting AI to crawl through lots of biomedical data has helped his team find a few drug targets they would not otherwise have considered; and then he added: "Our biologists then go and look at that and see if it makes sense".

In fact, the pipeline typically followed by biomedical researchers to draw information from scientific repositories hinges on two steps: study the papers relevant to their subject of study, and then model the retrieved knowledge via biological interaction networks. These networks connect various bio-entities, with each connection annotated to specify the nature of the relationship between them. These *labeled* networks (aka, graphs) are one of the hot topics in biomedical research, and not only there, at the time of writing the thesis. According to a Gartner report [3], "by 2025, graph technologies will be used in 80% of data and analytics innovations, up from 10% in 2021, facilitating rapid decision making across the enterprise", further showing the interest both from the research field and the industry.

The most widespread and used graph model to date is the one called *Knowledge Graph (KG)*. It is a data structure that was first popularized by Google in 2012 when it announced the first version of a network connecting relevant entities for human knowledge (e.g., people, locations, events, objects, etc.). This network was, and currently is, used to generate the knowledge frame on the right of most search results, in Google and other search engines, which includes several "related links" and information details about query terms referring to famous entities (e.g., try the search "Galileo Galilei" on Google).

Since then, Knowledge Graphs have been applied to several other fields: one of them has been precisely the biomedical one. As a recent example, in the last February issue of the MIT technology review [2], it is stated that "many companies are applying machine learning to the problem of identifying targets. Exscientia and others use natural-language processing to mine data from vast archives of scientific reports going back decades, including hundreds of thousands of published gene sequences and millions of academic papers. The information extracted from these documents is encoded in knowledge graphs: a way to organize data that captures links including causal relationships such as A causes B".

This challenging setting has been the main topic of investigation of this thesis, which focused on the study, design, and implementation of a platform for building Knowledge Graphs that are complete and correct about a queried set of biomedical entities, and can be visualized and navigated efficiently and effectively by a researcher. Given the complexity of the problem, we explored several research directions that eventually led to the design and implementation of several tools and libraries, whose contribution and significance are sketched below, while their technical details are described in the respective chapters of the thesis.

We started with the design of BioTagME, which is a large biomedical KG that contains roughly 162K nodes of 9 different types (e.g., diseases, genes, proteins) and 41M edges. Edges are extracted from two kinds of sources: some are imported directly from known biomedical ontologies (e.g., DiseaseOntology [4], PathBank [5], DrugBank [6]), while others are mined through Natural Language Processing (NLP) techniques directly from a large set of PM abstracts using a general-purpose entity linker, TagME [7]. Bio-entities found with TagME are then linked together via edges, whose weight is higher for relationships that have greater biomedical evidence (both from abstracts and ontologies). We evaluated the quality of the KG generated by BioTagME through two case studies: the first one focused on the gene "BSG", and compared the neighborhoods of this node in BioTagME's KG with manually curated (by experts) relations. This obtained a sensitivity of 92% and a specificity of 95%. The second case study compared the neighborhoods of the node "blood coagulation pathway" against a manually curated network, achieving a sensitivity of 70% and a specificity of 96%. Particularly interesting was that BioTagME was able to find some crucial relations that are not present in any ontology, but were present in some PM abstracts. BioTagME was published in *Frontiers in Genetics* in 2022 [8], released as a full dataset in Zenodo [9], and subsequently made publicly available at https://biotagme.eu.

The experience maturated with the design of BioTagME has been an interesting preliminary step in constructing biomedical KGs, and indeed it highlighted some obvious pitfalls and limitations. Firstly, the use of a general-purpose entity linker as TagME, allowed easy and fast development of BioTagME that, however, obtained somewhat noisy biomedical annotations. Secondly, its edges were weighted but not labeled, thus missing the crucial "explanation" of their presence in the network.

To tackle the first issue, we then designed and implemented a tailor-made entity linker, called OntoTagME, whose design necessitated the development of proper biomedical Knowledge Base (KB), which occurred in two versions. The first version, called OntoTagME 1.0 [10], integrates a set of 15 ontologies suggested by an expert in the biomedical field (such as GeneOntology [11], PathwayOntology [12], DrugBank [6], DiseaseOntology [4]). The second version, which is a major update, called Onto-

TagME 2.0 [13], uses instead a general purpose KG (i.e., Wikidata) and hinges on the design of a novel and effective filtering technique to obtain a high-quality biomedical subset of it. In the end, this approach obtained a set of roughly 3.5M unique biomedical entities of 15 different types. To further improve the annotation performance, we integrated OntoTagME with another high-quality tool, namely PubTator [14, 15]. We tested the quality of this combination on two datasets and compared it against the results achieved by OntoTagME 2.0 alone, or PubTator alone. The combination was the best performer on the two tested datasets, achieving an F_1-measure of 0.49 on genes and 0.71 on diseases. We released a public REST Application Programming Interface (API) of that combination of OntoTagME 2.0 and PubTator at https://sobigdata.d4science.org/web/tagme/ontotagme-api.

The second limiting issue in the design of BioTagME was much harder to address, and indeed it required the design of a more sophisticated algorithmic pipeline, which led to the implementation of a novel platform called NetME. Briefly, NetME is a biomedical network construction tool that offers two main innovative and valuable features: it is the first to work on the fly and to extract knowledge from PMC full-texts.

We released version 1.0 of NetME in 2022, along with its description in the *Applied Network Science* journal [10]; then, we improved it in many aspects, thus releasing version 2.0, which is now publicly available at https://netme.click, and whose description has been recently submitted for publication to the BioInformatics journal [13]. Since NetME sits on top of PM and PMC, users can build bio-medical networks on the most relevant or most recent papers uploaded in these repositories, as well as they can build networks on their PDF files or texts. Crucially, NetME's edges are extracted and labeled using effective NLP techniques, and users can deploy NetME's Graphical User Interface (GUI) to inspect the network structure in detail by, for example, clicking on an edge and seeing the related entities, along with a label testifying the kind of relationship connecting them (e.g., "detects", "up-regulates").

We tested NetME 1.0 over three case studies. The first two focused on comparing the results of NetME against other biomedical networks on specific queries. In particular, Case Study 1 used known gene-gene interactions from the STRING [16] ontology as a gold standard, and NetME 1.0 reliably extracted relationships having a large score on STRING: namely, it achieved 82% sensitivity and 86% specificity on gene-gene interactions with a score on STRING larger than 900. Case Study 2 focused on comparing the results of NetME with a manually curated (by a biological expert) network for the gene "BSG". There, NetME obtained almost perfect results: i.e., 99% specificity and 100% sensitivity. Instead, Case Study 3 focused on the prediction of 100 random associations between gene BSG and diseases, extracted from DisGeNET [17]: here, NetME successfully retrieved 63 out of 100 associations.

Given these promising results and its precious full-text NLP capabilities, we decided to dig into the design of NetME 1.0 by addressing some of its limitations. This led to a significant redesign of NetME's algorithmic pipeline, with the addition of some new software modules, which culminated in the release of a new version of the tool. Precisely, we deployed OntoTagME 2.0 for entity linking, we improved the extraction of edges and their labels by deploying state-of-the-art (SOTA) NLP techniques for sentence parsing, and we added a "data science" panel to the GUI which can now run several algorithms for network analysis, like graph traversal and clustering. It is worth highlighting that, when extracting edges, we switched from a verb-

centric approach (which resulted in many redundancies and a high computation time) to a dependency-centric approach, where NetME performs only one visit of the dependency-parse tree, thus reducing running time and increasing the annotation accuracy. Finally, and probably more importantly, we designed and developed a Retrieval Augmented Generation (RAG) solution to biomedical text generation [18, 19] that combines the capabilities of GPT-4 [20] with the KG built on-the-fly by NetME, thus originating a novel *Graph RAG* approach to the retrieval of biomedical information. This way, users can now ask for a summarized snippet in natural language, which explains in a synthetic well-formed linguistic way the results modeled by the NetME's KG.

In order to evaluate the improvement introduced by NetME 2.0 over its first version we performed three case studies using DisGeNET as a gold standard. The first one focused on PM queries about three genes: APP, BRAF, and BRCA1. For each Gene-Disease Association (GDA) present in DisGeNET for these three genes, we took the list of PM abstracts referring to that GDA (as indicated by DisGeNET) and built a network with NetME 2.0. In total, out of 46 GDAs, 10 of them were missed by NetME: but, manually inspecting those 10 mistakes, we noticed that only 2 of them were a real fault because the others were not present in input abstracts from PM, and thus they could be not inferred by NetME. The second case study was focused on estimating the quality of NetME's answers to open queries. In this case, we generated the networks of two genes (i.e., ABAT and CACNA1A) and counted the amount edges retrieved out of the manually curated set available in DisGeNET. Here, NetME found 17 out of 25 total edges. We then manually checked and found that NetME was indeed able to find most of the missing edges (5 out of 8) with a more focused query. We also manually checked the 3 remaining mistakes and found that there was no biological evidence in PM of those associations, so there was no meaningful way for NetME to find them. The third case study focused on a direct comparison with NetME 1.0, by re-running Case Study 3 above on 100 random DisGeNET associations: here, NetME 2.0 obtained a net improvement of +38%.

As a final test, we compared NetME 2.0 with 5 other well-known biomedical KGs, as well as with NetME 1.0 and BioTagME, over the same set of 100 randomly sampled GDAs from DisGeNET (mentioned above), and checked how many of those were modeled as edges of those graphs. The experiment showed that NetME's engine was the best one by achieving a Recall of 87%, with an absolute improvement over those KGs ranging from 2% to 87%. But, it is worth mentioning that, the tool obtaining the closest performance to NetME (i.e., DARLING [21]), simply connects bio-entities based on their co-occurrence in PM abstracts, thus without any labeling; conversely, the closest tool offering similar features to NetME's KG (i.e., BioKG [22]) is worse in Recall by an absolute 39%.

1.1 THESIS OUTLINE

The thesis is organized as follows.

Chapter 2 contains a description of the state-of-the-art tools and research topics that are the core of the studies performed in this thesis. The topics covered range from data structures to biomedical ontologies, from entity linking tools to Natural Language Processing pipelines, up to Knowledge Graphs for Biology, and more.

Chapter 3 presents BioTagME [8, 9], its back-end and front-end pipelines, and then shows the two case studies we performed to test the efficacy of its generated bio-KG

Chapter 4 presents the two versions of OntoTagME. First, we describe the reasoning behind the choice of TagME as its building block, and then we describe the first version of the tool, showing the integrated ontologies. Next, we show the proposed algorithms to create a biomedical view of two well-known general-purpose KBs (i.e., Wikipedia and Wikidata). Finally, we show OntoTagME 2.0 and its evaluation on two well-known datasets for genes and diseases. We also provide a short discussion about a possible generalization of OntoTagME, namely TopicalTagME, whose goal is to have an easy way of building a custom version of TagME on any KB, which may be of independent interest.

Chapter 5 shows the two versions of NetME. The chapter starts with a deep dive on NetME's GUI, thus showing the full set of functionalities of NetME's web app. Then we describe NetME 1.0 [10], along with the three case studies we performed to test the effectiveness of this tool. Finally, we focus on its next major release, namely NetME 2.0 [13], where we describe the new edge extraction pipeline, along with all the 2.0 exclusive features, like the Graph RAG engine that integrates OpenAI ChatGPT-4. We conclude the Chapter with a discussion of four case studies aimed at comparing the performance of NetME 2.0, against its version 1.0 and several other biomedical KGs on the task of GDA discovery.

Chapter 6 concludes the thesis, proposing some possible future directions for this research topic.

The work presented in this thesis is joint work with researchers at the University of Catania (members of the FerroLab[1]).

1 https://ferrolab.dmi.unict.it/

2 BACKGROUND

In this chapter, we discuss the related work on the topics that are relevant to this thesis, such as data structures, biological databases, Natural Language Processing pipelines, graph algorithms, entity linkers, and many more.

2.1 GRAPHS AND THEIR ALGORITHMS

We introduce the most relevant data structure for our research and the set of algorithms we will use. A graph $G(V, E)$ is a data structure that is identified by two sets, the set V of nodes (or vertexes), and the set E of edges. This structure is extremely powerful for modeling datasets where the core of the information is in the relation between data. For example, when modeling airport routes, airports can be the nodes of a graph, and edges may be the direct routes between them.

Edges are identified by couples (s, t), where s is the source node and t is the target node. Additionally, edges can be:

- *Weighted*: An edge weight is a numerical value that corresponds to a sort of *degree of impact* of it. In the example of the airport routes graph, weights may be the distance in km in a beeline.
- *Labeled*: An edge label is a textual string that describes the "type" of the relation between the two entities. In the airport routes example, this might be "CROSS_COUNTRY", which is only present in edges that cross the borders of at least one nation.
- *Directed*: An edge can be directed if the relation itself has an inherent source and target node, and this relation is not semantically bi-directional. In the case of the airport routes graph, edges that represent distances are undirected, while edges that represent routes are directed.

2.1.1 *Knowledge Graphs*

A KG is a data structure that has become increasingly popular in the last years [23]. Its first technological breakthrough came in 2012 when the Google KG [24] was announced. It is a specific instance of the graph data structure, and it is used to encapsulate and model knowledge and information.

It differs from regular databases thanks to its heavy focus on relationships between entities. More specifically, KGs are, in most cases, a specific instance of a graph that is directed and labeled.

As an example, let's take Google's proprietary KG. As of March 2023, the Google KG contained roughly 8 billion entities and 800 billion facts (edges). This is the data structure that Google mines (at least) to retrieve information to show in the search info boxes, along with related entities.

The power of the KG comes in the way information is accessed. On Google, if the user is searching for a famous person, it makes sense that Google provides their data, along with people in close contact with them on top of the search results. This is possible because of the edge labels. Close people have a high probability of sharing many edges on the KG, and this allows to make inference based on relationships, that were unthinkable on regular databases of that scale.

2.1.2 *PageRank*

The problem of estimating node importance in a graph $G(V, E)$ is historically one of the most interesting problems regarding said data structure. It is safe to assume that, out of the many proposed solutions, the most widespread one is the *PageRank* [25], which is allegedly used by Google for web page ranking.

PageRank is an algorithm based on random walks, and it outputs a probability distribution over graph nodes, which may be interpreted as the importance of each node in the graph.

The key idea underlining the PageRank approach is to slightly alter the Web Graph to guarantee the convergence of the Random Walk to a stationary distribution. This is enforced with the use of a teleport step which, when picked with a fixed probability at each step, forces the random walker to travel to a node in the graph with uniform probability distribution.

The PageRank formulation is the following:

$$r = [\alpha \Pi^T + (1 - \alpha) e e^T] \times r$$

where:

- $\Pi(i, j) = \begin{cases} 1/\#out(i) & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases}$
- r is the principal eigenvector ($|r| = |V|$) and is the result vector whenever the random walk reaches the steady state. This convergence is guaranteed by the properties of the graph (i.e., aperiodicity and irreducibility) enforced by the presence of a teleportation step.
- $e = [1/|V|, 1/|V| \dots 1/|V|]$, where $|e| = |V|$ is the uniform teleportation vector.
- α is the dumping factor ($\alpha = 0.85$ in the original paper) and expresses the probability that the random walker keeps traveling in the graph. The value $1 - \alpha$ is instead the probability of teleporting with uniform distribution to any node in the graph.
- $\#out(x)$ is the number of edges outgoing from node x.

The original PageRank algorithm has, as its main focus, the one of estimating the node importance in the whole graph. Other times, the relevance of a node with respect to another one, or a set of nodes, may be more useful for some kinds of applications. In this case, Personalized PageRank proves to be more useful (still provided in the original PageRank paper [25]). The only thing varying compared to standard

PageRank is the teleportation vector, which changes from being a uniform distribution to a specific probability distribution centered in the nodes one wishes to evaluate the relevance on.

2.1.3 *DT-Hybrid*

Domain-Tuned Hybrid (DT-Hybrid) [26] is an algorithm that provides a framework for in-silico prediction of drug and target relationships. DT-Hybrid extends the NBI algorithm proposed in [27] by adding application domain knowledge.

DT-Hybrid employs the recommendation method based on the bipartite network projection technique and the concept of resource transfer within the network. Given a set of entities (e.g., biological compounds, molecules) $X = x_1, x_2, \ldots, x_m$, and a set of categories (e.g., genes, proteins) $T = t_1, t_2, \ldots, t_n$; 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 $t_j \in T$ is in the graph if and only if the entity x_i is associated with the category t_j.

The network can be represented with an adjacency matrix A, where $a_{i,j} = 1$ if x_i is connected to t_j, and it is $\emptyset$ otherwise. Once the bipartite graph is built, a two-phase resource transfer begins, where 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 bipartite network as follows:

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

Where:

- $k(x_l)$ is the degree of the node x_l 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 t_i and t_j, respectively, and α is a tuning parameter.

Given the weight matrix W and the adjacency matrix A of the bipartite network, the recommendation matrix R is computed as 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 entities s_i having a higher interaction with the category t_j are in the first rows of the matrix column i. DT-Hybrid maintains only the top l column values, where l is a user-defined parameter.

2.1.4 *Graph Databases and Neo4j*

In order to represent graphs, using a regular SQL DataBase (DB) is not the best fit. This is due to the fact that databases traditionally have a strong focus on the design of tables. Relations are traditionally obtained, in SQL databases, through table joins. Of course, JOIN operations are traditionally well-optimized, however, relations are not the main focus by design.

For this reason, a large amount of graph DBs have been designed [28] and released to the public. In this thesis, to represent our large KGs, we will use Neo4j [29]. Neo4j

is a Java-based graph database. At the time of writing this thesis, Neo4j is the top-ranked graph database according to the DB-Engines Ranking [28]. This ranking estimates the popularity of DB-engines and updates those ranks monthly.

Neo4j uses a proprietary query language, called *Cypher* [30], in order to offer the user an SQL-like syntax that is best fit for graph data structures. The core design principle of Cypher is the one of instant-readability, making queries, in most cases, extremely easy to decipher from a user's standpoint.

The code shown in the listing 1 below describes a basic example of the syntax of this language.

As we can see, nodes are represented with round brackets. Any relevant information, used at query time, is also written inside the round brackets. Relationships are drawn by connecting nodes with dashes/arrows. All the relevant information is written succinctly inside the syntax.

The user is also able to assign variable names to a set of nodes or edges matching with specific properties, for future queries and updates.

The query language is powerful because nodes and edges are treated with the same level of priority. This implies that common operations on graphs are extremely optimized on Neo4j. Those operations include shortest path, PageRank, neighborhood computation, property retrieval, distance computation, and node degree retrieval.

2.2 NLP ALGORITHMS AND MODELS

Natural Language Processing is a core part of the work presented in this thesis. For this reason, many algorithms, techniques, and models are presented and explained in this section. Section 2.2.1 describes the classic approach to NLP, called bag-of-words, because it treats words like items in a bag. Section 2.2.2 shows Part of Speech (PoS) tagging, one of the most important techniques for phrase parsing and dependency extraction among its constituting terms. Section 2.2.3 describes a classic Machine Learning (ML) technique, Word2Vec [31], and one interesting application of this model to the Wikipedia KG. Finally, Section 2.2.4 shows a glimpse of the recent research and opportunities provided by Generative Pre-Trained Transformer (GPT) models.

Listing 1: Cypher syntax for nodes and edges

```
// node
(variable:Label {propertyKey: 'propertyValue'})

// relationship
-[variable:RELATIONSHIP_TYPE]->

// Cypher pattern
(node1:LabelA)-[rel1:RELATIONSHIP_TYPE]->(node2:LabelB)
```

2.2.1 *Bag-of-Words*

The first approach we are going to dive into is the standard NLP approach that handles text as a *bag of words*. Any text mining operation performed on document content is done by exploiting purely syntactic features. The most famous metric for bag-of-word mining is the TF IDF [32]. This metric leverages two factors. One, the term frequency (TF), takes into account the presence of a term in a text, and the other, the inverse document frequency (IDF), down-scales the importance of common words. The TF IDF metric of a term t in a document D is defined as:

$$\text{TF IDF}(t, D) = \text{TF}(t, D) \cdot \log \frac{n}{n_t}$$

where $\text{TF}(t, D)$ is the number of times in which term t appears in document D, n is the total number of documents in the collection, and n_t is the number of documents that contain the term t.

This metric was proposed because of the observation that, if a very uncommon word is shared between two documents, then the likelyhood of those two documents being similar is much higher than if they shared a very common word. For example, two documents sharing the word *"the"* does not necessarily imply that they are similar, while them sharing *"Netflix"* is a much stronger signal.

This metric depends on the term t, so it is perfect for word-document similarity. In the case of document-document similarity, the solution is to model them as vectors. The vector length n equals the total size of the collection word dictionary. Entry i of the vector is the TF IDF score of term i in the given document. Given that, and by modeling each document as a vector in the same n dimensional space, we can easily compute their relatedness using well-known metrics such as *cosine similarity*. The cosine similarity between two document vectors is defined as:

$$\cos(\vec{q}, \vec{p}) = \frac{\vec{q} \cdot \vec{p}}{\|\vec{q}\| \, \|\vec{p}\|} = \frac{\sum_{i=1}^{n} q_i p_i}{\sqrt{\sum_{i=1}^{n} q_i^2} \sqrt{\sum_{i=1}^{n} p_i^2}}$$

2.2.2 *PoS Tagging and Dependency Parsing*

PoS tagging is the process of identifying and marking words in a text with the tag identifying their type. It is an extremely useful technique in many fields, including linguistics and NLP. The set of all possible tags is not fixed, and it can vary from library to library, and from language to language. Each application has to provide its ad-hoc tagset. The most commonly used tagset for English plaintext is the Penn Treebank PoS tagset[1].

For a simple example on a PoS tagging process, say we run this algorithm on the phrase *"I like to read books"*. In this case "I" would be labeled as a pronome (PRO), "like" as a non-3rd person singular present verb (VBP), "to" as its own tag (TO), "read" as a verb (VB), and "books" as a singular noun (NNS).

The two main types of PoS taggers are *rule-based* and *stochastic*.

1 https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

Rule-based PoS taggers are the oldest algorithms that were developed for this task. This requires a large set of manually assigned rules that specify how to assign tags given the context surrounding each word. A rule-based tagger first gets all the possible PoS tags for the individual words. Then it parses all its internal rules and chooses the correct tag. For example, "play" can be either a verb or a noun, but there are rules that allow to disambiguate those cases (e.g., if the word "play" comes with an article before, then it is a noun).

Stochastic taggers are supervised models, and they exploit frequencies or probabilities of the tags in the training dataset to assign a tag to a new word. Two types of stochastic taggers exist. The first one relies on word frequency: here, the tagger assigns to a word the tag that is most frequent in the training set for that word. As expected, this method is very simple, but any unorthodox tag is missed, as this fails to keep track of any type of context.

A better version of this type of tagger relies on tag sequence frequency. In this case, the best tag for a word is determined using the probability of the tags of all the previous words in a defined window. This is the reason why this method is also called the *N-gram approach*. As expected, this method works better in practice, as it employs the best of the rule-based method (sliding window to check tags) and the statistic robustness provided by stochastic taggers.

For the purpose of this thesis, we need both PoS tagging and a strong *phrase dependency* framework. This is because we will often need to relate two nouns with their verb, and the relative dependency will be crucial. For this purpose, we use the spaCy [33] dependency parser[2].

According to the documentation, the spaCy parser uses a variant of the non-monotonic arc-eager transition system described by Honnibal and Johnson [34], with the addition of a "break" transition to perform the sentence segmentation. A transition-based dependency parser is a parser that scans the entire input string and is able to build a tree in $O(n^2)$ time in the worst case, where n is the number of words in the phrase. The parser employs a queue, a stack, and a set of transitions. Words are popped (in order) from the phrase and put into the stack. Each time a word is put in the stack, the system decides:

- *SHIFT*: push the next word in the buffer onto the stack.
- *LEFT-ARC*: add an arc from the topmost word on the stack to the second-topmost word. Pop the second-topmost word.
- *RIGHT-ARC*: add an arc from the second-topmost word on the stack to the topmost word. Pop the topmost word.
- *NON-MONOTONIC TRANSITION*: Optional set of operations that induce non-monotonicity. Without any, the tree-building algorithm is greedy and takes $O(n)$ time.

Honnibal and Johnson [34] added two operations as a means to achieve non-monotonic transitions, and they are:

- *REDUCE*: deletes the top element from the stack.

2 https://spacy.io/api/dependencyparser

- *UNSHIFT*: removes the top element from the stack, adding it as a first element to the buffer. A bit-vector is used to prevent SHIFT/UNSHIFT cycles.

ML can be plugged by training the network on a set of gold-standard dependency trees and then learning the best decision for each step out of the set of modeled ones.

The end goal of this process is to not only retrieve the set of PoS tags but also to construct a dependency tree between those tags, which will allow us to obtain relevant triples formed by two biological nouns and a verb connecting them in our biological context. Figure 1 shows an example of PoS tagging and dependency parsing performed by spaCy on the phrase "I like to read books".

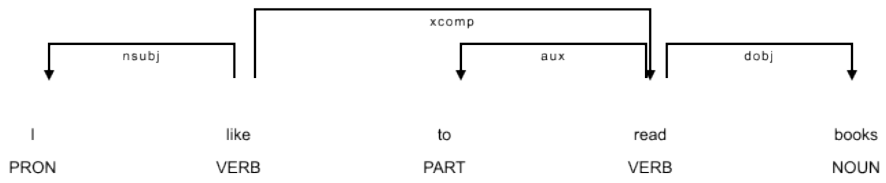


Figure 1: PoS tagging and dependency parsing example

2.2.3 *Vector Embeddings*

Vector embeddings are an extremely powerful ML technique to embed tokens (they may be words, entities, documents, or n-grams...) into a latent numerical vector space. The most common application is to embed single words. The assumption made is that embedding vectors of words having a similar meaning are expected to be close in the vector space. These vectors are extracted through unsupervised ML.

The most well-known and widespread application of this concept is the one of *Word2Vec*[31]. Here, vectors are trained using a two-layer Neural Network (NN) with the task of identifying the context of words in the English language.

The NN's architecture is based on either of the following models:

- The *Continuous Bag of Words* model (CBOW) predicts the current word based on a set of context words. The fact that context words are modeled as a set means that their ordering is not relevant. This model training is faster but performs worse for rare words.
- The *Continuous Skip-Gram* model uses the current word to predict the neighbor ones. Weights are increased for closer words with respect to farther ones. This model has a higher training time but performs better for rare words.

Word2Vec accepts a set of parameters to use during training, which include:

- *Training algorithm*: hierarchical softmax and/or negative sampling. The former is better for infrequent words, while the latter performs best for frequent ones. On low-dimensional vectors, negative sampling is better.

- *Sub-sampling*: removes words that are too frequent or very rare to improve performance on the most crucial words. Thresholds are set by the user.
- *Dimensionality*: Better performance and longer training times are expected by increasing the dimensionality of the vector. A good dimensionality value, according to the authors, ranges from 100 to 1 000. Increasing this value results in diminishing returns.
- *Context window*: Increasing this window increases the number of contiguous words checked for a given word. The recommended value is between 5 and 10.

There are many proposed extensions of Word2Vec, but we focus on the most relevant ones for the applications considered in the thesis, which are Wikipedia2Vec [35] and BioVectors [36].

Wikipedia2Vec [35] is a Python library that adapts the embedding approach to Wikipedia pages instead of single words. The library provides pre-trained models for various languages and some useful functionalities, like the ability to retrieve the *closest* (meaning most similar) Wikipedia pages with respect to the one provided as input.

The learning process depends on the graph's topology, and as such it retains latent information regarding its structure. The actual learning is performed by exploiting neighbor words as contexts (skip-gram model), but additionally, words that are close to Wikipedia anchors are used to keep track of the context when "jumping" to another Wikipedia page. Another thing that is taken into account to build the final embedding vector is the set of neighboring entities.

BioVec [36] is an extension of embedding vectors for n-grams in biological sequences (e.g., DNA, RNA, proteins). This representation can be widely used in applications of ML to several biomedical fields, such as proteomics and genomics. The achieved results suggest that BioVectors can characterize biological sequences in terms of biochemical and biophysical interpretations of the underlying patterns, that are implicitly learned through the latent features hidden in the training texts.

2.2.4 *GPT models*

GPT models are at the forefront of NLP research at the time of writing this thesis. We call "GPT models" every NN approach based on the transformer architecture, pre-trained on large data sets of unlabelled texts, and capable of generating text that closely resembles the one generated by humans.

The current rendition of this model is highly influenced by Google's BERT [37], another Large Language Model (LLM), which is based on the same principles, but is not generative. In the same year BERT was published (2019), the OpenAI consortium released the first generative approach to transformer models, releasing *GPT-1* [38].

The main improvement over BERT-like models is in the ability to generate human-like text, without sacrificing the ability to obtain excellent performance on other non-generative NLP tasks.

The main design principles adopted by GPT-1 were:

- Unsupervised massive pre-learning phase. This is obtained using a large NLP dataset where phrases are well-constructed and long. For this reason, the dataset of choice is BookCorpus [39].
- Smaller supervised fine-tuning phase after the unsupervised one. One of the tasks tackled in this phase is *Natural Language Inference*, where, given two phrases, the system has to decide whether the relation between those phrases is entailment, contradiction, or neutral.

OpenAI then started developing the GPT-n models. Each major update obtained a new number $n = 2, 3, 4$. At the time of writing this thesis, the latest model is OpenAI's GPT-4 [20]. Details on the training process, number of parameters, and costs of this architecture are not public. The main new features of GPT-4 are the ability to recognize images and their content, and the possibility of querying the internet for up-to-date information.

GPT models are not perfect by any means. Their most notorious issues are:

- *Hallucination*: it is the most prominent of issues for Generative-API text applications. This is the process of generating false or misleading information, but presenting it as a fact. According to The New York Times, ChatGPT hallucinates up to 27% of answers [40]. It can be mitigated through human supervision, but the process is extremely costly both in time and human resources required. This issue is inherent in the chatGPT architecture.
- *Lack of Explaination*: as for many black-box models in AI, they tend to provide answers, but fail to provide an explanation both as to why that answer should be satisfactory and to the reasoning employed to generate such an answer in the first place.
- *Lack of Transparency*: The precise design of chatGPT and its training are not public since GPT-3.
- *Harmful Uses*: OpenAI fine-tuned chatGPT through reinforcement learning with human annotations in order to ensure that the model refuses to answer any question that may be harmful, violent, or sexual content. This system, however, has been shown to be imperfect and easy to dodge.
- *Biases*: Microsoft researchers suggested that the most recent GPT model may exhibit biases [41].

2.2.5 *Retrieval Augmented Generation*

Products built on top of LLMs such as OpenAI's ChatGPT are surely brilliant and getting more and more sophisticated, with a growing number of applications in many domains. They generate human-like text by predicting the likelihood of a term given the preceding ones, using what is known as transformer-based architectures. Powerful as they are, current LLMs suffer from several drawbacks. Some were mentioned above, while we list the other most notable ones:

- they are *static*, LLMs are "frozen in time" because it is not feasible to keep frequently updated their gigantic training datasets

- they lack *domain-specific knowledge*, being them trained just for generalized tasks (see e.g., ChatGPT [20] or LLama [42])
- they generate responses based on *patterns learned* during training, without the ability to actively retrieve specific information.

RAG [18, 19] is a technique introduced recently to enhance the capabilities of LLMs. It involves combining generative pre-trained models with information retrieval systems. The key idea is to fetch up-to-date or context-specific data from an external database and make them available to a generalized LLM when asking it to answer a user query.

The key advantage of this approach is that being the external DB of smaller size, it can be kept up-to-date easier and faster, and it can be accessed by the LLM at generation time, thus reducing the likelihood of hallucinations. The result is a noticeable boost in the performance and accuracy of Generative AI applications, which can thus return more context-aware, precise, and informed responses.

The storage of external DBs nowadays hinges on so-called *vector DBs* (see e.g., Pinecone[3], and Weaviate[4]), namely indexed archives of multi-dimensional vectors that represent suitable embeddings of sentences drawn from the domain of interest and supporting nearest-neighbor (aka semantic) searches. The adjective "suitable" refers to the fact that those vectors, of thousands of components each, should be located in a multi-dimensional space so as to reflect the semantic similarity of sentences: the closer the vectors, the more similar the corresponding sentences.

A common RAG pipeline is as follows. The user query is parsed into sentences, and their embeddings are sent to the vector database, which performs a "nearest neighbor" search, finding the vectors (sentences) that most closely resemble the user's intent. The relevant sentences returned by the vector DB are then used to compose the *prompt* of the LLM (via its context window) and to perform its generative task.

However, for complex or polysemous queries, RAG may retrieve ambiguous or uncertain sentences, which affect the quality of the generated content. So researchers proposed to combine RAG with KGs (aka, Graph RAG [43]) that model entities and their relationship in some specific domains, and thus can be used to better comprehend the intent of complex queries, leading to more accurate and relevant search results. This advanced approach is the one we explore in Section 5.4, where we introduce a new graph RAG based on our biomedical KG.

2.3 ENTITIES AND ENTITY LINKERS

Entity Linkers are tools whose goal is to recognize and disambiguate entities (such as famous individuals, locations, events, organizations, or objects) in text. Entities are modeled as nodes/pages in KBs. Common choices for KBs are Wikipedia and Wikidata, which is the reason why this process is also called *Wikification*. For example, given the phrase "Italy won the 2023 Davis Cup", the entity linker should recognize that "Italy" refers to the Italian Tennis Team, while "2023 Davis Cup" refers to the tennis team competition. In this example, we call "Italy" the *mention* (part of the text mentioning an entity), while the linked page in the KB describing the "Italian Tennis Team" is the corresponding *entity*.

3 https://www.pinecone.io/
4 https://weaviate.io/

There are two types of Entity Linkers, disambiguation-only tools and end-to-end tools. The former takes as input the set of mentions, and their task is only to disambiguate those mentions with entities in the KB. End-to-end models are instead the ones doing the full pipeline: they discover the mentions and their candidate entities, and then disambiguate them. For this thesis, we will only deal with end-to-end entity linkers, and refer to them as *entity linkers*. The SOTA for entity linkers is shown in the web page from NLP progress on said task [44] and reported in Table 1 with the values at the time of writing this thesis. The values shown were extracted from the Gerbil Entity Linking (EL) platform [45].

Name	Citation	Year	Micro-F_1-Strong	Macro-F_1-Strong
REL	[46]	2020	**83.3**	81.3
E2E NEL	[47]	2018	82.6	**82.4**
CHOLAN	[48]	2021	83.1	-
WAT	[49]	2014	70.3	73.0
DBPedia Spotlight	[50]	2011	71.9	72.8

Table 1: NLP Progress: End-to-end Entity Linking state-of-the-art

For end-to-end models, the SOTA for the micro-F1-strong measure is held by "REL: An Entity Linker Standing on the Shoulders of Giants" by Hulst et al. [46], while the macro-F1-strong one is held by "End-to-End Neural Entity Linking" by Kolitsas, Ganea, and Hofmann [47]. The F_1 measure is a measure of the harmonic mean of precision and recall, while the difference between micro and macro measures is in the way it handles class imbalance for multi-class classification problems. Micro-F_1 considers class imbalance when averaging, while macro-F_1 considers all classes as equally important.

More information on those two systems is provided in Section 2.3.2.

2.3.1 *TagME*

TagME [7] is an entity linker specialized in the fast annotation of short texts, using Wikipedia as the entity KB. The first step applied by TagME is mention detection, by querying the dictionary of anchors built from Wikipedia[5] and then keeping the longest non-overlapping sequence of terms. The second step is the one of entity disambiguation, and it is performed with a *collective agreement* between the entity candidates corresponding to the mentions in the neighborhood of the one to be disambiguated. The *agreement score* is expressed via the *Milne&Witten* [51] relatedness measure, and at the end of the process only the highest-ranked entities are kept.

TagME also allows the use of a *fast* mode, which restricts the voting process only to a window of consecutive mentions/entities, reducing both the time cost of the process and the impact of topic drift. The output is a list of annotations, where each element is:

5 We define *anchors* those sequences of terms in a Wikipedia page that at least once appear as clickable links (whenever the text appears in blue and redirects to another Wikipedia page).

- *Wikipedia page identifier*: The Wikipedia ID of the page corresponding to the annotated entity[6].
- *Wikipedia Title*: Title of the Wikipedia page.
- ρ *Score*: float number in $[0, 1]$ that expresses the semantic coherency of a given entity with respect to the surrounding text. The higher it is, the more pertinent the entity is to the surrounding text. This is the main value to check to distinguish good annotations from noise.
- *Start and End*: two integer values that locate the position of the mention that is marked as a topic. This part of the text is called *spot* or *mention*.
- *link probability*: This value is also in $[0, 1]$, and it is computed as the number of times that the anchor text a (so the mention in the text which was highlighted by TagME) is used as an anchor divided by the occurrences of a in Wikipedia (as an anchor or not). The higher the value, the higher the chances that this anchor actually expresses some real-world entity. Low link probability values are usually found in adjectives, articles, and verbs, while subjects are usually on the higher end of the spectrum.

ANNOTATION PROCESS. The disambiguation of the candidate entities is done by TagME with a voting scheme, involving those entities. The pipeline is the following:

- *Mention selection*: The first step is spent by TagME parsing the input text, and identifying, out of all the sequences of terms (mentions), which ones are also *anchors*, thus extracting the set A_T.
- *Entity disambiguation*: Out of all the anchors in $a \in A_T$, first TagME finds the set of Wikipedia pages that could be pointed by a, and then applies a collective agreement vote (more details later) that returns a set of entities (one per mention), associated with their *link probability* (defined above) and their ρ score.
- *Mention pruning*: Low ρ score annotations are discarded.

The *collective agreement* voting process starts from the set of anchors A_T. For each anchor $a \in A_T$, TagME selects p_a as the set of pages for which a is an anchor. For each one of those pages, a vote is cast by all the other anchors in the text (or by a close subset in case TagME is applied in *fast mode*). More precisely, for each anchor $b \neq a$, TagME uses as a vote for p_a given b the average relatedness between all candidate pages p_b (linked to anchor b) and page p_a. Relatedness between pages is computed using *Jaccard* similarity (i.e., $J(A, B) = \frac{|A \bigcap B|}{|A \bigcup B|}$) over shared input pages, as proposed by [52].

2.3.2 *REL and E2E Neural Linking*

In this section, we describe the two SOTA systems for EL, called REL and E2E.

6 To visit a Wikipedia page from an ID, use this URL: https://en.wikipedia.org/?curid=18630637

REL [46] is a NN-based entity linker, coming with a public repository[7] and queriable REST API[8].

Its pipeline consists of three phases:

1. *Mention Detection*: in this phase, REL employs Flair [53] for the step of Named Entity Recognition (NER), where given an input text, Flair extracts the mention exploiting word embeddings.

2. *Candidate Selection*: here REL selects the top 4 candidates for each mention with respect to entity-mention probability $p(e, m)$. This probability is estimated by computing the total hyperlinks in Wikipedia, CrossWiki, and YAGO. An additional set of 3 candidate entities is then added by maximizing a similarity score with respect to the context of the mention. This context is encapsulated using the word embeddings of all the words in a window of 50 words surrounding each mention, computed using the embeddings of Wikipedia2Vec [35], which was explained in detail in Section 2.2.3.

3. *Entity Disambiguation*: in this phase, REL maximizes an ad-hoc coherence function that incorporates contextual similarity and global coherence, where this global coherence incorporates relations between mentions in a single document. This computation is done between embedding vectors, computed in a latent space. The final score is provided by a two-layer NN.

End-to-end neural entity linker [47], similarly to REL, also relies on ML techniques to perform the mention-detection step. The main design idea of E2E-neural-EL is the one of exploiting the mutual dependency between the first EL phase of *Mention Detection* and the second one of *Entity Disambiguation*.

To achieve this, the authors take into account all possible spans in the input text as potential mentions and employ NNs to learn textual similarity scores over the entity candidates.

For this purpose, the authors use:

- *context-aware mention embeddings*: the authors employ regular Word2Vec [31], they encode the context using a bi-LSTM layer on top of the word embeddings and then concatenate the two parts (contextual vector and word vector).

- *entity embeddings*: the authors employ the approach of Ganea and Hofmann [54] to train embeddings. These embeddings are trained by approximating the word-entity distribution probability.

- *mapping between mention and entity*: this map is used to disambiguate entities in the end. A NN is trained using the contextual embeddings in the previous step in order to learn optimal entity-mention choices, which ensures that the disambiguation step takes into account both prior probabilities and the context. This map is also built by Ganea and Hofmann [54].

7 https://github.com/informagi/REL
8 https://rel.readthedocs.io/en/latest/

2.3.3 *SWAT*

SWAT [55] (Salient Wikipedia Annotator of text) is another topic annotator based on the Wikipedia KB which was used during the project. It offers a different approach with respect to the TagME annotation process. *SWAT* is more efficient on long and well-formatted text, and outputs a set of topics, together with their salience score. The higher the score is for an entity, the higher the importance of that topic in the queried text (TagME instead outputs the semantic coherence). Candidate entities are chosen using another tool: *WAT* [49]. *WAT* generates annotations by assuming that input text is less noisy and more refined with respect to TagME and extracts more information based on the syntactic structure of the phrases.

Both *WAT* and *SWAT* outperform TagME when dealing with structured text, but they are only available in English.

The output fields of a *SWAT* query are very similar with respect to TagME ones and are a list of annotations as described in the following:

- *wiki_id*: Wikipedia unique identifier.
- *wiki_title*: Wikipedia page title.
- *spans*: List of indexes (*start* and *end*) which report where in the text the given entity appears.
- *salience_score*: Score assigned by *SWAT* to this annotation. It lies in the range $[0, 1]$.
- *salience_class*: Boolean value that is 1 when the entity is considered salient by *SWAT*, and $\emptyset$ otherwise.

We are going to present some key algorithmic ideas underlying the way SWAT detects and ranks salient entities in the text.

SWAT applies three main steps:

1. *Document Enrichment*: the text is expanded with syntactic, semantic, and latent features.
2. *Feature Generation*: previously retrieved features are used to map entities to real-valued vectors.
3. *Entity Salience Classification*: entities are efficiently and efficaciously classified between salient or not salient, deploying their vectors.

The *Document Enrichment* step deploys various SOTA techniques and modules to capture as much information as possible from the input text, namely:

- *CoreNLP* [56] is deployed for PoS tagging (see Section 2.2.2) and creating the co-reference chain between words (which word, be it noun, pronoun, or anything else, refers to which other).
- *TextRank* [57] is deployed to score words in the input text, via a random walk, and produces a summary of it.

- *WAT* [49] is a topic annotator, the successor of TagME, aimed towards structured text, which maps mentions to entities (in this case, Wikipedia pages). The disambiguation phase is performed using two metrics: *commonness*, which represents the probability that a specific mention is disambiguated with entity e, and *coherence*, which represents the semantic coherence between the extraction and the textual context (similarly to *TagMe*'s ρ score). The weighting process is done by building a graph and scoring entities with the Jaccard relatedness measure. *WAT* effectively deprecates *TagMe* on well structured text.

- *Word2Vec* [31] is deployed to enrich documents with latent information. It is applied to all the entities extracted by *WAT* on the input text. Effectively, two kinds of latent embedding representation are used in this step: *Entity2Vec* and *DeepWalk* [58].

The second step is the one of *Feature Generation*. Features are built from the output of the four modules described above. The amount of features generated is large, but we are going to present the main ones below:

- *Position-based features*: capture the distribution within the input text of the entities' occurrences in order to predict their salience score.

- *Summarization-based features*: keep track of those entities that appear also in the summarized text, which is a strong indicator of their importance.

- *Linguistic-based features*: exploit dependency trees of the sentences where the entities occur to keep track of *CoreNLP* dependency relations between tokens.

- *Annotation-based features*: exploit the output from *WAT* (mainly commonness and coherence) in order to increase robustness in the topic annotation process.

- *Word2Vec-based features*: aim at keeping latent information about words, entities, and phrases that would otherwise be lost with standard syntactic features.

- *Relatedness-based features*: try to capture how related entities are to all the other ones in the input text. Relatedness is both computed with standard measures like *Milne&Witten* [51] and *Jaccard* similarity, or with cosine similarity between entity embedding vectors. The concept of entity relatedness is expanded upon in Section 2.3.4.

The last step is *Entity Salience Classification*, where entities are classified as *salient* or not by deploying gradient tree boosting on the features defined above.

2.3.4 *Entity Relatedness*

A measure that is often important in KG applications is the one of entity relatedness. This can either be used in cases where the set of nodes is known, but they have to be connected together, or to perform specific queries based on the *"closeness"* between the queried entity and the ones available in the KG.

In the literature, several measures have been proposed, both based on corpus text and on graph structure.

The first approach starts from a collection of documents (i.e., the corpus), and builds a statistical model to capture co-occurrences of terms in the corpus. The most widespread examples of these models are:

- *Vector Space Model*: retrieves the relatedness between two entities by computing the cosine similarity of their TF IDF vectors (more in Section 2.2.1).
- *Explicit Semantic Analysis* [59]: similar to the previous metric, but it is explicitly adapted to the content of Wikipedia pages.
- *Language Models* [60]: computes the probability of possible word sequences by aggregating counts from the corpus. The relatedness is estimated through a custom function of these counts.
- *Latent Dirichlet Allocation* [61]: a probabilistic model that fits each document to a set of k latent topics, where those topics are learned from the input corpus. Entity relatedness is estimated by computing the cosine similarity between the vectors of the respective topics.
- *Wikipedia2Vec* [35]: computes the embedding vector of each entity, and applies the cosine similarity directly to those.

The other type of measure is based on the graph structure of the source KG. Most commonly, this graph is Wikipedia. The most notable relatedness measures are:

- *Neighborhood Metrics*: A lot of standard metrics have been developed exploiting graph neighborhood, following the intuition that similar nodes will have a similar neighborhood set. The most well-known and adopted one is the one by Witten and Milne [51]. Other effective metrics are shown in the paper by Ponza, Ferragina, and Chakrabarti [62].
- *Personalized PageRank* [63]: to compute the relatedness of entity e_1 and e_2, run personalized PageRank two times, teleporting only to e_1 for the first run, and teleporting only to e_2 for the second run. The relatedness is then the cosine similarity of the two PageRank vectors. The description of the PageRank algorithm is in Section 2.1.2.
- *CoSimRank* [64]: expands on the previous approach by adding synonyms and translations of words.
- *WikiWalk* [65]: expands on the Personalized PageRank approach, but changes teleportation vectors to the ones provided by the Explicit Semantic Analysis method discussed before.
- *Commute Time* [66]: the Commute Time between nodes u and v is defined as the average number of steps that a random walk, starting at node u, takes to reach node v for the first time and then comes back to u.
- *DeepWalk* [58]: performs a random walk from a focus node, and then uses Word2vec to compute an embedding of the generated sequence of nodes.
- *Line* [67]: variant of DeepWalk, but adapted to weighted and directed graphs.

2.4 BIOMEDICAL NLP

In this section, we describe a small set of biomedical NLP tools. Additionally, we show some issues with the naming of bio-entities, more specifically genes (Section 2.4.1), and a set of recent general-purpose LLMs that were adapted to the biomedical field (Section 2.4.2).

Gu et al. [68] propose *MarkerGenie*, an approach for automatically generating and labeling datasets. This is useful because of the extremely high volume of unlabeled data present in the biomedical field. The authors claim that such tools are valuable because they can provide thorough literature surveys. *MarkerGenie* models the problem of finding relevant biomedical entities as a sentence-level binary/multiple relation classification task. They then manually add a set of rules to ensure that as much noise as possible is reduced (e.g., remove terms shorter than 4 characters unless they are found to be an abbreviation). They then generate artificial training data by exploiting a co-occurrence frequency matrix, which captures the common sense that relations that are common in the text are more likely to be *reliable*.

Zheng et al. [69] propose one of the first (2015) entity linkers for biomedical applications. To achieve this feat, the authors build a KG from 300 biomedical ontologies. Then they extract the mentions using a name tagger and use regular expressions to merge together mentions that were considered separate but could be merged together. They retrieve the candidate entities through a dictionary lookup, which contains both entity mentions and aliases. Lastly, they rank the candidate entities via an entropy-based non-collective approach. The main idea is to assign the entities with higher popularity a higher score. To reach collective agreements between different entities, the authors adopt a collective inference approach, which analyzes relations among multiple entity mentions and ranks the candidates simultaneously.

Abdurxit, Tohti, and Hamdulla [70] propose a biomedical entity linker based on attention mechanisms, with a specific focus on annotation speed. Their method relies on some preprocessing of the text on which to perform EL. More specifically, they replace all the abbreviations in the text with the full name of the relative bio-entity using the study from Sohn et al. [71]. Additionally, the authors uniform the format by removing numerical patterns in biological mentions, along with removing punctuation. In order to generate candidate entities, they compute similarity scores for each pair of biomedical mentions in the corpus and entities in the KB, thus returning the top entities as candidates. The ranking of the candidate entities is performed by a NN.

Chen, Varoquaux, and Suchanek [72] propose a lightweight neural model for biomedical EL, by adapting the well-known LLM BERT [37] to the task of EL, greatly reducing the number of parameters in the process. Their pipeline starts by similarly pre-processing all names to the tool of Abdurxit, Tohti, and Hamdulla [70]. The candidate entity set is generated by calculating a score for the mention and each entity in the KB, returning the top 20 ones. These scores are computed first by embedding both entity names and mentions into a vector space, and then computing their cosine similarity. Finally, entities are ranked through a deep learning architecture that can approximate a sort of similarity score for each pair of mentions in the corpus and names of the entities in the set of candidates.

For the purpose of this thesis, we will not use any of the above entity linkers, but we will design and implement our own one, called OntoTagME (Chapter 4). We decided to do as such because of three main reasons:

- *cost-efficiency*: most approaches employ NNs, which require a GPU and a large RAM to run efficiently. Our goal is to avoid GPUs and to reduce RAM usage to achieve better efficiency and make the tool widely adoptable.
- *ease of interpretability*: most NN-based approaches are black boxes, meaning that results are presented with no clear reasoning on why said result has been chosen by the machine and why it should be satisfactory. Traditional models, like TagME, offer stronger interpretability, as computations are done solely according to the graph structure.
- *full control*: not all the presented tools are publicly available or maintained. Since we need to tweak the back-end KB, we would like full control over the code.

2.4.1 *Issues with Gene Naming*

Genes are especially hard to be detected by any tool in biomedical texts. This is mostly caused by the fact that gene naming has always been subject to a very chaotic process, resulting in many genes being called following weird patterns like:

- *everyday words*: e.g., gene "blue"[9], gene IGKV1D-39 having *"O2"* as an alias[10], gene *"displaced"*[11], gene *"attenuated"*[12].
- *Inconsistent naming*: e.g., gene BSG is also called *"CD147"*[13]
- *Genes that are named after their function*: e.g., gene *"step"*[14]
- *Genes named after diseases*: e.g., BRCA1 having as alias *"breast cancer 1"*[15]
- *Genes named as stop-words*: e.g., gene *"an"*[16], gene *"with"*[17].
- *Extremely high variability in name length*: reaching a minimum length of 1 letter (e.g., gene *"a"*[18]).

In general, the naming of bio-entities is always messy and rarely standardized for any bio-entity type. Each bio-entity type has one or more ontologies that decide naming and identifiers. The gene situation is particularly critical, since:

- Many tools, including PubTator (more in Section 2.4.5) use NCBI as a standard identification tool[19].

9 blue - https://www.wikidata.org/wiki/Q29719314
10 O2 - https://www.wikidata.org/wiki/Q18039443
11 displaced - https://www.wikidata.org/wiki/Q29726548
12 attenuated - https://www.wikidata.org/wiki/Q29732195
13 BSG - https://www.wikidata.org/wiki/Q14911959
14 step - https://www.wikidata.org/wiki/Q29728945
15 BRCA1 - https://www.wikidata.org/wiki/Q14911959
16 an - https://www.wikidata.org/wiki/Q29708491
17 with - https://www.wikidata.org/wiki/Q29713349
18 a - https://www.wikidata.org/wiki/Q29715454
19 https://www.ncbi.nlm.nih.gov/home/genes/

- HGNC [73][20] proposes standard naming for the human genome.
- MGI [74][21] proposes standard naming for the mice genes.
- FlyBase [75][22] proposes standard naming for the genes of the family of the Drosophilidae insects.
- In general, each species has its own naming consortium.
- GeneOntology: this is an ontology that aims at grouping as much spread information for genes of different species.

2.4.2 *Biomedical GPT models*

One of the first attempts in adapting GPT models (as explained in Section 2.2.4) to the biomedical field was done by Microsoft in 2022 when they released BioGPT[23] [76]. This is a natural course of action, as the transformer systems that GPT models are based on (i.e., BERT [37]), received multiple adaptations to the biomedical field: such as BioBERT [77], PubMedBERT [78], SciBERT [79].

The process of building a bio-GPT model is very similar to that of general GPT models. The pre-training phase is done on PM, using 15 million abstracts. Those consist of all the valid titles and abstracts in PM before 2021. This also means that any info that was captured after this year will not be in the model. Because of the extremely large model of GPT-3 (roughly 15 billion parameters), the authors choose to employ the GPT-2 model, which results in 347 million parameters.

The fine-tuning process was performed on three tasks: end-to-end relation extraction, question answering, and document classification.

2.4.3 *Injecting Knowledge from Language Models into KGs*

One of the main issues of LLMs is that it is often not easy to plug domain-specific knowledge into the fine-tuning phase with the goal of achieving better results. This is often a crucial point because training from scratch a GPT model is no longer a viable option for most researchers due to the sheer cost and size of the model itself. It would be ideal to leave the costly (both in time, space, energy, and money) process of training the model to the provider of GPT models (such as OpenAI, Google, and Microsoft), such that the user can then simply "plug" their own domain-specific information into the model in a sort of second *fine-tuning* phase.

One of these approaches is *KnowledgeDA* [80], where the authors propose a method to embed knowledge from domain-specific KGs into language models. Their method employs three modules, which correspond to the three main phases of the process:

1. Entity retrieval without using general-purpose entity linkers, but based on non-exact matches based on string similarity, where these similarities are computed between mentions in the text and entities in the KG.

20 Human gene naming consortium: https://www.genenames.org/
21 Mice gene naming consortium: https://www.informatics.jax.org/
22 Drosophilidae gene naming consortium: https://flybase.org/
23 https://github.com/microsoft/BioGPT

2. Data Augmentation, performed exploiting two different views. The first one consists of replacing entities with other close entities belonging to the same category. The second view involves entities that are far in the KG, but they may be helpful because they appear in similar patterns in the text.

3. Augmentation quality assessment is performed to ensure that the augmentation process added significant value to improve the model performance.

At the time of writing this thesis, general-purpose GPT models have been integrated into our research, more specifically in NetME's summarization module (more in Section 5.4). We mention the biomedical application of GPT models and *KnowledgeDA* because they offer a view of interesting opportunities for future works.

2.4.4 *PubAnnotation*

PubAnnotation [81] is an ecosystem based on agile approaches to text mining and annotation procedures. The main goal is to both provide a platform for entity annotation and to share them with the public. This is done in two main ways.

- *PubDictionaries*[24]: This tool enables users to upload dictionaries of annotations through TSV files. A dictionary is a collection of labels and their corresponding identifier. Labels are the natural language terms used to describe the element identified by the identifier specified in the dictionary. After being uploaded, dictionaries can immediately be queried by users through a REST API.

- *PubAnnotation*[25]: This tool exploits dictionaries located on PubDictionaries for text mining. Given a dictionary set and a text, PubAnnotations computes a set of annotations and returns them to the user. Users can evaluate the quality of those annotations, and if they deem their quality to be poor, they can either revise the dictionaries and/or revise the annotations themselves, thus improving future queries.

2.4.5 *PubTator and PubTator Central*

PubTator Central [15] is a web-based system providing automatic annotations of biomedical concepts (e.g., genes, diseases, and mutations) from PM abstracts and PMC full-texts, expanding PubTator [14].

PubTator employs several NLP tools for identifying different types of modeled entities. For example, they employ DNorm [82] for disease NER. PubTator Central provides automated concept annotations of several biomedical concept types (genes/proteins, genetic variants, diseases, chemicals, and species) across all PM article abstracts and PMC full-text articles. It can be queried in five different ways:

- *PubMed*: same thing as querying directly PM.

- *Gene*: returns all articles relevant to a specific gene or gene product.

24 http://pubdictionary.org
25 http://pubannotation.org

- *Chemical*: returns all articles relevant to a specific chemical.
- *Disease*: returns all articles relevant to a specific disease or syndrome.
- *PMID List*: returns articles in the PM Identifier.

PubTator Central also allows for three types of annotation tasks on top of PM.

- *Document Triage*: bio-curators are used to select articles.
- *Entity Annotation*: the tool can extract genes, proteins, genetic variants, diseases, chemicals, and species in textual articles
- *Relationship Annotation*. the tool can extract relations between entities from a specific set, such as protein-protein interactions, gene-disease relations, and similar.

Full-text annotations are one of the main improvements that were presented in PubTator Central against its original version, which was only able to annotate abstracts. For this reason, the technological stack was rethought and re-trained in order to be able to tackle longer and more structured text. For example, its authors switched the disease NER tool to TaggerOne [83]. Additionally, they designed a new disambiguation tool based on Convolutional NN, whose goal was to identify the most likely bio-concept type using the syntax and semantics of both the span being classified and the surrounding words.

PubTator Central sometimes provides the identifier of the found entities. This happens roughly 50% of the time, and it provides external IDs to a standard set of ontologies. The IDs used by this tool are.

- *Genes*: NCBI gene ID;
- *Diseases and Drugs*: MeSH descriptor ID;
- *Chemical Entities*: ChEBI ID;
- *Species*: NCBI taxon ID;
- *Cells*: Cell Ontology ID;
- *Cell lines*: Cellosaurus ID.

PubTator Central will be one of the key components of the thesis, as it will be combined with our tool OntoTagME to obtain better results (more in Section 4.4).

2.4.6 *BERN2*

BERN2 [84] is a Named Entity Recognition and Normalization tool based on NNs. It is able to model a set of crucial biomedical entities: namely gene/protein, disease, drug/chemical, species, mutation, cell line, cell type, DNA, and RNA. It comes with a publicly available REST API[26], although we found it to be often slow and unreliable.

The normalization task is only performed on some of the entities retrieved. This means that roughly half of the returned entities will have a normalized external ID to another database, while the others will be "CUI-less", which means "with no ID".

26 http://bern2.korea.ac.kr/

The NER part of the task is performed by using pre-trained language modes. More specifically, BERN2 employs Bio-LM [85] as a backbone, and then a set of task-specific layers, where each layer corresponds to one of the modeled bio-entity types. To fine-tune the Bio-LM layer, the authors use entity-specific datasets. For example, they use BC2GM [86] for genes and NCBI-disease [87] for diseases.

For the task of Neural Entity Normalization, BERN2 employs rule-based systems, and when they fail, it uses entity embeddings to find the closest matches with respect to the candidate words.

When an external ID is returned, BERN2 effectively behaves as an entity linker. The external IDs come from different ontologies based on the category assigned to the labeled entity. The set of external IDs used in BERN2 is the same as PubTator.

2.5 BIOMEDICAL RESOURCES

Biomedical research papers are one of the most important ingredients to build the networks discussed in this thesis. The most important paper hubs in the biomedical field are the following:

- PM[27] is a massive repository with, at the time of writing this thesis, more than 36 million biomedical papers, and is maintained by the NCBI. PM contains titles and abstracts of research papers coming from MEDLINE, all the most important life science journals, and online journals. This is by all means the most important collection of titles and abstracts in the biomedical field. It is also available for local download through the Entrez APIs [88].

- PMC[28] is a digital archive of biomedical open-access full-texts, also maintained by the NCBI. It contains, at the time of writing this thesis, more than 9 million open access full-text articles. Papers on PM that have an open-access full-text will also have a link in them, redirecting the users to the corresponding PMC article, where they will be available for fruition and download. Roughly one in four papers available on PM will be also available, along with their full-text, in PMC.

- *BioRxiv*[29] is a preprint archive of biomedical papers, operated by the Cold Spring Harbor Laboratory, a non-profit research and educational institution. Peer review is not required for a paper to be shown in bioRxiv, as the goal is for researchers to obtain feedback before publishing to a conference or journal.

- *EuropePMC*[30] and *ResearchSquare*[31] are two biomedical preprint and paper platforms. EuropePMC contains over 43 million items, which include publications, preprints, and various others. It is hosted by the EMBL's European Bioinformatics Institute. The focus of both platforms is on having items coming only from trusted sources.

27 https://pubmed.ncbi.nlm.nih.gov/
28 https://www.ncbi.nlm.nih.gov/pmc/
29 https://www.biorxiv.org/
30 https://europepmc.org/
31 https://www.researchsquare.com/

2.5.1 *Ontologies and Bio-Databases*

Biological resources are a crucial part of any biomedical research. This is because knowledge in biology is usually curated by institutions that deal with specific bio-entities, from naming to any type of standardized feature that is required.

For the purpose of this thesis, we will use a very large set of different ontologies and bio-databases. All the numbers shown in this section refer to the versions of the ontologies and bio-databases we will use for the purpose of this thesis. The details of the versions used in the thesis are in Section 4.2.

Those resources are be described in the following paragraphs.

DRUGBANK [6][32] contains data about *drug names, drug synonyms, drug-drug interaction*, and other comprehensive drug-target information. It contains 13 367 drugs entries, including 2 611 approved small molecule drugs, 1 300 approved biotech (protein/peptide) drugs, 130 nutraceuticals, and over 6 315 experimental drugs. Additionally, 5 155 non-redundant protein (e.g., drug target, enzyme, transporter, carrier) sequences are linked to these drug entries.

HGNG (HUGO Gene Nomenclature Committee) [73][33] assigns unique and informative gene symbols and names to *human genes*. The standardized HGNC-approved nomenclature is fundamental because it is used in publications and biomedical databases to remove ambiguity and facilitate communication between researchers worldwide. It contains more than 40 000 approved gene symbols of which over 19 000 are for protein-coding genes. The HGNC also names a set of small and long non-coding RNA genes and pseudo-genes (659 since 2017). The genes are grouped on the basis of several shared characteristics such as homology, associated phenotype, and encoded protein function.

ENSEMBL [89][34] contains *genome annotation* (e.g., genes, variation, regulation, and comparative genomics) across vertebrates, and it is maintained by the European Bioinformatics Institute. It is effectively a genome browser for the retrieval of genomic information.

DISGENET [17][35] contains collections of genes and variants associated with human diseases. It integrates data from scientific literature, genome-wide association study catalogs, expert-curated repositories, and animal models. Additionally, several original metrics are provided to assist the prioritization of genotype–phenotype relationships. DisGeNET releases two types of databases, GDA and Variant-Gene Association. DisGeNET contains more than 1 million GDAs, linking together 21 671 genes and 30 170 diseases, disorders, traits, and clinical or abnormal human phenotypes. The number of variant-disease associations is 369 554, between 194 515 variants and 14 155 diseases, traits, and phenotypes.

32 https://go.drugbank.com/
33 https://www.genenames.org/
34 https://www.ensembl.org/index.html
35 https://www.disgenet.org/

CIVIC [90][36] is an expert crowd-sourced KB for Clinical Interpretation of Variants in Cancer describing the therapeutic, prognostic, diagnostic, and predisposing relevance of inherited and somatic variants of all types. Its data is fully available for download through its publicly open REST APIs.

PHARMGKB [91][37] is an interactive tool for researchers that investigate how genetic variation affects drug response. It displays genotype, molecular, and clinical knowledge integrated into pathway representations with links to additional external resources. A user may search and browse the KB by genes, variants, drugs, diseases, and pathways.

OBO FOUNDRY [92][38] is the Open Biological and Biomedical Ontology (OBO) Foundry. It provides a family of interoperable ontologies that are both logically well-formed and scientifically accurate. They contribute to developing an evolving set of principles and common syntax based on ontology models that ensure the proper functioning of the system.

THE GENE ONTOLOGY [11][39] project provides a uniform way to describe the functions of gene products from organisms across all kingdoms of life. it contains more than 44 000 GO terms, 7 million annotations, 1.5 million gene products, and 5 thousand species.

THE HUMAN DISEASE ONTOLOGY [4][40] is a standardized ontology for human disease with the purpose of providing the biomedical community with consistent, reusable, and sustainable descriptions of human disease terms, phenotype characteristics, and related medical vocabulary disease. It contains roughly 10 000 entries, and it is widely used as a standard for diseases.

PATHWAY ONTOLOGY [12][41] is a controlled vocabulary for pathways that provides standard terms for the annotation of gene products. The goal of the Pathway Ontology is to cover all types of biological pathways and to capture the relationships between them within the hierarchical structure of a Directed Acyclic Graph (DAG).

PROTEIN ONTOLOGY [93][42] defines a set of taxon-specific and taxon-neutral protein-related entities in three major areas: proteins related by evolution, proteins produced from a given gene, and protein-containing complexes.

BRENDA tissue/enzyme source [94][43] is a structured controlled vocabulary for the source of an enzyme comprising tissues, cell lines, cell types, and cell cultures. It is maintained and developed at the Institute of Biochemistry and Bioinformatics of the

36 https://civicdb.org/
37 http://www.pharmgkb.org
38 http://obofoundry.org/
39 https://geneontology.org/
40 https://disease-ontology.org/
41 https://bioportal.bioontology.org/ontologies/PW
42 https://proconsortium.org/
43 https://www.brenda-enzymes.org/

University of Braunschweig. Data on enzyme function are extracted directly from the primary literature by scientists holding a degree in biology or chemistry. Formal and consistency checks are done by computer programs, each data set on a classified enzyme is checked manually by at least one biologist and one chemist.

ANATOMICAL ENTITY ONTOLOGY [95][44] is an ontology of anatomical structures that expands CARO, the Common Anatomy Reference Ontology, to about 160 classes using the "is_a" relationship; it thus provides a detailed type classification for tissues. The AEO is useful in increasing the amount of knowledge in anatomy ontology, facilitating annotation, and enabling interoperability across anatomy ontology.

PHENOTYPE AND TRAIT ONTOLOGY [96][45] is used in conjunction with other ontologies such as Gene Ontology or Anatomical Ontology to refer to phenotypes. Examples of qualities are red, ectopic, high temperature, fused, small, edematous, and arrested.

CELL ONTOLOGY [97][46] is designed as a structured controlled vocabulary for cell types. This ontology covers cell types from prokaryotes to mammals, excluding plant cell types.

CELL LINE ONTOLOGY [98] is a community-driven ontology developed to standardize and integrate cell line information and support computer-assisted reasoning.

THE STRING [16][47] database collects and integrates functional interactions between the expressed proteins for a large number of organisms. These associations include direct (physical) and indirect (functional) ones. For each protein–protein association stored in STRING, a score is provided. These scores indicate the estimated probability that, given the supporting evidence, an interaction is biologically meaningful, specific, and reproducible. This supporting evidence can come from experiments (from a laboratory experiment), databases (asserted by a human expert), textmined (present in an abstract or a line in full-text), and more.

UNIPROT [99][48] is a database of protein sequence and functional information. It is maintained by the UniProt consortium, which consists of several European bioinformatics organizations and the Georgetown University Medical Center in the USA. It consists of four main components:

- *UniProt Knowledge-base* (UniProtKB): expertly curated protein database with cross-references to multiple sources.
- *UniProt Archive* (UniParc): sequence repository reflecting the history of all protein sequences.

44 http://obofoundry.org/ontology/ehdaa2.html
45 http://obofoundry.org/ontology/pato.html
46 http://obofoundry.org/ontology/cl.html
47 https://string-db.org/
48 http://www.uniprot.org/

- *UniProt Reference Clusters* (UniRef): merges closely related sequences based on sequence identity in order to reduce the elapsed time for search queries.
- *UniProt Metagenomic and Environmental Sequences* (UniMES): repository developed for the metagenomic and environmental data.

Biomedical ontologies and DBs are key components of the tools studied and implemented in this thesis, as they will be deeply integrated into BioTagME and the first version of OntoTagME. More details on this integration are in Chapters 3 and 4.

2.5.2 *Biomedical Knowledge Graphs*

The task of building biomedical KGs is not a new one. In particular, the three most common tasks of this type are:

- Build a small biomedical interaction network [10, 21]. The most relevant use case is the one of literature review. In particular, researchers in the biomedical field usually retrieve the most relevant/recent papers on the topic they want to catch up on, and they build interaction networks between the various bio-elements.
- Build large KGs [100–102]. This is most useful for obtaining a broader view of a field. For example, *drug re-purposing* is the task used to accelerate the drug discovery process by identifying novel clinical uses for an existing drug approved for a different indication. One way in which this is achieved is through mining for *similar* patterns in the behavior of different drugs/diseases. These patterns can be retrieved from a large KG because it documents the behavior of drugs with respect to many different types of bio-entities.
- Build highly specialized networks [16, 17] on some specific bio-entity type. One example is genetic interaction networks, which are especially useful for understanding the relationship between genotype and phenotype.

Nicholson and Greene [103] describe the current landscape of biomedical KG construction. They state that most of the early attempts in building bio-KGs were done by integrating knowledge coming from several different manually-curated ontologies. Examples of those KGs are PharmGKB [91] and UniProt [99]. Some of those manually curated KGs employ a semi-automatic pipeline. This pipeline aims to take a large set of long texts and sentences from papers, extract all the phrases where two entities co-occur, and then provide those sentences to human annotators, tasked with manually integrating this knowledge into the set of relations of the KG. However, more recently, more and more automatic methods have been employed for full KG construction. Those methods employ NLP techniques and text-mining to extract entities and relationships from texts. Those techniques can either be:

- *Rule-based*: exploit grammar rules and pattern matching.
- *Unsupervised*: exploit co-occurrence probabilities and clustering techniques. For example, STRING [16] is built using an unsupervised approach on PM abstracts.

- *Supervised*: they use labeled sentences from well-known datasets to construct generalized patterns. They then use these patterns on new text to extract relations.

The authors also provide a set of fields where the use of bio-KGs can be beneficial: such as gene expression studies, pharmaceutical applications, and clinical applications.

BIOS [104][49] is an algorithmically generated biomedical KG containing about 4 million concepts, 7 million terms, and 7 million relations. For entity extraction, they use DS-NER [105], a NER tool based on Distant Supervision. The authors train a specialized annotator to predict the category of an entity using both its mention and its surrounding text. As a ground truth, they extract the mentions from UMLS [106], along with their associated categories (among 18 distinct ones). The final goal is to extract relationships between bio-entities. For this, they mine PM abstracts to find phrases where the source and target entities co-occurred as head and tail of the phrase respectively. These phrases constitute the set of positive examples (i.e., the relationship between the two entities exists). The authors then artificially synthesize a set of negative samples, which are used to balance the training set. On these positive and negative examples, they train a classifier, which is eventually used to predict new relations for their KG.

SPOKE [101][50] is a massive bio-KG mined from a very large set of ontologies and biomedical databases. They find 27 million nodes of 21 different types, and 53 million edges of 55 types, using a total of 41 bio-databases. SPOKE is updated on a weekly basis. The authors also provide a publicly available REST API and a web GUI to perform neighborhood queries. The authors provide a set of potential use cases, like drug discovery, anatomy-driven searches, food-driven searches, and disease-driven searches.

CLINICAL KNOWLEDGE GRAPH (CKG) [107][51] is built from ontologies, bio-databases, and literature (7 million PM abstracts). The focus of this work is on the interpretation of clinical proteomics data. Their KG consists of 36 different node types (bio-categories) with 47 different relationship types, for a total of 20 million nodes and 220 million relationships. The authors provide a set of potential uses for their clinical KG, which include bio-marker discovery (an indicator of the severity or presence of some disease state), alternative treatment options analysis, and drug re-purposing.

RTX-KG2 [108][52] is a biomedical KG that integrates knowledge coming from 70 ontologies. The KG contains roughly 6 million nodes and 40 million edges and it represents biomedical entities and their relations using the Biolink [109] model, a high-level ontology mapping semantic and relation types to other ontologies. The user can install the KG locally and then query it through the provided Neo4j application.

49 https://bios.idea.edu.cn/
50 https://spoke.rbvi.ucsf.edu/
51 https://data.mendeley.com/datasets/mrcf7f4tc2/3
52 https://github.com/RTXteam/RTX-KG2

XU ET AL. [110] build a bio-KG with the goal of relating bio-entities to their respective most relevant author specialists (most relevant in terms of publications). This affiliation information is extracted from the National Institutes of Health (NIH), which collects the affiliation history and educational background of authors from ORCID. To extract entities, they employ a NER tool based on BioBERT [77]. The disambiguation phase is handled by 5 different tools, one for each modeled macro-category (e.g., species are disambiguated through dictionary lookup). The disambiguation of authors is instead done by integrating different external affiliation/paper databases. In the end, they were able to find 114 345 178 total entities. Those are not related to each other, but relationships are drawn towards the "authoritative authors" relative to them, along with the affiliation of those researchers.

BIOKG [22] is a biomedical KG that combines information coming from 13 open biomedical databases. The KG contains roughly 100K nodes and 2M edges. Edges are also extracted from ontologies and their focus is on supporting the most well-known bio-entity interaction types (e.g., disease–pathway associations, protein relations to diseases). In total, BioKG models a total of 17 different relationship types. The authors also provide a software library for mapping bio-entities from their IDs in one ontology to other ones, called *BioDBLinker*.

DARLING [21][53] is a tool for building on-the-fly biomedical networks derived from a full KG pre-built from a set of 881 185 PM abstracts up to 2021. It uses NER to extract bio-entities from PM articles, focusing mostly on disease-centric ontologies (i.e., ontologies containing either only diseases or relationships between diseases). The NER tool of choice is EXTRACT [111], which is entirely based on dictionaries. Edges are created in case of co-occurrences between bio-entities in PM abstracts, and their weight is stronger for entities that co-occur frequently. DARLING offers an interactive GUI for customizing the query and analyzing in depth the generated graph. For example, users may search for the term "HIV", and DARLING will provide them with the set of publications that are relevant with respect to this query, mining this information from DisGeNET [17]. Users can then manually select the set of articles they want to build the network on through the browser. DARLING will then build said network on the PM abstract of those articles, and show the network, a word cloud, and other relevant plots to the query.

HETIONET [100][54] is a heterogeneous network of biomedical knowledge constructed from a collection of 29 publicly available databases and millions of publications. The network models nodes of many different types (e.g., Genes, Diseases, Compounds). In addition, the relationships have been grouped into several categories such as "up/down-regulates", "inhibit", etc. The v1 of Hetionet contains 47 031 nodes with 11 different data types and 2 250 197 edges that represent 24 different relationship types. It was originally created as part of Project Rephetio, whose goal was to predict new uses for existing drugs. In the last few years, it has been modified to address a broader variety of purposes:

53 https://bib.fleming.gr:8084/app/darling
54 https://het.io/

- *Drug Re-purposing*: perform heterogeneous edge prediction to foresee which disease is treated by each available compound.
- *Prioritizing Disease-Associated Genes*: exploit edge prediction for genes that are associated with diseases, based on a predecessor network.

NICHOLSON ET AL. [102] integrate the knowledge coming from Hetionet [100], and the abstracts in PM, exploiting the annotations coming from PubTator Central [15]. They focus on a very small set of bio-entity types (i.e., genes, diseases, compounds), and to retrieve edges, they search for co-occurrences of all the entities present in Hetionet inside the PM abstracts. The edges are extracted through the spaCy dependency parser, which allows the authors to understand dependencies between all the found co-occurring entities. In this case, the goal of the authors is not to build a biomedical KG, but to design a methodology for the creation of edges (and edge labels). The authors apply their methodology and analyze their results only for a very small set of entity types.

YUAN ET AL. [112] build a biomedical KG from PM abstracts. To extract the knowledge needed for building the graph, the authors first use a dictionary-based NER. They then perform entity disambiguation by checking the candidate mentions found in the NER phase, and discarding improbable ones heuristically. Then they take the entity that statistically maximizes the probability of being associated with that candidate mention. Entities are represented through embedding vectors and relations are derived through the embeddings by doing a sort of "subtract" operation for each entity embedding vector couple. Finally, they run a variation of the KMeans algorithm to cluster all the computed embedding vectors of the relations into distinct sets. Intuitively, each cluster corresponds to a different relationship label (e.g., "symptom_of", "gene-disease").

THIS THESIS aims to provide quality biomedical networks. For this purpose, we envisioned three applications. The first one is BioTagME (described in Chapter 3), which is a biomedical KG, where nodes are extracted with TagME, properly adapted filter out all of the non-biomedical annotations. The second application is NetME (described in Chapter 5), which is an on-the-fly network builder that is entirely based on a custom biomedical view of Wikidata. Nodes are extracted with OntoTagME (described in Chapter 4), a tailor-made low-cost entity linker that is built on top of TagME. The third application is a planned future work, whose goal is to build the first biomedical KG based on PMC full-texts.

TABLE 2 summarizes the relevant information of the KG described above. In the case of NetME, the "Nodes" column shows the number of unique bio-entities modeled. We do not show the works from Nicholson, Himmelstein, and Greene [102] and Yuan et al. [112], because their goal is not to build a KG, but to propose a methodology for it. DARLING [21] is shown among the full-KGs because it statically builds a large biomedical KG, and it performs a sub-graph extraction on the fly at query time. The number of nodes shown for Xu et al. [110] only contains the bio-entities, and disregards all nodes modelling affiliations and authorships. Additionally, bio-entities

Type	KG	Uses	Nodes	Edges	Node Types	Edge Types
Precomputed	BIOS [104]	abstracts	27M	70M	27	13
	SPOKE [101]	ontologies	27M	53M	21	55
	CKG [107]	abstracts and ontologies	16M	220M	36	47
	RTX-KG2 [108]	ontologies	6M	39M	56	77
	Xu [110]	abstracts	490K	-	5	-
	BioTagME [8]	abstracts and ontologies	162K	41M	9	W
	BioKG [22]	ontologies	100K	2M	5	17
	DARLING [21]	abstracts	79K	5M	10	W
	Hetionet [100]	ontologies	47K	2.3M	11	24
on-the-fly	**NetME** [10, 13]	full-texts and abstracts	3.6M	-	15	28

Table 2: Biomedical Knowledge Graphs recap, sorted by decreasing number of nodes. Tools presented in this thesis are bolded. The numbers shown are approximated

are not connected in this KG, as the focus is on relations between authors, projects, funding, and bio-entities. We put "W" as a value in the "Edge Types" column when the bio-KG uses weighted unlabeled edges.

In Section 5.5.4, we draw a comparison between the main tool designed in this thesis, named NetME, and the known biomedical KGs just described above. Here, we anticipate a summary of the main points that make NetME a successful choice with respect to the known KGs:

- No need to *update* the bio-KG periodically with up-to-date information, since the most recent articles will be present in PM and PMC.
- It is the only tool to work on PMC *full-texts*.
- It offers a powerful *categorization* ability, having both a wide set of macro-categories (e.g., genes, proteins, diseases), and a large set of other categories that offer more detailed information (e.g., tumor marker, tumor suppressor gene)
- its edges are *labeled*, and they carry valuable informative biomedical information (e.g., "up-regulates", "inhibits"), which are useful to explain the query results.
- Its performance on edge prediction is the best out of all known network-building tools. To test this, we have evaluated the known tools and NetME over a task of edge prediction for 100 random DisGeNET GDA annotations.

3

BIOTAGME

BioTagME is the first research step we worked on for building an effective biomedical KG on abstracts of research papers. The goal of this study was mostly exploratory, to see whether building a biomedical KG using *simple* algorithmic bricks was possible.

We aimed to achieve this goal by following this set of design principles:

- Employ strong and *well-established* technology.
- Results of the various algorithms can be *easily explained and justified*.
- Integration of *external biological ontologies*.
- Effective information mining from *scientific papers*.
- Broad coverage both for the papers both in *dates* and in *topics*.
- Working only on the simplest and most information-dense snippets of text (*titles and abstracts*).

BioTagME was published in 2022 in Frontiers in Genetics [8], and this publication came with a dataset on Zenodo [9] (complying with *FAIR* principles [113]), and with a web app[1] where users are able to query the full KG.

More technically, BioTagME is a framework backed by two different pipelines (Figure 2) for building a biomedical KG from PM documents' titles and abstracts.

The first pipeline is the core of the framework. It transforms the whole set of PM abstracts and the ontological knowledge into nodes and edges of the KG. The second pipeline allows a user to extract information from the graph.

The first pipeline is built on top of the Apache SPARK analytic engine and Hadoop Distributed File System (HDFS). This was a deliberate choice. The sheer amount of papers present in PM demanded a scalable implementation by design (36+ million papers at the time of writing this thesis). These engines guarantee large-scale data processing through cluster managers (Apache Meson, YARN, Stand Alone, Kubernetes). The first pipeline, described in Section 3.1, collects data coming from several freely available online ontologies into Pandas DataFrames. In addition, the complete set of PM titles and abstracts is stored using the Spark SQL language, and they are also processed in order to retrieve other entities and relationships needed for building the graph. Info from bio-ontologies, bio-DBs, and PM abstracts are then used in order to build a complete life science KG.

Sections 3.1 and 3.2 will describe in detail the first (back-end) and second (front-end) pipeline, respectively focusing on all the steps and their inner workings.

Data processing is done in PHP and bash to achieve high performance. In addition, all the GUI modules have been realized natively in REACT.

1 https://biotagme.eu/

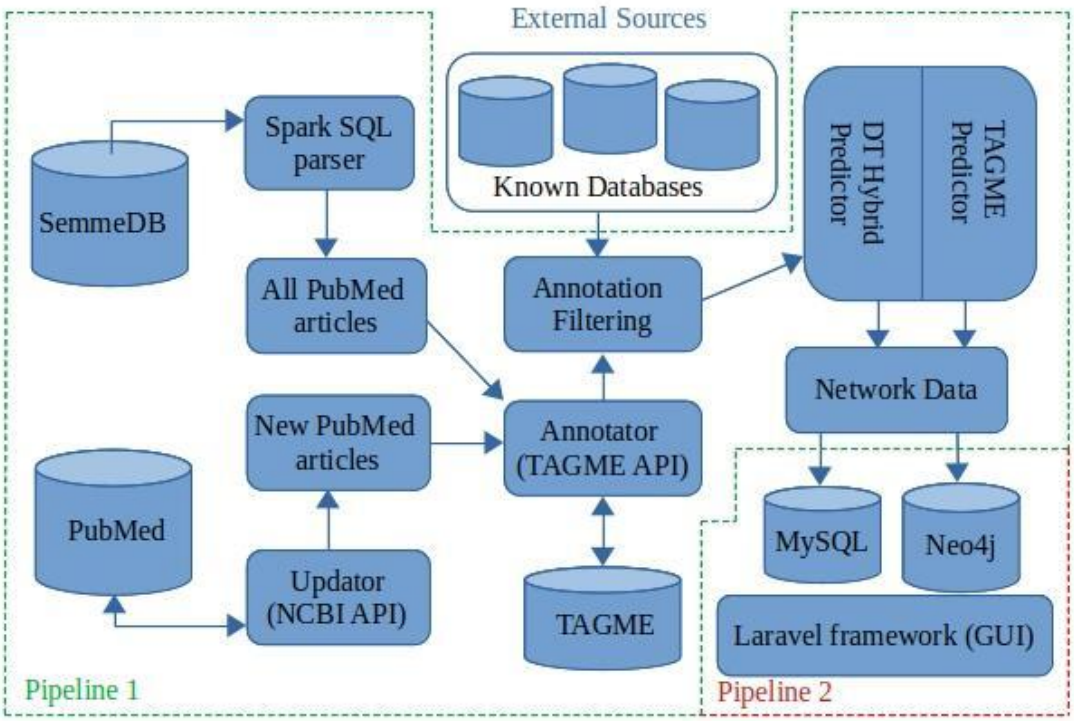


Figure 2: BioTagME pipelines

3.1 PIPELINE ONE: BACK-END

This section describes all components and functionalities of the first pipeline underling BioTagME. This pipeline is the one that effectively loads all the required data, processes them, and builds the network.

3.1.1 *Download and Import*

Firstly, the pipeline loads and imports all the necessary ontologies into the HDFS through a custom bash script. This is done in three phases:

1. First, it downloads titles and abstracts of PM articles through the SemmedDB SENTENCE table [114]. This table contains all the sentences of all the articles and abstracts present in PM.

2. Then, it downloads the ontologies (the full list is provided in Section 3.1.3). They are used for:
   - filtering noisy annotations, caused by the fact that TagME is not perfect, and its disambiguation process can yield some errors. Additionally, Wikipedia is a general-purpose KB, which is not a perfect fit for biomedical applications
   - building edges that model biomedical evidence resulting from lab experiments, biomedical and biophysical processes, and information coming from research papers.

We also use these edges to evaluate the quality of BioTagME prediction.

3. The import section transfers the downloaded DBs from the local file system to the HDFS.

3.1.2 *SQL to JSON parser*

Although SemmedDB guarantees faster downloads than NCBI Entrez APIs, it comes with some perks, that make the two API endpoints different:

- title and abstract of each document in PM are split into sentences;
- sentences are passed in an SQL format, which, by default, is not supported by the Spark engine.

To solve these issues, we design a new Spark module, named *SQL2Json parser*. The goal of this module is to extract headers and data rows from a table (in our case, the sentence table) by applying Spark SQL Window methodology. We process each row in order to reconstruct the full titles and abstracts, aggregated through Spark built-in functions such as *collect_list, concat_ws, and group-by*. Finally, we convert the parsed data into JSON format and store it within the HDFS.

3.1.3 *Ontologies integration*

As previously mentioned, several ontologies are integrated into our system to obtain better coverage of entities and relationships. However, there are a few issues to consider:

- Different DBs often use different words to describe the same entity. For example, DisGeNET uses *"Colorectal cancer, hereditary nonpolyposis, type 1"*, while DiseaseOntology uses *"Lynch syndrome 1"* to refer to the same disease.
- Attributes are named differently from one DB to another. For example, an ontology might use the attribute name *"mirna_nr"*, while another one might use *"id"*.
- Different ontologies might use different file formats (JSON, XML, TXT, CSV, TAB, OBO, GTF, FASTQ, SQL, etc. . .).

To tackle those issues, we use a module that executes the following pipeline.

First, we load all the ontologies into Spark DataFrames, for which we used simple built-in functions of the engine for all the files in a standard and recognized format.

To import OBO, GFT, SQL, and FASTQ files, we employ custom Spark modules that convert those files from their original format into DataFrames. The Databricks Spark-XML[2] library is used for XML files.

Then, we process each DataFrame and we subject it to a schema redefinition by using ontology metadata, synonyms list, and references (toward other ontologies) list

2 https://github.com/databricks/spark-xml

to harmonize the contents of the different data sources. This module is a fundamental intermediate layer that transforms all ontologies into new ones having the same schema, attributes, format, and nomenclature.

More specifically, the full list of integrated ontologies is shown in Table 3. Note that some ontologies, such as DrugBank, PharmGKB, and Brenda, require free registration or authorization in order to be downloaded. Such a procedure is left to the user.

Source Name	Citation	Data Type
DisGeNET	[17]	human gene-disease association
DiseaseOntology	[4]	human disease
DiseaseEnhancer	[115]	human disease-associated enhancer
DrugBank	[6]	drug and drug target
PharmGKB	[91]	human-genetic var. on drug resp.
HGNC	[73]	human gene
ENSEMBL	[89]	vertebrates genomic information
LNCipedia	[116]	human long non-coding RNAs
miRcode	[117]	human microRNA-target pred.
miRBase	[118]	microRNA sequences
miRTarBase	[119]	microRNA-target interactions
miRCancer	[120]	microRNA expr. profile in cancer
Reactome	[121–123]	pathway
PathBank	[5]	pathway
UniProt	[99]	protein sequence
STRING	[16]	protein-protein interaction
BRENDA	[94]	enzyme

Table 3: BioTagME Ontologies

3.1.4 *Annotation*

This module is tasked with the real annotation of titles and abstracts of the previously retrieved and stored PM articles. Thus, for each document d_i, we issue a query to the TagME API through an HTTP POST request. A query is performed for each article, and the text we send is a union of the title of the document and its abstract, which we denote by TI_AB.

Under the hood, TagME removes all stop-words and punctuation symbols from the TI_AB text at first. Then TagME sends back to the user a list of annotations, formatted as described in Section 2.3.1. Each annotation contains its mention (part of the query text where that entity was identified), the corresponding Wikipedia page title, Wikipedia page categories (they are listed at the very end of each Wikipedia page), and Wikipedia page ID. Each entity will be a node of the KG we are going to build.

TagME annotations are not fully accurate. The authors of the original paper [7] provide an estimate F_1 measure of 0.78, where F_1 is the harmonic mean between the precision and the recall of the annotation process. However, this does not consider any change in performance due to

- more up-to-date Wikipedia dumps: the original paper is from 2010, while the API version on the SoBigData infrastructure[3] works on top of a dump from march 2017.
- The filtering employed by BioTagME in order to remove all the general purpose and irrelevant entities for our task. For this pruning process, we used the Biological Portal of Wikipedia[4] and took only the entities that have at least one category from this restricted set.

Finally, the documents, along with their annotation entities extracted by TagME and then filtered, are sent to the prediction module to generate the relationships (edges).

3.1.5 *Prediction*

Our methodology aims to predict a potential relationship between the i-th entity and the j-th entity based on the BioTagME score value ($BioTG_{i,j}$). This score is defined as the product between the DT-Hybrid score $s_{i,j}$ and the TagME relatedness score $r_{i,j}$. The higher the score value, the higher the meaningfulness of the predicted relationship.

DT-Hybrid, which was described in Section 2.1.3, returns a recommendation score based on a bipartite network projection technique that implements the concept of resource transfer within the network and predicts the robustness of the relationship between a pair of entities. The TagME relatedness score is computed through the online TagME service available on the SoBigData infrastructure[5]. Its value is in the range $[0, 1]$ and expresses how much two entities are semantically related within the Wikipedia corpus. The value zero means no relationships between them; the value one means equivalence between two entities.

The output of the prediction module is a set of relations between entities that represent the edges of the bio-KG under construction. These relations are then integrated with others coming from the ontologies.

3.1.6 *Network Construction*

As soon as the documents have been annotated and the prediction module has been completed, the last step of the pipeline is to build the KG containing logical or physical relationships among biomedical elements. Physical relationships represent the real connection between biomedical entities. Instead, the logical one represents the effect that a biomedical entity (e.g., Drug) could have on another one (e.g., Disease or Gene)

For every $Entity_i$–$Entity_j$ association obtained by the prediction module, BioTagME creates three different edge types:

3 https://sobigdata.d4science.org/group/tagme/tagme-help
4 https://en.wikipedia.org/wiki/Portal:Biology
5 https://tagme.d4science.org/tagme/

- Literature: indicates an interaction derived from a research paper, describing a piece of biological evidence resulting from laboratory experiments, biological and biophysical processes, and more.
- STRING: represents the predicted protein-protein associations derived from the STRING *Homo sapiens* database.
- BioTagME: the edges predicted by our tool as described above.

Both BioTagME edges and STRING edges are marked with the corresponding score value to indicate the interaction's likelihood.

3.1.7 *Updating*

BioTagME annotates PM abstracts in order to predict the relationships among their corresponding biomedical entities. Of course, those predictions become outdated as time goes on, and more and more papers are submitted to conferences and journals, eventually becoming available on PM. Therefore, a periodical update is needed.

BioTagME carries out the following steps to achieve this purpose. First, it downloads all the PMIDs (Documents' identifier in PM) within an established date range through an NCBI esearch POST request [88]. The start date usually refers to the last updating date, whereas the end date is usually set to the date when the refresh procedure takes place.

Once the PMIDs list has been obtained, the updating module downloads the title and abstract of these PMIDs using the NCBI efetch API [88]. For performance reasons, the PMIDs list is partitioned into chunks of proper size, and then several chunk-based NCBI efetch post requests are generated and sent to the PM server to obtain the required data. NCBI does not impose a maximum on the number of requests to be submitted, especially when a POST request is used. However, we suggest keeping this value under 10 000 to reduce the computational burden of this job.

Once the papers from PM have been downloaded, we re-run the previous phases of the pipeline (annotation, prediction, and network construction procedures) on the newly downloaded set of titles and abstracts, to update the nodes and edges of the KG.

The update procedure is incremental. It does not require the entire PM abstracts corpus. It runs on a subset of abstracts within the specified date range, and it can generate a KG only on those abstracts. This also opens up the possibility to create temporal KGs over a certain topic of interest.

3.2 PIPELINE TWO: FRONT-END

This second pipeline is tasked with the import procedure of the KG into the Neo4j DB, so to support queries on the biomedical network. Important types of queries include: getting the neighborhood of a biomedical element (representing bio-entities that are connected to the query one) and computing the shortest path between two nodes.

The interface module for network querying is crucial to exploit such graphs and infer potential novel biomedical relationships. We employ the Laravel "model view controller" and the React Native framework to implement the graph view back-end

and web page components. In what follows, we describe such modules and refer the readers to the web application, available at https://biotagme.eu, whose welcome page is shown in Figure 3 below.

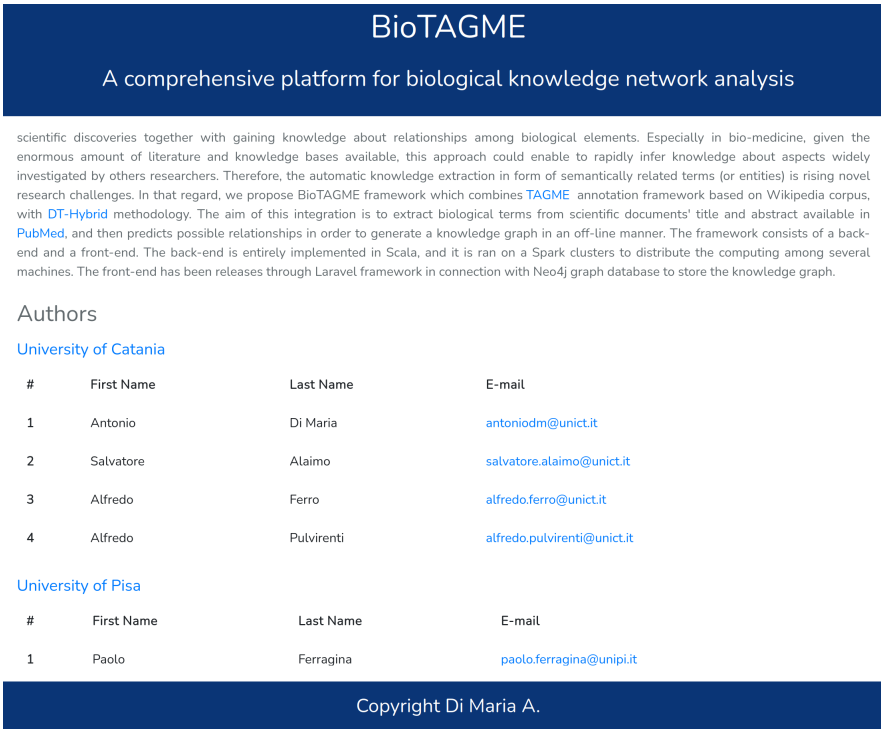


Figure 3: BioTagME welcome page

3.2.1 *Network Import*

A user may access the upload section through the "biological element search" panel by clicking on the "network files upload" link (Figure 4, left side). Then the user has to follow a three-phase procedure:

1. First, the user has to do an "authentication phase" to ensure that only authorized users may execute the import procedure. A snapshot is shown in Figure 4.

2. Second, the "files selection phase" is enabled (as shown in Figure 5, left side), and the user can select both "nodes.csv" and "edges.csv" files containing the network components, and the "Name_Aliases.csv" file about biomedical element aliases. Since the size of the files is large (order of GB), our system uses the "Pion" library[6] to split the file into small chunks (client side) and to reassemble them as soon as these are correctly received (server side).

3. Finally, after all the files have been received, the user can trigger the loading of the network on Neo4j by clicking the import button (Figure 4, right side).

6 https://github.com/splunk/pion

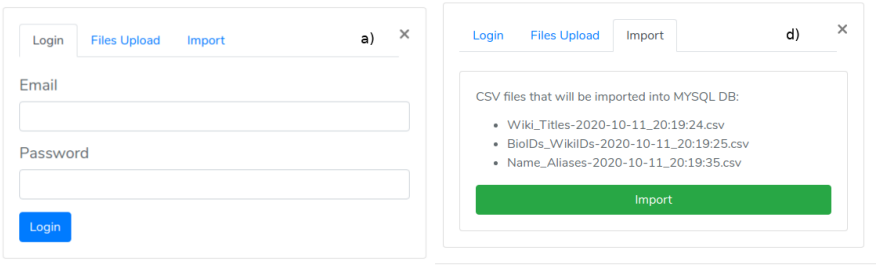


Figure 4: BioTagME login and recap prompt

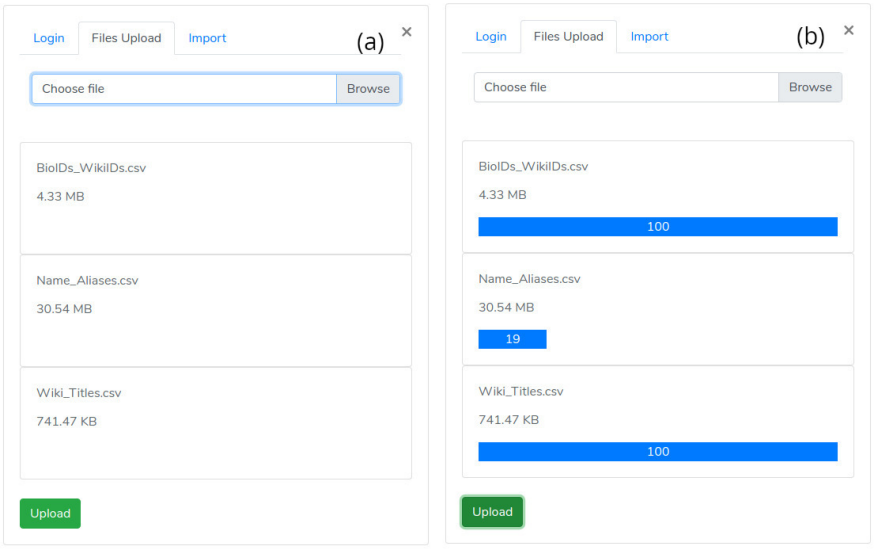


Figure 5: Upload Panel

3.2.2 *Searching Module*

Once the network has been imported, the user may execute several types of queries through the GUI composed of the following panels:

- *Searching panel* (Figure 7): shows a snapshot of this panel in the GUI, more info on its algorithms and inner workings are shown later.
- *Graph panel* (Figure 6): shows the graph view on the network of interaction between the *biological process* "blood coagulation" and any gene. In this specific example, we set a limit of 30 on the number of nodes to be displayed for easier readability.

Edges are colored based on the way they have been extracted:

- Yellow edges represent the set of edges predicted by BioTagME.
- Red edges represent the ones that were found in ontologies and external databases.
- Orange edges are predicted both by BioTagME and extracted from the ontologies.

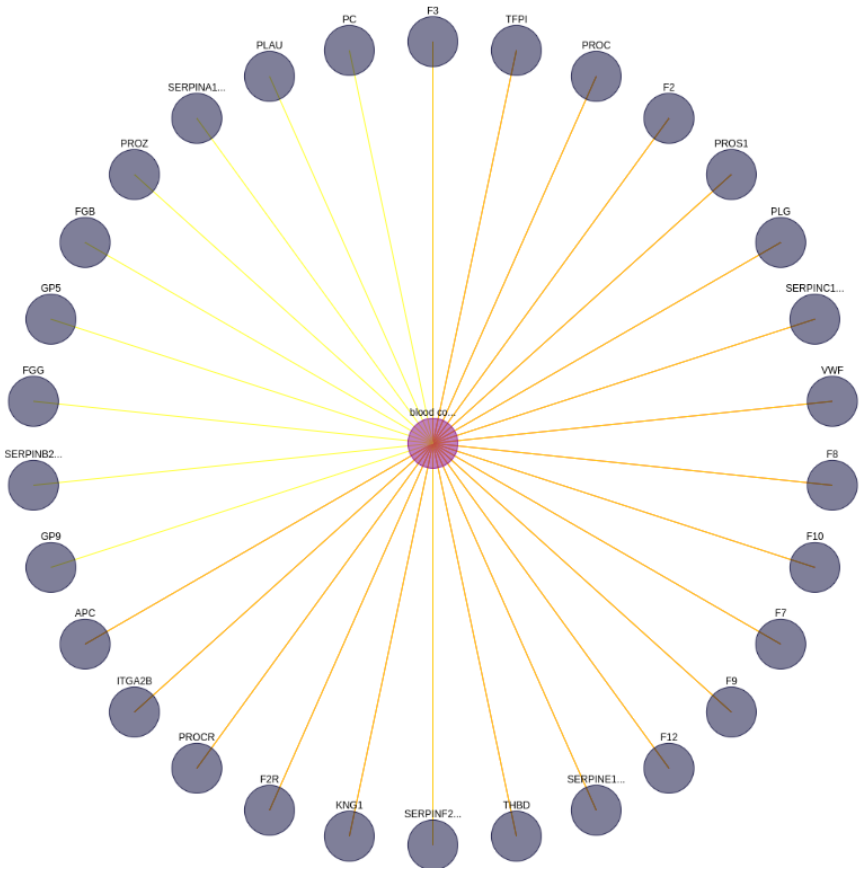


Figure 6: Blood coagulation - gene interaction network

The "Searching Panel" is used for setting the query parameters based on the selected menu:

- When the *Echo Network* option is selected, a user may search the *Echo Network* of a *biomedical entity*, say "be_i". Therefore, the user provides the type and/or the name of the biomedical entity to be analyzed "be_i" (the red rectangle highlighted in Figure 7) and the type of the other entities (the orange rectangle in Figure 7) to include within the echo network. To avoid building a large graph, a maximum number of entities has to be supplied (the allowed range is from a minimum of 10 to a maximum of 200 nodes) through the "Top n" section (shown on the green rectangle of Figure 7). Once all the required parameters have been set, the search process can be triggered by clicking the Submit button. This process transforms the specified parameters in a "Cypher query" (Neo4j proprietary query language) that looks for the "Top n" nodes having one or more links from/to the biomedical entity "be_i".

- When the *Shortest Panel* option is selected (Figure 7), a user looks for the shortest path between two biomedical entities. First, the user specifies the type and name of the source "el_src" and destination "el_dst" entities (the red rectangle in Figure 7), and then BioTagME transforms all these parameters into a proper "Cypher query" which is mainly based on a Neo4j shortest path computation.

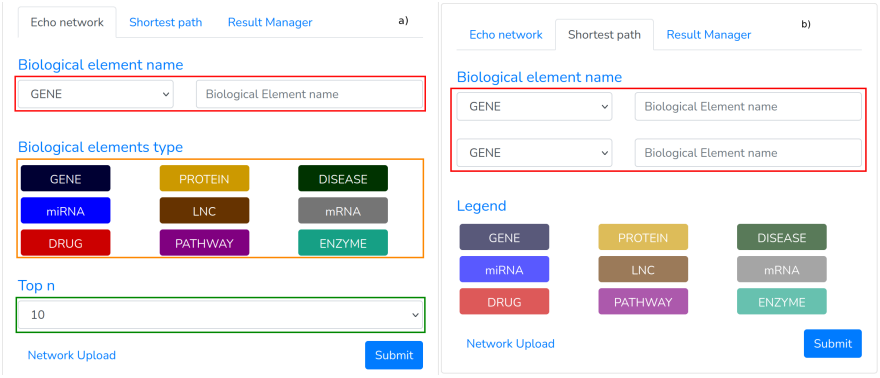


Figure 7: Echo Network and Shortest path panel

The *Graph panel* is used to plot (by using the CytoscapeJS library [124]) the sub-graph (Figure 6) returned for a user-submitted query. The edges of such a sub-graph are interactive. Thus, if a user clicks on them, a relationship window (Figure 8) containing the following data is shown:

- A table 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 reports the literature evidence (1 if the relationship is reported in at least one of the literature DBs, $\emptyset$ otherwise).
- A navigation panel with three different options. The first two (named "Element 1 Wikipedia Pages" and "Element 2 Wikipedia Pages") show several links between Wikipedia pages and source or destination nodes, respectively. The last one (named PubMed articles) shows all the links to PM articles containing the selected relationship.

3.3 EVALUATION

We analyzed the reliability of BioTagME on two case studies. The first one aimed at determining its prediction quality by evaluating its ability to extract "Basigin" relationships. We evaluated our results by comparing the performance of BioTagME against a set of edges extracted from STRING [16]. The second case study focused on the construction of a "blood coagulation" network, which is then compared against a network obtained from the links available in the ontologies employed in BioTagME (i.e., this is a sort of "gold network" based on the literature).

We chose this approach because of two main reasons:

- Some datasets for biomedical NER exist, but none covers a wide variety of bio-entity types, and none covers the full EL pipeline, as we wish to evaluate.
- Evaluating edges implicitly evaluates also the quality of nodes. Being unable to retrieve either the source or target node of an edge counts as a missed one.

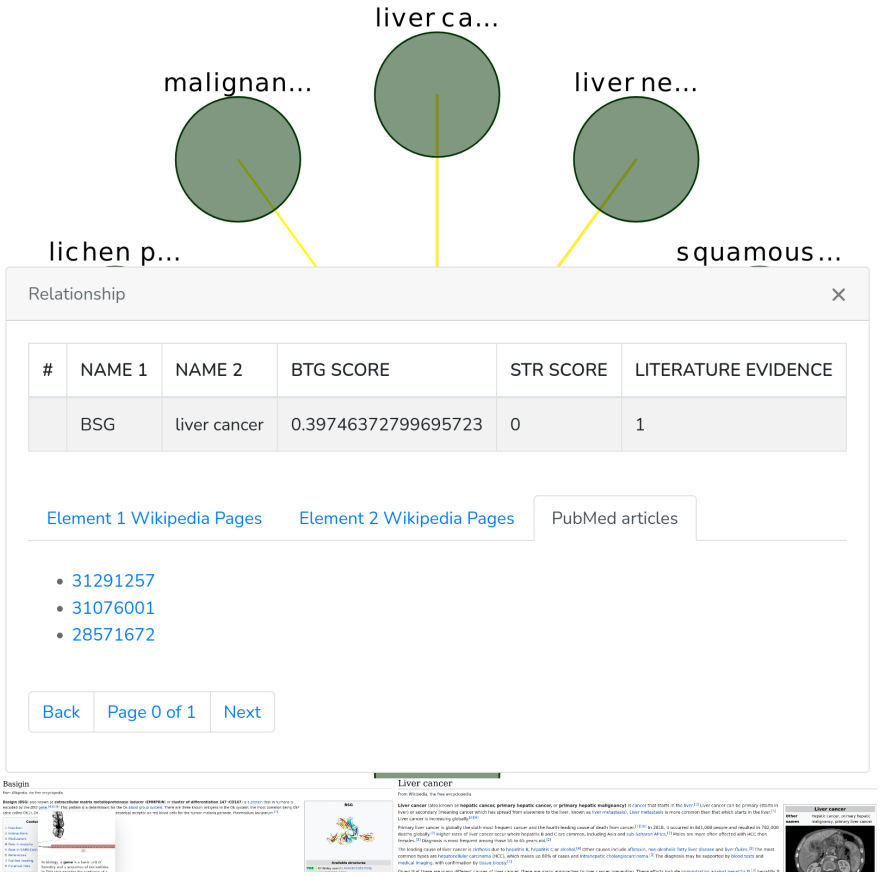


Figure 8: Relationship window

3.3.1 *Case study 1*

Many tools and computational models [125] rely on existing network DBs, such as KEGG [126] and Reactome [121–123]. However, despite the enormous amount of available data, these DBs are still incomplete and therefore have partial information.

In this first case study, we have chosen *Basigin* (BSG), also known as CD147 or EMMPRIN, as a starting point to construct a protein-protein functional network. This gene represents an example of a biomedical element that should be supplemented to the KEGG network since it is not currently described in their pathways. BSG is a transmembrane glycoprotein of the immunoglobulin superfamily, expressed in many tissues and cells. It is known to participate in several highly relevant biological and clinical processes. Furthermore, BSG is a crucial molecule in the pathogenesis of several human diseases [127].

Missing a crucial gene within a biomedical network can compromise the effort of scientists to understand certain molecular mechanisms. However, the most reliable approach to date remains manual curation through careful and time-consuming literature analysis. On the other hand, a manually constructed network provides partial information due to the limited number of articles that a scientist can read.

Our case study tackles this issue by providing a practical example of how BioTagME can create valuable networks by analyzing a large set of PM abstracts.

Figure 9 shows a snapshot containing the top 30 nodes of the network built by BioTagME.

Figure 9: Basigin-Proteins interaction network

BioTagME inferred 426 true positive relations and 38 false negatives. In addition, such a network has been compared with STRING to assess sensitivity and specificity. Qualitatively, this network includes most of the interconnections mentioned in STRING, thus providing a reliable and comprehensive overview of the molecular function of *Basigin*. Quantitatively, BioTagME achieved a sensitivity of 91.8% and a specificity of 94.8%.

3.3.2 *Case study 2*

The second case study aimed at building a general functional network related to the "blood coagulation pathway". Blood coagulation is a complex chain process involving a series of stimulus responses in conjunction with coagulation factors and enzymes, whose intent is to stop blood fluxes when a vascular tissue injury occurs [128].

To evaluate the quality of the network built by BioTagME, shown in Figure 6, we compare it again against a "literature network" generated by data and relationships present in the ontologies.

BioTagME was able to infer 54 true positive and 23 false negative relations. Quantitatively, it achieved a sensitivity of 70.12% and a specificity of 96.43%. Indeed, it

could predict the relation between blood coagulation and PROS1, a gene that plays a crucial role in the mechanism of PtdSer exposure during immunity and blood coagulation [129]. Moreover, BioTagME was able to predict the relationship between blood coagulation and the Thrombin and Plasmin enzyme. The role of the Thrombin enzyme is to catalyze the initiation and propagation phases of blood coagulation, and it converts soluble Fibrinogen to insoluble fibrin [130].

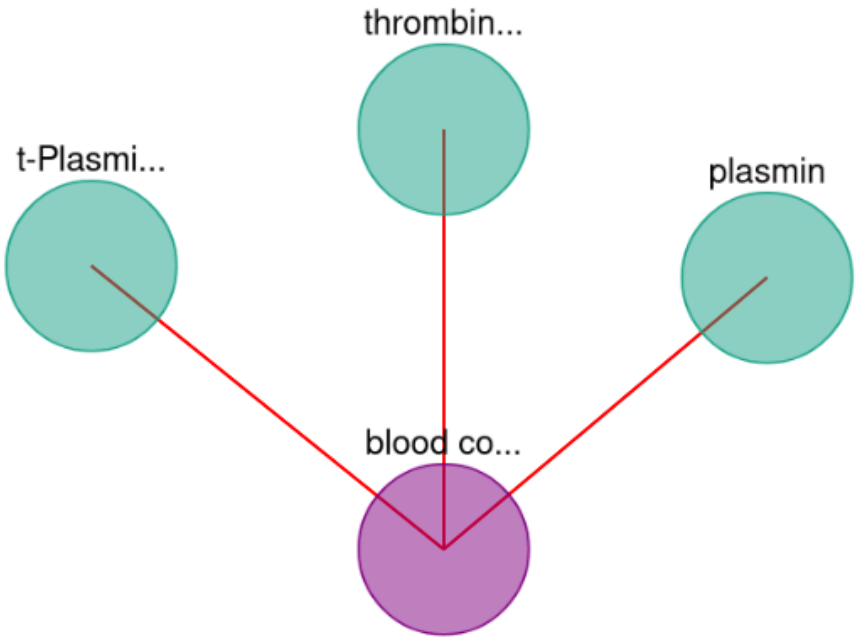


Figure 10: Blood coagulation and enzymes interaction

4

ONTOTAGME

OntoTagME is a biomedical Entity Linker. We developed this tool to obtain biomedical annotations at a low computational cost. OntoTagME is built on top of TagME, an already well-established general-purpose Entity Linker, widely used both in academia and industry. TagME uses Wikipedia as a back-end KG to identify mentions in the text that refer to real-world entities. More detailed information on TagME is provided in Section 2.3.1, while in Section 4.1 we discuss the why (and the why not) TagME is a satisfactory approach for biomedical annotations. Being built on top of TagME, we designed a method for extracting a biomedical view starting from a general-purpose KG. More details are shown in Section 4.3.

It is important to keep in mind that OntoTagME was always envisioned as a *building block* for NetME, a tool for on-the-fly network constructions, and this influenced some of its design principles. A preliminary version (i.e., 1.0) of OntoTagME was released in 2022 and is described in Section 4.2, then in 2023 we designed an improved version (i.e., 2.0) that is described in Section 4.4.

4.1 WHY (OR WHY NOT) TAGME

As discussed in Section 2.3, the SOTA has a wide range of choices for the task of EL. In this section, we argue why TagME is a great choice for this application, along with its pros and cons.

TagME is one of the most widespread entity linkers in use. The main reasons are the effectiveness when querying short texts, the accessibility of the publicly available REST API, and the code repository for local installations. TagME, being one of the first studies on EL, is a relatively old (i.e., 2012) technology.

Other entity linkers (e.g., REL, E2E neural linker, SWAT) outperform TagME in terms of raw F_1 measure, however TagME has a set of pros that make it the perfect pick for us.

- *Ease of adaptability*: TagME does not employ any ML model, which means that its results solely rely on the dataset that is fed to it as a KB. Changing the back-end KGs immediately adapts the results. This was a perfect fit for us.
- *Fast annotation process*: speed of annotation is surely a very welcome trait. Since OntoTagME will be used for on-the-fly network construction, the speed of annotation is surely a positive factor.
- *Resource efficiency*: in addition to not requiring a GPU (which is required by every other SOTA entity linker), TagME requires a reasonable amount of memory. More specifically, TagME requires around 60 GB of memory for indexing and running on the entire Wikipedia corpus (which is roughly the same amount required by other annotators), however, TagME can run using only 5 GB of RAM when using the biomedical ontologies mentioned in Section 4.3.2.

- *Cost and Energy efficiency*: it should not come as a surprise that a computational-efficient system is also an energy-efficient one. Being able to remove the GPU requirement, also reduces the power needed to keep the system up.
- *Code Availability*: this point is an obvious pro. However, most of the SOTA entity linkers also have a public code repository. Specifically, REL[1] and E2E Neural Entity Linker[2] have a public repo.
- *Multi-Language*: TagME is, as far as we know, the only entity linker with multi-lingual capabilities.

However, TagME is not a perfect system. More specifically the cons are:

- *Sub-optimal performances*: Purely measuring the results in F_1 measure on the general-purpose case, TagME is sub-optimal with respect to the SOTA.
- *Relies on the graph structure*: It is strange to classify this point as a con because TagME's strength is exactly this reliance on the graph. In our case, however, knowledge is often provided in a more "*tabular form*". Most of the bio-databases and ontologies simply provide a list of the biomedical entities, without proper relation between each other. This means that, for TagME to shine, we must be able to build a good enough set of relationships between entities.
- *Built for short text*: TagME's design is strongly influenced by the choice of working best on short snippets of text. This works well for titles and abstracts, but we cannot expect exceptional performance when moving to full-texts. To reduce the impact of this problem, we split full-texts into paragraphs and annotate them separately.

4.2 ONTOTAGME 1.0

Before the current release of OntoTagME, we presented a first working version in [10]. This version, which we refer to as OntoTagME 1.0, was very different from the live one (which we will refer to as OntoTagME 2.0 for the remainder of this thesis). OntoTagME 1.0 was envisioned as a building block for NetME 1.0 since the step of entity extraction was crucial for building biomedical networks of high quality.

Its original design features a careful mix of biological ontologies and databases. We define a set of crucial entities and find the relative ontology that is the best fit for extracting them. Then we carefully merge these ontologies in order to ensure, to the best of our abilities, that no duplicates from different ontologies clash. More details on this phase are described in Section 4.2.

A glimpse at the full architecture of OntoTagME 1.0, can be seen in Figure 11. As shown, ontologies and bio-databases are downloadable in different formats. For each ontology, we use some custom Python parsers that adapt their content to one format only. Then we merge everything in three CSV files, which are then transformed into files that can be fed directly to TagME. The TagME version is a slightly modified one to better work with a biomedical language.

1 https://github.com/informagi/REL
2 https://github.com/dalab/end2end_neural_el

All the images shown in this chapter are taken from our work published in Applied Network Science [10].

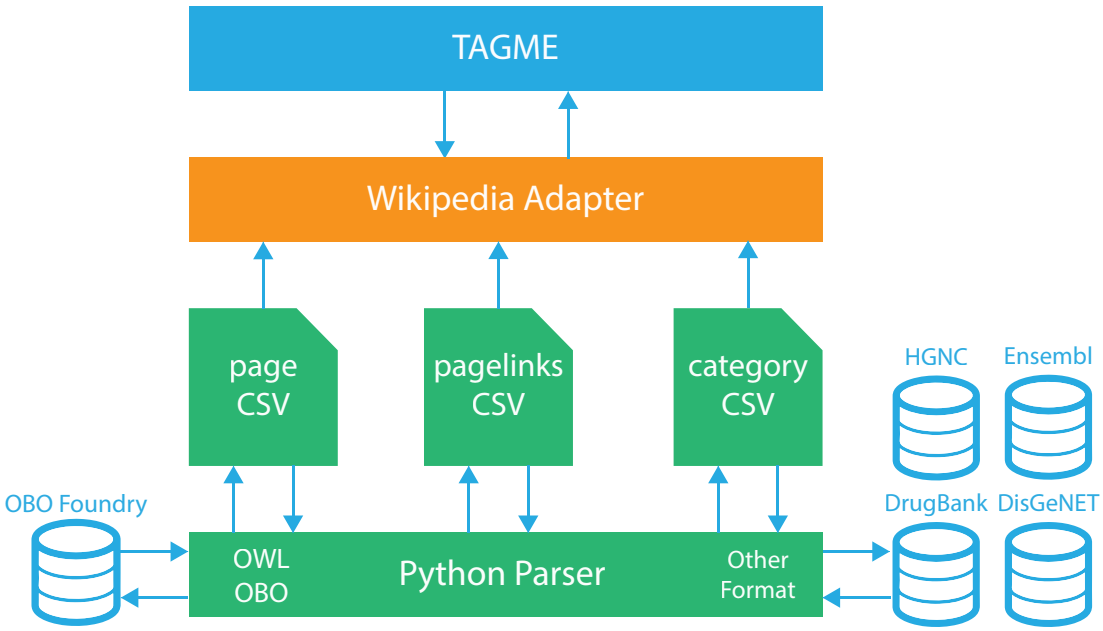


Figure 11: OntoTagME pipeline architecture

The first goal of OntoTagME is to build a biomedical KB starting from a set of ontologies (explained in Section 2.5.1). We use the following set of ontologies:

- Gene Ontology [11]
- Disease Ontology [4]
- Pathway Ontology [12]
- BRENDA tissue ontology [94]
- Protein Ontology [93]
- Anatomical Entity Ontology [95]
- Phenotype And Trait Ontology [96]
- Cell Ontology [97]
- Cell Line Ontology [98]
- v5.1 release of DrugBank [6] for drugs.
- Disgenet [17] for diseases
- the 2022 release of HGNC [73] for the human genome.
- the 2022 release of ENSEMBL [89] for vertebrate genomes, downloaded through their FTP service.
- CIViC [90] for clinical interpretation of variants in cancer

- PharmGKB [91] for pharmacogenomics

All data in Ensembl are used in combination with those coming from HGNC to detect gene names and symbols within a text. The data relative to the number of nodes and relationships extracted from each mentioned ontology are listed in Table 4.

Ontology	N. Nodes	N. Edges
Gene Ontology	43 917	142 086
Disease Ontology	10 862	29 938
Pathway Ontology	2 619	6 210
BRENDA tissue ontology	6 515	9 378
Protein Ontology	326 811	846 366
Anatomical Entity Ontology	248	523
Phenotype And Trait Ontology	4 610	413 027
Cell Ontology	10 809	34 410
Cell Line Ontology	44 712	91 966

Table 4: Number of nodes and edges per ontology

After having downloaded all the ontologies, we run a short pipeline to integrate all their information.

1. The first step is adapting all the ontologies to a set of files that more closely resemble the structure of the files used by Wikipedia. This is achieved by creating short Python scripts that are ontology-specific. The output of this step is to create one folder per ontology, containing three files: (i) *page.csv*, contains all the entities and aliases, (ii) *category.csv*, contains the categories of the various entities, and (iii) *pagelinks.csv*, contains all the links between entities.

2. The second step is to merge all those files into just three files, which are *final_page.csv, final_category.csv, and final_pagelinks.csv*. The categories, in this case, refer to the macro-one, and are defined for each entity. For example, if an entity is extracted from the HGNC database, we put *"gene"* as the only category.

3. In the third step, each row of the *final_page.csv* file is converted into an XML file containing a unique ID generated by our system (increasing integer), the name (title), type (category), and the description (page's body) of the biomedical element. TagME does not accept files in CSV format, so we need to adapt the files to a set of XML and SQL files tailored for it. This step is the one where the resolution of conflicting entities has to be managed. The process is similar to the one used by BioTagME, described in Section 3.1.3. In particular, we use external database metadata, synonyms lists, and references toward other external database lists to group together entities from different data sources that are deemed as equal. This CSV to XML transformation is needed because, of course, each page can have a set of *linked pages* (which correspond to our pagelinks, e.g., diseases DOID:0002116 and DOID:10124 are linked with an edge labeled *"is a"*)

and a set of aliases (also called redirects in Wikipedia terms, which TagME borrows the language from, e.g., gene BSG is also called CD147 and Basigin).

Our process generates a tuple containing the relationship for each element in the set of the pagelinks, and a tuple with the aliases for each element belonging to the set of aliases. These tuples are then stored in the SQL files "wiki-latest-pagelinks" and "wiki-latest-redirect", respectively.

Finally, the SQL and XML files are fed to TagME, and they get indexed and then exploited as a KB. The complete generated OntoTagME network contains about 331K entities, about 700K aliases, and about 4 million edges.

4.3 BUILDING A BETTER KNOWLEDGE BASE

In an attempt to improve the quality of the annotations, and considering the high volume and quality of available biomedical data, we aim at building a biomedical KG starting for a large and widely used general-purpose ontology. This is done to:

- reduce the amount of noise generated during the annotation phase;
- remove potential duplicates given by an imperfect merging process of ontologies;
- allow us to drastically increase the sheer size of the ontology in terms of the number of modeled entities;
- increase the number of *cross-category* edges, as ontologies are usually specialized on a single category (e.g., gene, disease, drug).

Some key factors must be kept in mind when dealing with biomedical entities. This is due to the fact that biomedical knowledge has been historically "developed" and researched widely around the globe, by many different people from many different countries. This causes naming conventions to vary, and even today standards often fail to stick. Some key factors to consider are:

- *Non-Standard Naming*: Naming is not fully standardized for many biomedical entity types. Some entities, like genes, suffer from this problem extensively.
- *specie-specialized ontologies*: For some specific cases, like the ones of genes, different consortiums handle the naming of said bio-entity for different species. For example, in the case of genes, HGNC handles the human genome, MGI handles mouse genomics and many more.
- *Encyclopedic Focus*: Biology is a field that is based on interactions. This means that networks and graphs are the preferred data structures of choice to keep information. This, unfortunately, does not correspond to the reality of biodatabases and ontologies, where information is usually kept in tabular form. Relations are handled by their own specialized ontology (e.g., DisGeNET).

We present now the notion of *macro-category*, which will be used extensively for the remainder of this chapter and in the next Chapter 5. We call macro-category a category that identifies the main types of bioentities we wish to model. They are gene, cell

type, gene variant, cell, biological pathway, symptom, cellular component, species, compound, physiological condition, chemical entity, disease, enzyme, drug, and protein. Instead, we call "categories" the ones that identify a group of biomedical entities whose categorization is either too specific or a subset of one of the macro-categories we defined before. For example, the category "tumor suppressor gene" is a subclass of our macro-category "gene". Nevertheless, categories add a lot of data that can be extremely interesting to query for users.

The two main data sources we considered for addressing the task of building a better biomedical KG are Wikipedia and Wikidata. We attempted to build a full biomedical KG from both of them, as both have a high volume of manually curated pages. Of course, being subject to manual curation is a double-edged sword, and it can cause some potential abuses and mistakes, like in the well-known case of the Scottish Wikipedia [131]. However, in our opinion, the large volume of data, the powerful categorization capabilities, and the quality of the links on Wikipedia strongly outweigh the cons. Therefore, in Section 4.3.1, we describe the full process we followed for building a biomedical view of Wikipedia, along with design decisions, pros and cons, and why we then decided to drop this route. In Section 4.3.2, we describe the same process, but over Wikidata, and we discuss why we decided to stick with this option for OntoTagME, discarding the previous one on Wikipedia.

4.3.1 *Biomedical Wikipedia*

In order to extract a *topic-specific* ontology from Wikipedia, the process we envision is quite intricate. This is mostly due to the fact that its item categorization is strongly inconsistent, although very powerful. We observe, however, that the process we follow is sufficiently general that it can be easily applied to other fields.

We will base our Wikipedia filtering on the categories associated with a Wikipedia page. Each page is assigned a set of categories, which are intended to group together pages on similar subjects. They are displayed at the very end of a Wikipedia page, and they can range from very general (e.g., "Protein") to very specific (e.g., "Genes on human chromosome 17").

Unfortunately, categories are far from standard in Wikipedia. Say we pick the *"lung cancer"* page[3] as an example. The categories, at the time of writing this thesis, are *"Lung cancer, Health effects of tobacco, Types of cancer"*. The category *"disease"*, which we would expect to be present for said entity, is not even present in the set. Now let's see the Wikipedia page of another type of cancer, for example the *"T-cell lymphoma"*[4]. In this case, along with other categories, we see *"Diseases and disorders, Cancer, Lymphoma"*. This shows that some diseases are indeed labeled with the category *"disease"*, but others are not.

Wikipedia also has a concept of "sub-categorization", which means that a category and its sub-categories are arranged in a *best-effort* DAG, where two categories share a child-parent behavior if the child category is deemed to be an over-specification or subset of the parent one. This effectively means that ideally, cycles should not be

3 https://en.wikipedia.org/wiki/Lung_cancer
4 https://en.wikipedia.org/wiki/T-cell_lymphoma

present in this DAG, but unfortunately, some cycles have been shown[5] to exist. To visit this DAG, then cycles must be taken into account.

Categories for a page are often misleading. As an example, we show the Wikipedia page about the gene *"BRCA1"*[6]. Here, the 4 proposed categories are *"Genes on human chromosome 17, Breast cancer, Tumor markers, Tumor suppressor genes"*. At first glance, we immediately see that the page is not labeled as a gene, but as humans, we are able to understand that it is indeed a gene by the fact that there is a category that locates it in chromosome 17. We are also able to see that a category is *"breast cancer"*. This page makes sense, as this specific gene is a marker for breast cancer, but it's extremely misleading to put this as one category. This shows that it is not uncommon to find categories of *related bio-elements* in the categories of other elements, possibly even of different bio-entities. This makes the job of deciding the macro-category of a Wikipedia page harder and harder.

Wikipedia also offers a Biology portal[7]. This is a page that provides some broad explanation on the field, a small set of daily pages for users to browse, and then some information which may interest us more. In particular, it shows all the set of sub-categories of the main category "Biology" (at the time of writing this thesis, they are 35). This portal is shown in Figure 12. By clicking on the arrows, we are also able to see all the sub-categories of the selected one. Immediately at first glance, we are able to spot a set of categories that are not important to us, since we focus on an Entity Linker that models biomedical entities. For example, we can rule out "Biology Education" and "Biology and culture".

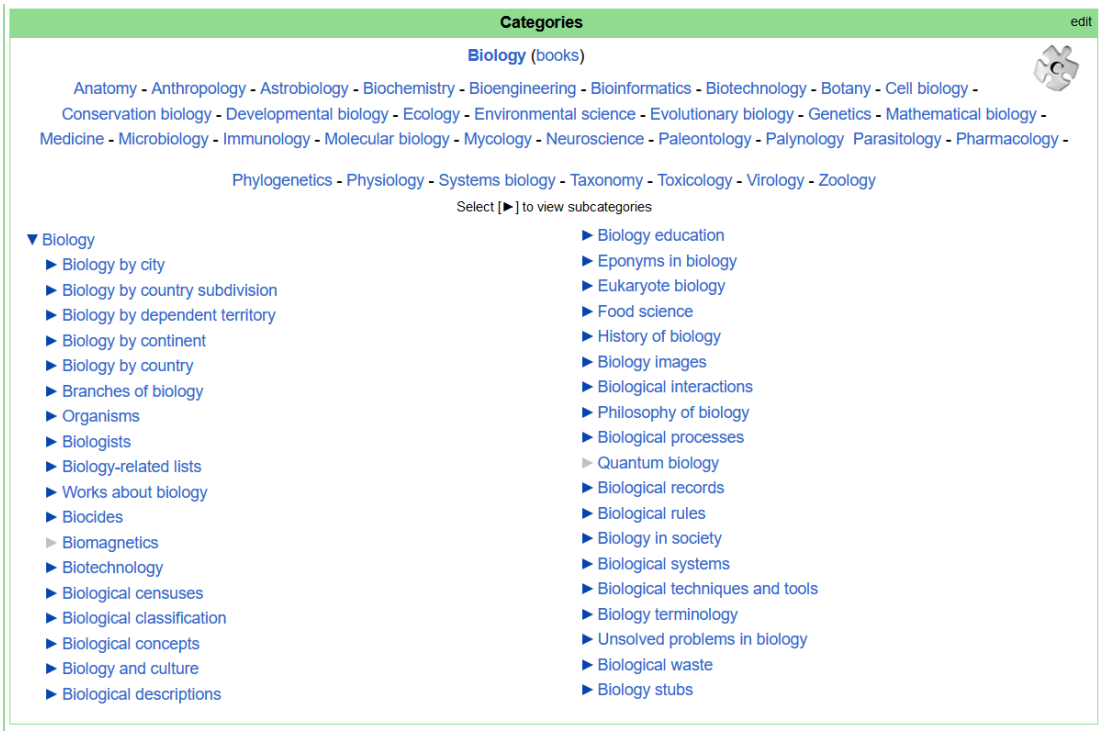


Figure 12: Wikipedia categories in the biological portal

5 https://en.wikipedia.org/wiki/Wikipedia:Dump_reports/Category_cycles
6 https://en.wikipedia.org/wiki/BRCA1
7 https://en.wikipedia.org/wiki/Portal:Biology

It must be noted that following the *Biology* category tree will not lead to satisfactory results, since:

- It will also find pages that are not relevant to biology. For example, following this path in the category DAG:

  Biology → *Biology and Culture* → *Fiction about creatures*

  we reach page *"The Jiggler"*, which is related to an episode of the TV series *"Adventure Time"*.

- The visit will not attach any *macro-category* to the visited pages (genes, proteins, diseases...). This is due to the inconsistent categorization which does not ensure that every gene is labeled as a gene.

- Cycles of varying lengths are present in the category DAG, so it could lead to loops if visited without care.

Following that, we found three methods for extracting a biomedical dump from Wikipedia, which will be described, along with their pros and cons in the next paragraphs. The Wikipedia dump files were downloaded through the WikiDumps page[8].

METHOD 1. For our first attempt, shown in Figure 13, we decided to explore the Biology sub-tree up to layer n of Wikipedia, where we manually picked n to minimize the noise.

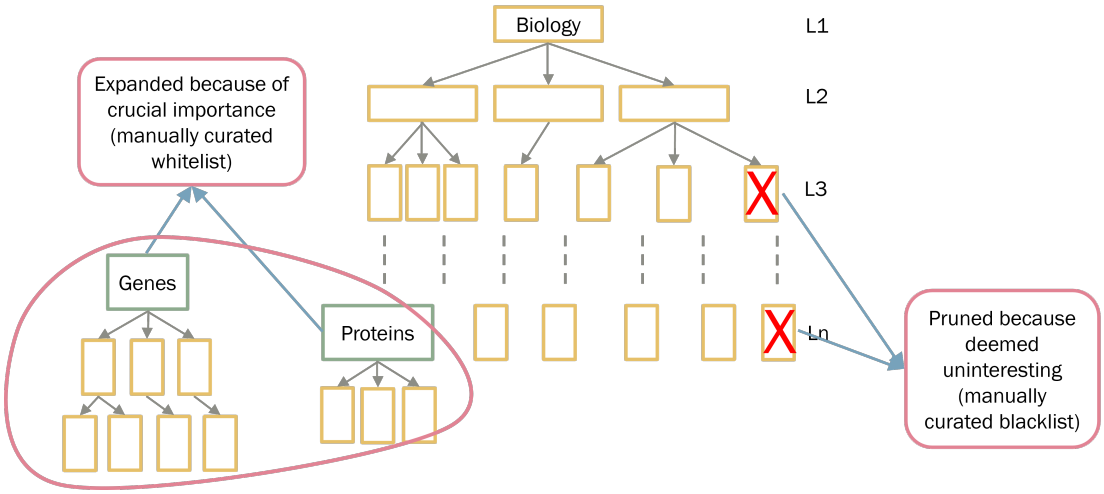


Figure 13: Wikipedia dump, first attempt schema

Decreasing n results in fewer pages and less noise. Additionally, to ensure that the most important sub-trees are explored, we also explore the sub-trees of a manually curated set of categories (e.g., genes and proteins, human diseases). Those trees are expanded up to layer m, where m is manually provided by the user, not necessarily equal to n.

We also allow the user to provide two blacklists. The first one contains a set of categories that are deemed uninteresting. This means that any page labeled with

8 https://dumps.wikimedia.org/

said category and any sub-category of it is immediately skipped. The second one is for some pages that are blocked because they are considered to consistently provide significant noise in the annotation phase.

Unfortunately, with this method, no matter the cure we put in curating the lists and parameters, we got these recurring problems:

- noisy annotations, mostly coming in the form of non-biomedical mentions;
- no macro-categories, which is a crucial feature for our future applications in NetME (explained in Chapter 5).

METHOD 2. Figure 14 shows the schema we designed for the second attempt.

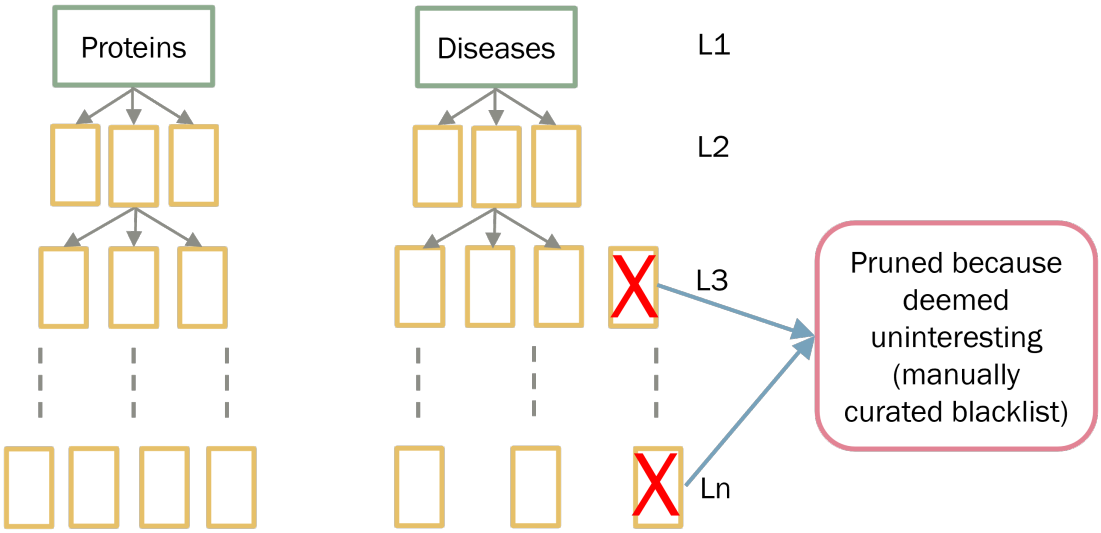


Figure 14: Wikipedia dump, second attempt schema

Here, we dropped completely the Biology macro-category, both for being too broad to obtain satisfactory results, and too filled with noise from other fields, like culture and trivia. In this second attempt, we limit ourselves to a restricted set of categories, each one associated with an integer number n_i, which corresponds to the number of levels to explore in its sub-tree. Additionally, we provide a black list containing all the pages to discard and all the categories to prune, similarly as in the previous attempt.

This second attempt results in significantly less noisy annotations, however, it still suffers from the same lack of *macro-categories* of method 1.

METHOD 3. Figure 15 shows the schema of the third, and best-performing, attempt. We envisioned this method as a way to minimize noise while adding the possibility of annotating also macro-categories, which are extremely useful for modeling biomedical networks. The core idea comes from the fact that we focus on a set of categories that we deem to be representative of a macro-category. For example, let's focus on diseases. As we discussed before, diseases are not always labeled with the category "diseases and disorders". So, we found a set of categories that have a prevalence of

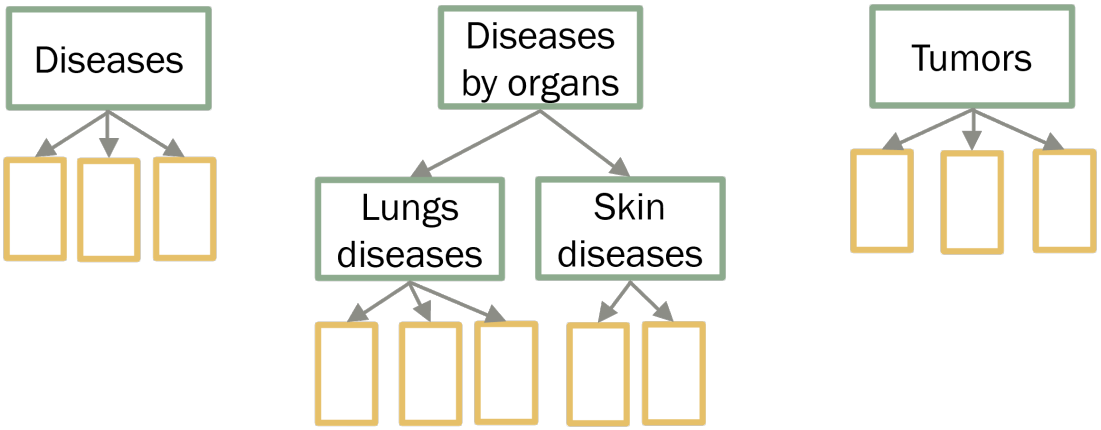


Figure 15: Wikipedia dump, third attempt schema

disease pages in Wikipedia, and we manually explored their sub-trees to understand empirically which is the best number of layers to explore from them. For the case of diseases, we found a set of 35 categories, and each of them is associated with a number that expresses the number of sub-categories to explore, where $\emptyset$ means only the pages (e.g., $\langle$*Human diseases and disorders*, 0$\rangle$, $\langle$*Drug-induced diseases*, 1$\rangle$). After, we extracted each page within any of the categories we manually added to the set, and their children of variable depth as discussed. For each of the entities we find, we also keep track of their depth with respect to each of the categories we manually chose. This is to associate each entity with a macro category. For example, in the case of gene BRCA1[9], we reach this page from many paths. The four categories associated are:

- *Genes on human chromosome 17*: This category is reached at depth 1 for macro category *"gene"* following path: *Human genes* $\rightarrow$ *Genes on human chromosome 17*.
- *Breast cancer*: This category is reached at depth $\emptyset$ for macro-category *"disease"*.
- *Tumor Marker*: This category is reached at depth $\emptyset$ for macro-category *"gene"*.
- *Tumor Suppressor genes*: This category is reached at depth $\emptyset$ for macro-category *"gene"*.

In the end, we reach page BRCA1 three times from categories about genes (at depths 0, 0, 1) and once from categories about diseases. This allows us to assume that BRCA1 is indeed a *gene*, and we are able to assign to it that macro-category.

This solution is surely the most beneficial of the three we discussed above, as it obtains the best results noise-wise while retaining the ability to assign macro-categories. Unfortunately, this comes at a cost of a lot of manual work for us to explore a very messy category DAG.

In the following Section 4.3.2 we discuss an approach that exploits a different KB, Wikidata, and obtains better results in an easier manner.

9 https://en.wikipedia.org/wiki/BRCA1

4.3.2 *Biomedical Wikidata*

Wikidata is a free and open KB that can be read and edited by both humans and machines. Its focus on being a KB makes it a perfect fit for programmatic access. The list of potential aliases per entity is easily retrievable and located at the top of the page, right under the page title. Wikidata then provides a tabular view, where each entry of the table is characterized by a property. Each property is uniquely identified, clearly documented, and can be easily browsed[10].

For example, some useful properties of genes include:

- *P1057 - chromosome*: specifies the chromosome where the gene can be located.
- *P351 - Entrez gene ID*: the Entrez ID of said gene, which is a manually curated ontology, used by many researchers as a standard for naming.
- *P703 - found in taxon*: specifies the species associated with that gene (e.g., human, mice, mammals).
- *P1916 - gene substitution association with*: provides links from that gene to diseases (aka a GDA).
- *P644 and P645 - genomic start and genomic end respectively*: provides the coordinates of the gene in its chromosome.

The categorization of the pages retains the power of Wikipedia's categorization (and its depth of information) but it is better organized for macro-categories.

For extracting all this information and building the necessary files for OntoTagME, we forked the repository *simple-wikidata-db*[11]. This repository provides a set of Python scripts to extract all the required info for a specialized ontology from the full Wikidata dump.

The way this works is the following:

- The user provides a dictionary containing as keys the Wikidata identifiers of the class of interest, and as values user-defined macro-categories. For example, we might put as a key "Q113681859", which corresponds to the Wikidata page *"type of chemical entity"*. As a value, we add "chemical entity" as a macro-category. This is useful in order to reduce the amount of main categories the system relies on. This dictionary is needed to tell the script which is the correct subset of pages to extract.
- The user provides a mapping to external IDs. This step is needed to integrate other services that use different IDs with respect to Wikidata. For example, PubTator provides annotations for genes using the NCBI Gene identifier. Wikidata, for some of the genes, provides also the NCBI gene ID under property P351. In this mapping, we can specify those properties used by different systems, so that a new file "external_ids.csv" can be built for future use during integration with other services.

10 https://www.wikidata.org/wiki/Wikidata:List_of_properties
11 https://github.com/LorenzoBellomo/simple-wikidata-db

- The user provides a black list, containing a set of pages and aliases that must be dropped from the entire processing pipeline.

After launching the full script under those circumstances, the code prepares all the files required for proper indexing. Then, by calling the *fetch_bio_entities.py* script, the code builds a set of files that are more digestible by TagME. The files are:

- *page.csv*: contains all the extracted pages, along with their aliases.
- *category.csv*: contains all the categories for each page extracted in step one. The categories are extracted from 3 Wikidata properties, that are P31 ("instance of"), P279 ("subclass of"), and P361 ("part of").
- *pagelinks.csv*: this file is the one required to make *a graph out of Wikidata*. To build this graph, we take each Wikidata page and check all of its properties. If a property points to another Wikidata page, we check whether this page is present in our subset of relevant pages. When this happens, we add a link between those two.
- *external_ids.csv*: This file is only useful for integration with external services that use a different set of identifiers for the various entities.

Table 5 shows some numbers regarding the biomedical view of Wikidata we have built. Table 6, instead, shows some data regarding the distribution of pages per macro-category.

Description	Number
Pages	3 545 733
Aliases	4 635 736
Categories	17 279 837
Avg. cats per page	4.87
External IDs	1 670 352
Graph Edges	8 012 196

Table 5: Size of the Bio-KG extracted from Wikidata

4.4 ONTOTAGME 2.0

The goal of the most recent version of OntoTagME was to improve the performance of the EL process.

To achieve this goal, we decided to shift the focus from using a set of well-defined, but fully separated ontologies, to using one comprehensive general-purpose ontology, adapted to the biomedical field of interest. The KG of choice was Wikidata, and the detailed description of the reasoning behind this choice and the design process has been fully illustrated in Section 4.3.2. Essentially, the main reasons for this design are:

- *Unique identity*: Working with bio-ontologies means carrying a lot of different identifiers, as each ontology uses its own. In the new OntoTagME, we can just

Macro-Cat.	Number
gene	4 406 910
cell type	10 245
gene variant	35
cell	1 053
biological pathway	7 178
symptom	4 454
cellular component	14 039
species	136
compound	2 700 398
physiological condition	263
chemical entity	10 715
disease	98 236
enzyme	49 880
drug	23 569
protein	3 385 962

Table 6: Size per Macro-Category of the bio-KB

use Wikidata IDs, that possibly include additional external IDs, useful for linking our items with external ontologies.

- *Large categorization capabilities*: Categories in ontologies are usually very general (e.g., gene, disease). Wikidata offers a set of extremely useful information, which can allow the user to perform interesting queries (for example, which genes are related to at least one tumor marker).
- *Cross category links*: Most bio-ontologies offer a tabular view, with several ontologies dedicated to crucial relations (e.g., GDAs). In the case of Wikidata, entities of different macro-categories can be related. These links are also used to perform a better disambiguation process by the underlying TagME installation.

The second upgrade present in OntoTagME 2.0 is a dedicated Mongo-DB[12] installation. This has been added to improve the support for future uses (e.g., the full biomedical KG we envisioned as a future work). More specifically, OntoTagME requires performing many lookups to a set of dictionaries for enriching the returned information. For example, this happens for both the categories file and the files of external ontology IDs (used for the integration with Pubtator). Since the most common operation to be performed is the lookup by ID, we found non-SQL DBs to be the perfect fit for this. We decided to deploy MongoDB for this purpose so that all OntoTagME applications can simply query the mongo-DB installation to retrieve categories and external IDs.

12 https://www.mongodb.com

The third upgrade is the added capability of OntoTagME to be integrated with external tools. More specifically, we added support for PubTator, a biomedical NER tool, whose detailed description and integration will follow.

In order to improve the quality of the returned annotations, we also need to envision some techniques for noise removal. This is crucial because some bio-entities, most notably genes, are extremely hard to properly annotate, as we widely discussed, in Section 2.4.1. Therefore we added to the design of OntoTagME the following noise-reduction approaches:

- We filter out from the annotations the Wikidata pages whose title includes a very common word (such as "case", "patch", or "line", which usually come as aliases of various genes).
- We filter out from the possible mentions the ones (i.e., aliases) that match at least one of the following constraints: (i) composed of just one word, (ii) have no numerical characters, (iii) have no special characters, (iv) they are formed either by all lowercase or have only the first letter capitalized. We then stem all the remaining mentions and see if any of them match. We then take all the matches and check if their macro-category is equal. In this case, we "uniform" the annotations by adopting as name and list of categories the ones provided by OntoTagME. For example, say OntoTagME and PubTator find as a mention "cancer" and "cancers" respectively, and both of them have "disease" as a macro-category. In this case, we can change the returned data attached to all these two annotations to the ones provided by OntoTagME because their stemmed mentions are equal and their macro category is the same (disease).

Being built on top of TagME, OntoTagME provides its result in a similar format. More specifically, for each annotated mention, OntoTagME provides:

- *WID*: internal numerical identifier.
- *mention*: substring where the entity was found.
- *start_pos*: index of the starting character of the mention.
- *end_pos*: index of the ending character of the mention.
- *Word*: title of the Wikidata page denoting the entity associated with that mention.
- *categories*: list of categories associated with that Wikidata page, comma separated.

For all the other features, OntoTagME 2.0 essentially behaves exactly as OntoTagME 1.0 does, so any point not related to this changelog still stands in version 2.0.

Another important feature of OntoTagME is its great interoperability with other Bio-NER tools. In particular, we investigated the use of PubTator Central[13] [14, 15] and BERN2[14] [84], that were built to solve the task of NER in the biomedical setting, by possibly linking mentions to some external biomedical ontology.

13 REST API: https://www.ncbi.nlm.nih.gov/research/pubtator/
14 REST API: http://bern2.korea.ac.kr/

After some testing, we decided to focus on PubTator and discarded BERN2 because the BERN2 public REST API incurs a high latency. Conversely, PubTator is a very effective Bio-NER tool that performs efficient annotations of PM abstracts and PMC full-texts via deep-learning techniques, and it further categorizes them into genes/proteins, genetic variants, diseases, chemicals, cell lines, and species. Sometimes the system is also able to identify which specific bio-entity is referred to in a specific textual mention and, in these cases, PubTator enriches the data by adding an external ID to external ontologies such as:

- *Gene*: NCBI entrez gene ID (corresponding to the Wikidata property P351);
- *Species*: NCBI taxonomy ID (corresponding to the Wikidata property P685);
- *chemicals*: ChEBI ID (corresponding to the Wikidata property P683);
- *diseases, drugs and chemicals*: Mesh Descriptor ID (corresponding to the Wikidata property P486);
- *cell lines*: Cellosaurus ID (corresponding to the Wikidata property P3289);
- *cell types*: Cell Ontology ID (corresponding to the Wikidata property P7963).

To further enrich the annotations discovered by OntoTagME, we also created an *external_ids* file that holds them. However, OntoTagME could be faced with different entities for the same mentions; so in those cases, it proceeds according to the following policy:

- If PubTator and OntoTagME both find an annotation for a specific textual mention, then only OntoTagME's is kept (because this offers a wider categorization).
- If PubTator finds an annotation that OntoTagME missed, then two possible cases can arise: (i) if PubTator provides an external ID that refers to a Wikidata page, the entity takes the name of that Wikidata page and its categories are enriched with those found in Wikidata; (ii) otherwise, PubTator provides the annotation with no further enrichment at all.

4.5 EVALUATION

This section discusses the approaches we followed for evaluating OntoTagME's performance.

OntoTagME, being an entity linker, performs both the task of NER and the task of Entity Disambiguation. In biology, the task of NER is significantly more studied than the one of entity disambiguation, which results in a significant lack of datasets. Additionally, the task of entity disambiguation in biology poses a new problem concerning the general case. While in general-purpose EL most of the datasets focus on Wikipedia pages as a KB, the biomedical field is riddled with different identifiers, which makes it even harder to have a standard ID system.

Therefore, we decided to evaluate OntoTagME's performance only on the task of bio-NER. We took this decision because of the following reasons:

- *Lack of datasets for EL*: As discussed above, no datasets for biomedical EL exist, so this would require us to build one from scratch, which would take a lot of time, effort, and feedback from experts.
- *Lack of uniform identifiers*: One of the biggest problems in genes is that naming standards differ between species. Additionally, there is not yet to be a convincing standard for identifiers. The closest thing to a standard is DiseaseOntology [4].
- *Focus on human readability*: Since OntoTagME has been designed as a building block for NetME, whose focus is to build human-readable biomedical networks, it is crucial that OntoTagME can understand where the entities are, and which is their macro-category.
- *Few synonymous words*: The biomedical field does not have the same degree of synonymy as natural language. Scientists may sometimes call the same bio-entity with two or more different names, but they tend to actively avoid naming two distinct entities the same to avoid inconsistencies and misunderstandings in papers. This *soft* rule is often broken, especially when dealing with genes, as described in Section 2.4.1.

Considering all those factors, we considered a bio-NER evaluation satisfactory. The types of Named Entities considered are exactly the set of macro-categories we defined in Section 4.3, and evaluated OntoTagME on two datasets:

- *BC2GM* [86]: Gene mention dataset. We run our evaluations on this dataset because of the crucial importance of genes in the biomedical field. Additionally, genes are the entities where OntoTagME performs the worst, so with this dataset, we should be able to show how effective the integration with PubTator is. As commented several times now in this thesis (see Section 2.4.1), we stress again that the focus on genes is due because they are significantly harder to annotate than other biomedical entities. BC2GM contains 15 000 short phrases, with the mentions where genes are present; a total of 18 265 genes, which means that on average each phrase will have slightly more than 1 gene.
- *NCBI disease* [87]: Disease mention dataset. It contains 100 short phrases, with a total of 961 mentions of diseases: hence, on average, each phrase contains about 9.6 disease mentions.

We ran our experiments on the following 4 entity annotators:

- *Baseline*: a simple annotator that scans the input phrases and matches exactly the entries present in a given biomedical ontology, namely, we used DiseaseOntology [4] for the test on NCBI-Disease and HGNC [73] for the test on BM2GM.
- *OntoTagME*: The individual OntoTagME, built on the biomedical view of Wikidata (as described in Section 4.3.2).
- *PubTator*: The individual PubTator, queried by using the publicly available REST API (as described in Section 2.4.5).

- *OntoTagME + PubTator*: A combination of the previous two entity linkers, as discussed in Section 4.4.

Table 7 reports the results achieved by running those four annotators over the two above datasets.

Dataset	Bio-type	Method	precision	recall	F_1
BC2GM	Genes	Baseline	0.43	0.05	0.09
		OntoTagME	0.40	0.29	0.34
		PubTator	**0.81**	0.32	0.46
		OntoTagME + Pubtator	0.57	**0.43**	**0.49**
NCBI	Diseases	Baseline	0.65	0.30	0.41
		OntoTagME	**0.93**	0.45	0.61
		PubTator	0.85	0.49	0.62
		OntoTagMe + Pubtator	0.87	**0.60**	**0.71**

Table 7: metrics on the BC2GM and NCBI disease datasets

On the BC2GM dataset, we notice that the baseline does get to less than 10% of F_1, OntoTagME achieves an improved performance up to 34% of F_1, which is however poorer than PubTator's 46%. PubTator achieves a significant 81% Precision on genes, but it is pretty *conservative* in its detection so that the Recall is only 32%. Rather significant is the performance achieved by the combination OntoTagME +PubTator that gets the best Recall (+11% absolute with respect to PubTator) and F_1 scores (+3% absolute with respect to PubTator). All these figures are yet less than 50%, and this is likely due to the sheer amount of inconsistencies in the naming of genes. Moreover, many genes often carry aliases or have polysemic names which further complicate their detection and annotation (e.g., gene IGKV1D-39 has alias "O2", or a gene has name "displaced"[15] or "attenuated"[16]).

On the other NCBI dataset (for which diseases do not incur the issues we mentioned for genes), the F_1 performance of OntoTagME improves by 27%, thus achieving an interesting value of 61%, with a Precision of 93% and a Recall of 45%. These figures are significantly larger than the ones achieved by Baseline. The best results, F_1 and *recall* wise, are still achieved by the combination of OntoTagMe and PubTator, which further shows the strength of this combination.

Regarding time efficiency, we cannot clearly measure the speed of PubTator, since it is queried via its REST API, so our evaluation is on the individual OntoTagME that turns out to be pretty fast by taking 24.3 seconds to annotate the 15 000 short phrases present in the BC2GM dataset, and 45 minutes to perform the full-build from the Github repository through docker. The build done on the Docker image, instead, requires less than one minute.

One of the main algorithmic properties in favor of OntoTagME is to be computationally efficient. Conversely, most state-of-the-art bio-NERs and biomedical EL tools

15 https://www.wikidata.org/wiki/Q29726548
16 https://www.wikidata.org/wiki/Q29732195

require great computational power (i.e., GPUs) in order to run their underlying LLM. For example, BERN2 requires 63.5 GB of RAM and 5 GB of GPU dedicated RAM to run; on the other hand, OntoTagME requires 5 GB of RAM to work. This feature has a significant impact on the yearly cost of running these services. In fact, according to the Amazon Web Services price calculator (which we used to estimate the yearly cost of hosting these two different services), the rate of hosting a g4dn.4xlarge GPU instance is 1.41$ per hour, which corresponds to more than 12 000$ per year. Hosting instead the lowest cost machine (which is enough for running OntoTagME), requires 0.04$ per hour, which corresponds to about 350$ per year.

Given this efficiency, we deployed OntoTagME as a free-to-use REST API on a low-cost machine of the *SoBigData* infrastructure[17] and thus made it publicly available to all scientific and commercial communities.

4.6 GENERALIZING ONTOTAGME - TOPICALTAGME

OntoTagME is just one of the many possible applications of specialized TagME, which can be ideally adapted to run on virtually any field, provided that one changes consistently its back-end KB, as we did indeed with the "biomedical view" of Wikidata.

The reasons why TagME is a great fit for custom applications are:

- *Multi-Language capabilities*: TagME has been built with multiple languages in mind. The original version of TagME provides the files for English, Italian, German, and French, but it is very easy to add others and adapt the code. Of course many recent NLP models support multiple languages (including GPT models), but none of them work in a low-cost environment as TagME does.
- *Ontology Flexibility*: TagME only requires the user to specify two things: the language of the input and the set of ontologies. The required information in the ontologies is: (i) a list of pages (entities); (ii) a list of redirects (aliases); (iii) a list of associations between pages and a list of categories; (iv) a list of pagelinks, where links between pages exist when there is a close relation between
- *Extremely Fast*: TagME is able to annotate phrases in the order of milliseconds.
- *Low Resources*: TagME is able to run on a non-GPU machine. The memory size requirement depends on the size of the ontologies, and the time cost is the one of indexing. Running requirements in memory are way lower than the one discussed now. To give some references: (i) it can index the entirety of Wikipedia with 60 GB of RAM, (ii) it can run OntoTagME (ontologies are less than 2 GB in size) with 5 GB of RAM.

For this reason, we made available a Github repo[18] with the code of TopicalTagME. This repo allows users to run a TagME installation, and customize on their specific field, easily and with low memory requirements (when the KB size is not massive). The only required things are:

- *a tab-separated page file*: similar to the one described for OntoTagME, which consists of one row per page/alias, and indicates:

17 https://sobigdata.d4science.org/web/tagme/ontotagme-api
18 https://github.com/LorenzoBellomo/TopicalTagME

  - For pages: the entity name and the entity ID
  - For aliases: the alias, the entity ID, and the regular entity name

- *a tab-separated category file*: for each page, this file has an entry with the entity name, entity ID, and the categories (semicolon separated).
- *a tab-separated pagelinks file*: for each edge in the KB, the source entity name and the target entity name.

After providing TopicalTagME with all those files, and building the image with Docker, TagME will need some time to index fully (45 minutes to index the biomedical Wikidata, whose size is shown in Table 5).

5

NETME

One of the most important tasks in biomedical research is the creation of bio-entity interaction networks. This is because of the amount of information that can be fit in them. This field already has a large amount of ontologies and bio-DBs, but they are rarely up to date with the most recent research. This effectively means that information will often be authoritative, but partial, incomplete, and non-recent. This is why researchers often have to resort to scientific papers to extract the information that is most relevant to them. Since papers revolve around many different types of entities (e.g., genes, diseases, drugs), and their interactions, the most fit data structure to model this kind of knowledge is the graph. This is where researchers take information from all their sources (research papers, ontologies, bio-DBs, experiments, and their background), and build an interaction network between all the relevant elements. Nodes are biomedical entities, categorized by type (genes, diseases, drugs...), while edges are labeled (e.g., *"upregulates"*, *"is expressed"*, *"is associated"*...) and directed.

This network is then usually used as a starting point by researchers. For example, the analysis of those graphs can reveal patterns in *gene-gene interactions*, *gene-disease associations*, or even interactions that might suggest studies on *drug re-purposing*. The crucial point to gauge is that those networks are extremely useful, but very hard to synthesize from scratch due to the sheer size of the ever-growing scientific work currently available.

Say a researcher is set on catching up with the SOTA on a specific gene. The first step, for most researchers, is to query PM for said gene. At the time of writing a thesis, querying PM for gene *"BSG"* provides 2 244 results. All of this is before checking all the aliases for gene BSG (checking on the BSG Wikidata page[1], we have as aliases 5F7, CD147, EMMPRIN, M6, OK, TCSF, Basigin, EMPRIN, SLC7A11, HAb18G). Additionally, there are unpublished papers or pre-prints, which can be found in alternative scientific paper hubs like BioRxiv.

Of course, not every single one of those research papers must be checked by the researcher to catch up with the SOTA on that gene, but the sheer number of results can indeed make it a daunting task.

The goal of NetME was to synthesize quality networks in order to reduce the workload on biomedical researchers. Networks are built on the fly, and prompted by a query formulated by users, which may be textual (e.g., the query "BSG" builds the network on the most relevant articles for that gene), or by PM IDs (the user can specify the set of biomedical abstracts or full-texts that NetME must use to build a network). The on-the-fly approach ensures that networks will always be synthesized with the most up-to-date information possible, which is crucial in an environment like the biomedical one, where the daily publish rate of new papers is in the order of thousands [1].

1 https://www.wikidata.org/wiki/Q14911959

NetME was published in Applied Network Science [10] in 2022 (this version will, from now on, be referred to as NetME 1.0, see Section 5.2); since then, it has been significantly improved to a new version, named NetME 2.0 [13], submitted for publication at Oxford BioInformatics, whose details are provided in Section 5.4, and that comes with a web application[2] that is publicly available. Sections 5.3 and 5.5 report an extensive experimental evaluation on both versions of NetME, and their comparison with respect to known biomedical KGs.

5.1 FUNCTIONALITIES AND GUI

In order to show the full set of functionalities offered by NetME, we will comment on them with reference to its GUI, and explain the relevant features of its back-end. We will also highlight the ones that are exclusive to version 2.0.

Section 5.1.1 explains the screen used to make queries to the system, while Section 5.1.2 explains the page showing the resulting network and all the functionalities provided to inspect it.

5.1.1 *Query GUI*

Figure 16 shows a screenshot of a portion of the query panel GUI of NetME. In this panel, we can see the three types of query that can be issued to NetME.

IN PM AND PMC QUERIES, NetME allows the user to add a query in the same format as PM and PMC allow. In the text field, we can range from very simple queries (e.g., "PTEN") to more complex ones (e.g., "(BSG[Title/Abstract]) AND (BRAF[Title])"), and they can be easily built in the advanced query panel directly on PM. In general, this textual field allows any syntax from PM and PMC, because the query will be redirected directly to those services to obtain the most relevant articles. PM and PMC queries will be annotated with OntoTagME and enriched with PubTator (as described in Section 4.4). In case the PubTator REST API is down, then the papers will be downloaded through a backup REST API (NCBI RESTful API[3]), and annotated with OntoTagME alone. Additionally, NetME allows to add a set of additional optional parameters, which are:

- *Search on*: This can be set to:
  - "Search from query terms", in which case we provide PM or PMC with a textual query, and it will be up to PM and PMC to provide the n most relevant papers according to the query.
  - "Search from specific Paper ID", in which case we provide PM or PMC with a list of papers to annotate, and NetME will build the network directly on these abstracts or full-texts respectively. The format is ID1, ID2... for PM and PMCID1, PMCID2... for PMC.
- *Search type*: This can be set to "Full-Text Article (PMC)" or "Abstract (PM)", and the network will be built using full-texts or abstracts respectively.

2 https://netme.click/
3 https://www.ncbi.nlm.nih.gov/research/bionlp/APIs/BioC-PMC/

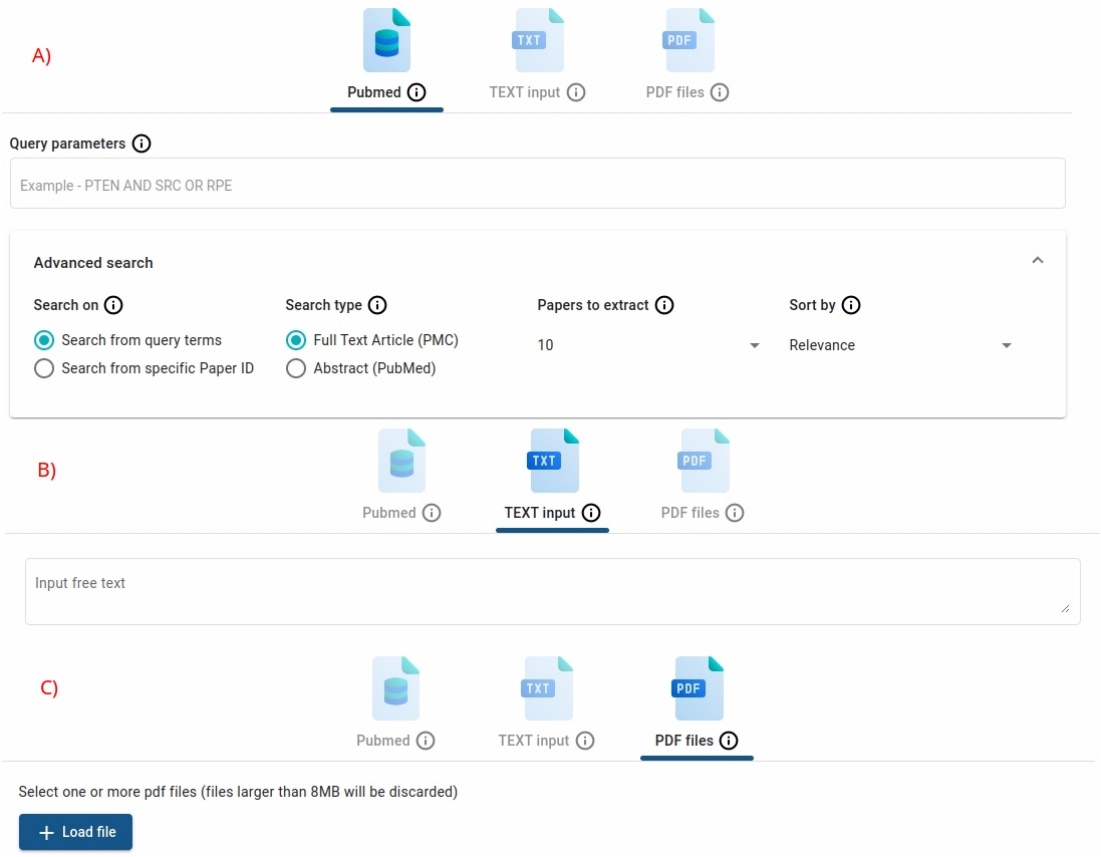


Figure 16: NetME "query" interface

- *Papers to extract*: Number of papers to annotate, where the possible numbers are $10, 20, 50, 100, 500$. Keep in mind that, in the case of PMC queries, this number has to be considered as an upper bound. This is because we annotate only the *open-access* articles in the set of n most relevant full-texts with respect to the user's query.
- *Sort by*: This can be either set to "Relevance" or "Date". In the first case, we rank papers according to the PM and PMC search ranking function, while in the second case, we will provide the n most recent papers satisfying that query.

TEXTUAL QUERIES are more straightforward, as they just provide the user with a text box to fill. There, the user can input a string of any length, and a network will be generated on said text. This text will be annotated with OntoTagME, and no PubTator enrichment will be performed. We took this decision because of the large amount of time required to annotate free text on PubTator.

THE PDF QUERY allows the user to directly upload a set of scientific papers for NetME to process. A limit of 8 MB has been put on papers in order not to stress the system. Papers of larger size will be discarded. in order to parse them, we use the Python library PDFminer[4]. The PDF parsing module exploits a set of strategies to understand whether the paper is written in one or two text columns. It also checks

4 https://pypi.org/project/pdfminer/

the width of the text parts in order to remove captions, tables, and similar non-textual parts from the parsed page. Annotations, like for textual queries, will be performed by OntoTagME alone, with no PubTator enrichment.

5.1.2 *Network GUI*

NetME's main GUI is the network panel, shown to the user as a response to the query. It is the view where the on-the-fly generated network is shown, along with tools to inspect and modify it. Figure 17 shows a snapshot of an example query. It was generated by making a PMC query with the text "colon cancer". The GUI consists of four panels that are boxed in red (left), orange (center), green (right), and blue (bottom), each one referring to a different feature.

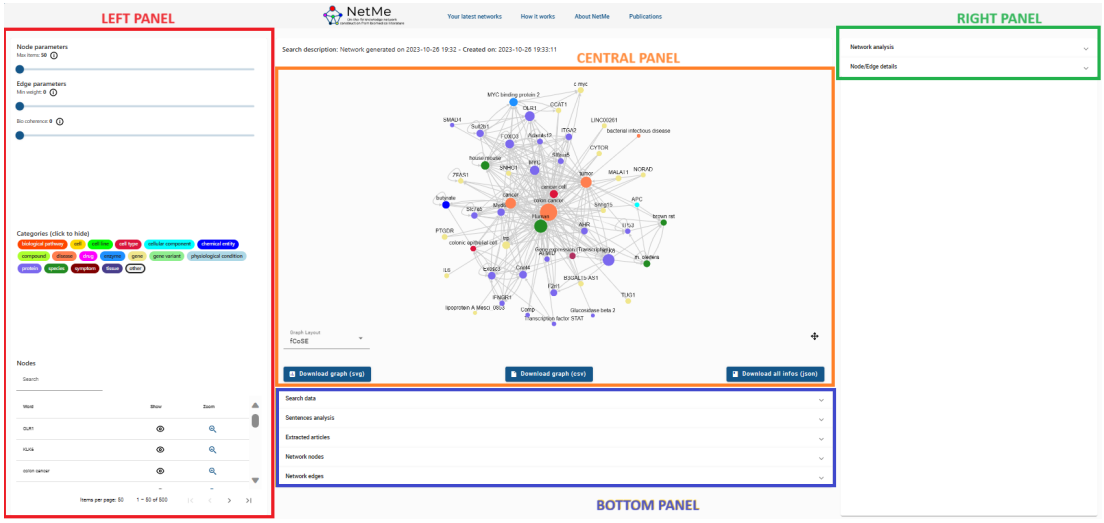


Figure 17: NetME GUI showing the result of the query "colon cancer"

The spotlight goes to the resulting network, shown in the middle, which is displayed using the Cytoscape library.

A closer look at the central panel can be seen in Figure 18, which shows an example resulting from performing the query for the 10 most relevant full-texts on PMC for "*CACNA1A*". This image was downloaded from NetME directly using the button "Download Graph". As we're able to see, the system displays the n most relevant nodes to the query. Bigger nodes refer to more important ones with respect to the query, where this importance value is computed using a specific variant of the Personalized PageRank algorithm (more details in Section 5.4). Nodes have different colors for each of the macro-categories we selected during the design of OntoTagME (shown in Section 4.3.2). For example, light yellow is for genes, dark green is for species, light green is for gene variants, and orange is for diseases.

NETWORK INSPECTION. Both nodes and edges can be clicked on in order to obtain details about them, and they will be shown on a dedicated panel on the right-hand side of the web app, as shown in Figure 19. This refers to the green panel in Figure 17. The left panel is the one that shows whenever the user clicks on a node (in the case of Figure 19, gene CCAT1). In this case, the GUI shows a table containing one row

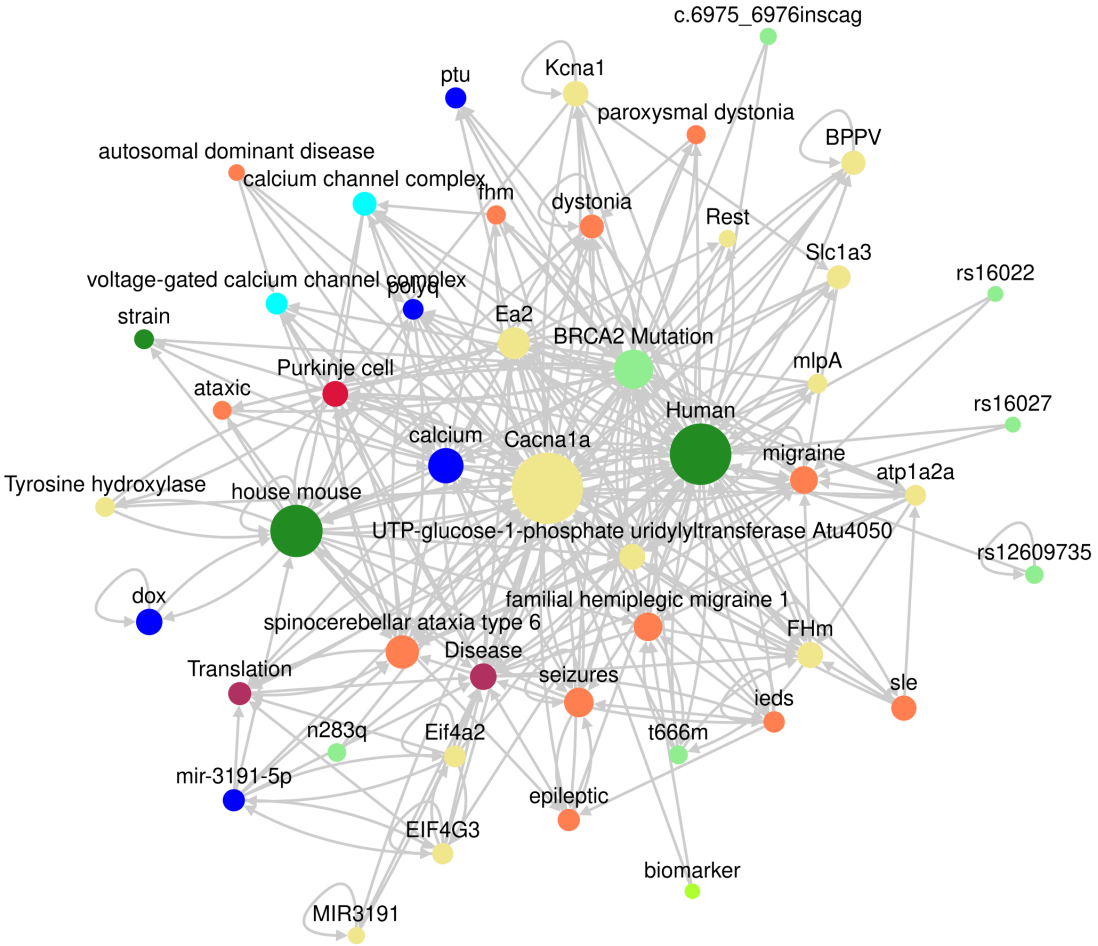


Figure 18: NetME example network for gene CACNA1A

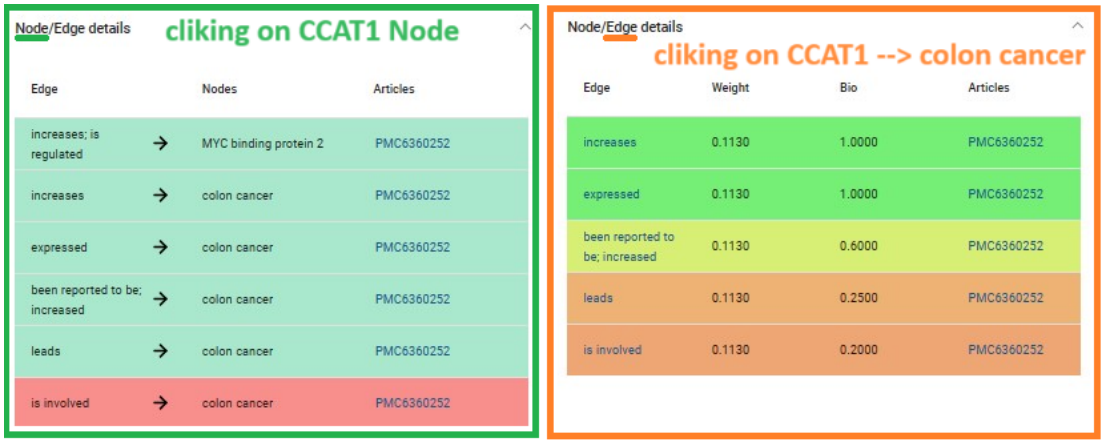


Figure 19: NetME's network inspector. The left panel is for nodes, the right panel is for edges

for each outgoing edge from the selected node. For example, the first row means that there is an edge going from "CCAT1" to "MYC binding protein 2", labeled "increases; is regulated", and this edge was extracted in the full-text PMC6360252. Green is used for edges that have a higher degree of relevancy with respect to the set of biological

verbs we model, while red is used for less relevant verbs. More information on this is in Section 5.2.1.

The arrow is bigger as the *weight* of the edge grows. The way this weight is computed will be described in Section 5.2.1. The user can click on the article and a special panel showing the phrase where the edge was extracted will show. Figure 20 shows this panel for the edge *CCAT1* $\xrightarrow{\text{increases}}$ *colon cancer*.

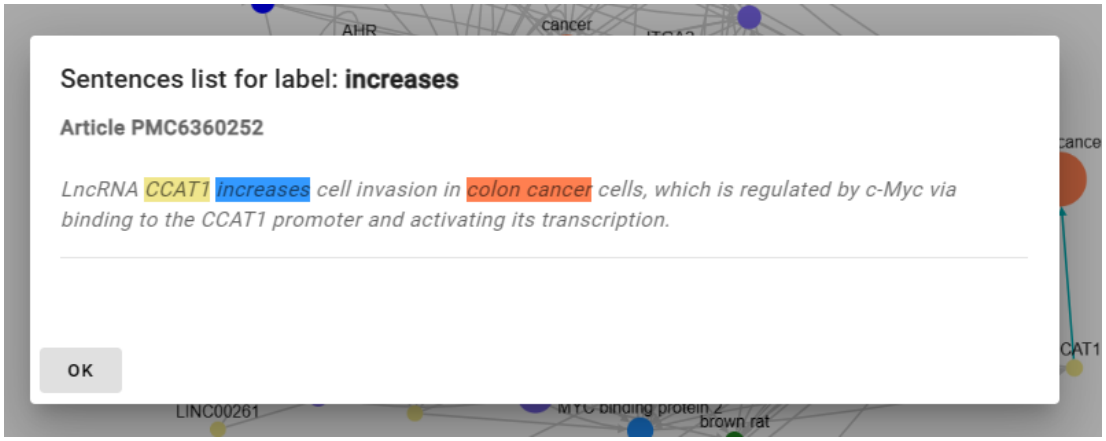


Figure 20: NetME's sentence details panel

As it is possible to see, the web app shows the full relevant phrase and the PMC ID where the interaction was found. The verb is highlighted, along with the two mentions of the two entities associated. The colors shown are the ones of their respective macro-category.

If the user instead clicks on an edge, then the panel view will be the orange one shown on the right of Figure 19, where we show the view for the relation *CCAT1* $\rightarrow$ *colon cancer*. In this case, we get a row for each unique *verb* that was found relating the two entities in this specific direction. The displayed information is the verb, the weight, the bio-score, and the list of articles where this relation was found. Weight and bio-score are two numerical properties that is described in detail in Section 5.2.1, and they express respectively the importance of the edge and the closeness to the set of biological verbs we consider. The verb is clickable, and in case the user does they are shown the same phrase view as the one in Figure 20. By instead clicking on the paper ID, a tab will be opened on their browser with the PM page of the article where the edge was found.

THE USER CAN MODIFY THE NETWORK VIEW to obtain a visualization that best fits their needs. This refers to the red panel on the left of Figure 17. Figure 21 shows the panel that can be used to do so. In the image, the top panel provides three sliders:

- The first one allows to change the total number of nodes to be displayed in the network panel. Only a maximum of 300 nodes can be shown for ease of visualization (the default value is 50). Changing this slider will keep only the n most relevant nodes according to their *personalized Page-Rank* which is be described in Section 5.4.
- The second slider allows the user to change the weight. High-weight edges correspond to the most important ones and the ones that have the most biological

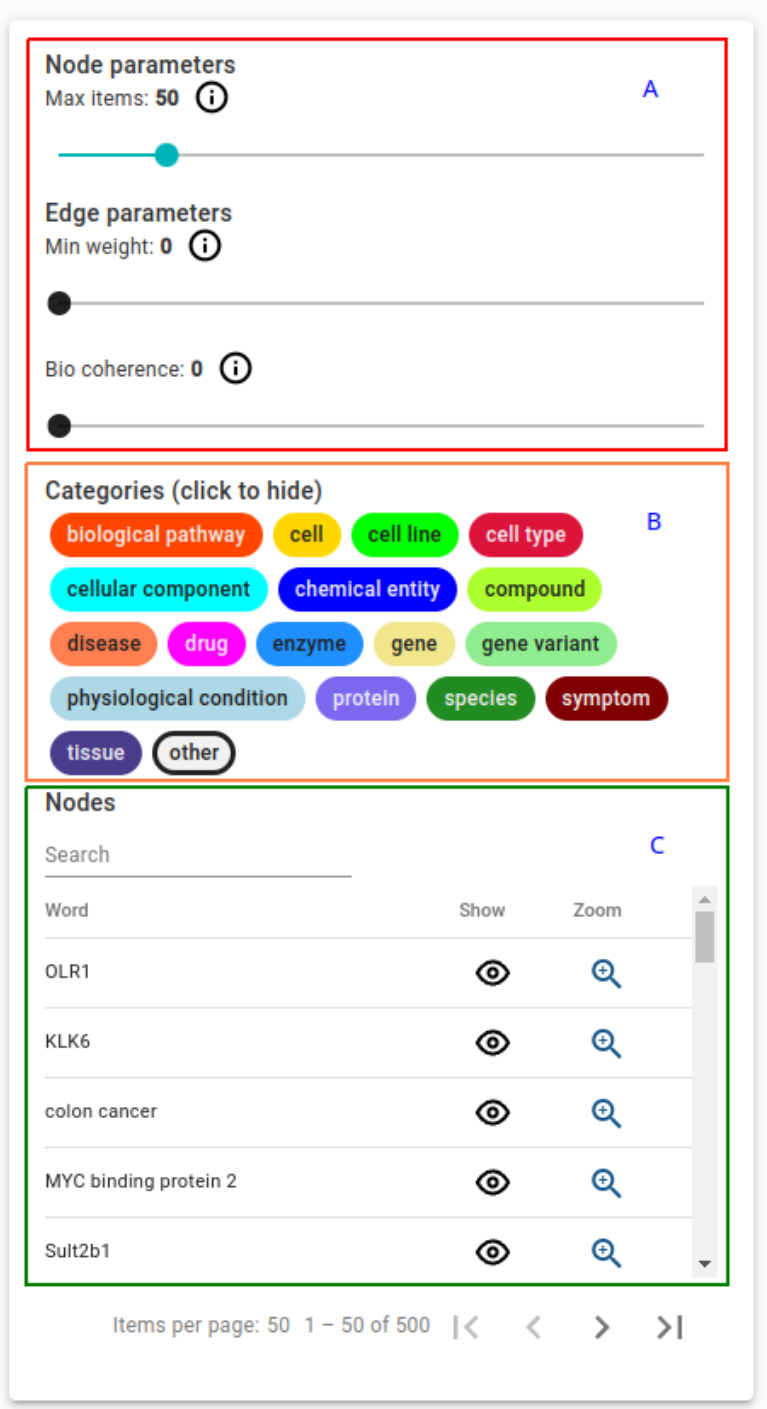


Figure 21: NetME's network inspector panel

evidence in the set of provided papers. Setting this slider to ∅ disables this filter, and all the edges will be shown (default value is ∅). More details on the computation of the weight are in Section 5.2.1

- the third and final slider allows to set the bio coherence value. Setting it to ∅ disables the filter, and setting it to 1 filters out all of the *less-biological* edges

we found (default value ∅). This concept and the detailed explanation of the bio-coherence parameter are in Section 5.2.1.

Under the sliders, there is the category panel. This panel shows a togglable option for each macro-category, from the set of macro-categories modeled by OntoTagME. Clicking on one of those options will enable/disable any node in the graph of that category. By default, all of those options are active.

Below, the nodes list is displayed, sorted by decreasing *personalized PageRank* score. Clicking on the "Show" button will make the selected node appear/disappear, and clicking on the "Zoom" button will highlight the node in the network panel, in order to ease the process of finding it.

THE NETWORK ANALYSIS PANEL is a NetME 2.0 exclusive. It allows the users to run a set of standard data science algorithms on the network. It offers a set of panels with relevant algorithms:

- *Traversing*: The user can check the neighborhood of a node or the connected components.
- *Search*: The user can perform a BFS, DFS, or do a shortest path computation using the Dijkstra algorithm.
- *Centrality*: Computes the betweenness centrality and displays the PageRank of all the nodes.
- *Clustering*: The user can perform a clustering algorithm. They can choose between *Markov Clustering*, *K-Means*, and *Hierarchical Clustering*. The results of the clustering algorithms can be exported in CSV files for deeper analysis.

THE NETME BOTTOM PANEL is the blue panel at the bottom of Figure 17, and it allows for a deeper analysis of the network. Its expanded view is shown in Figure 22.

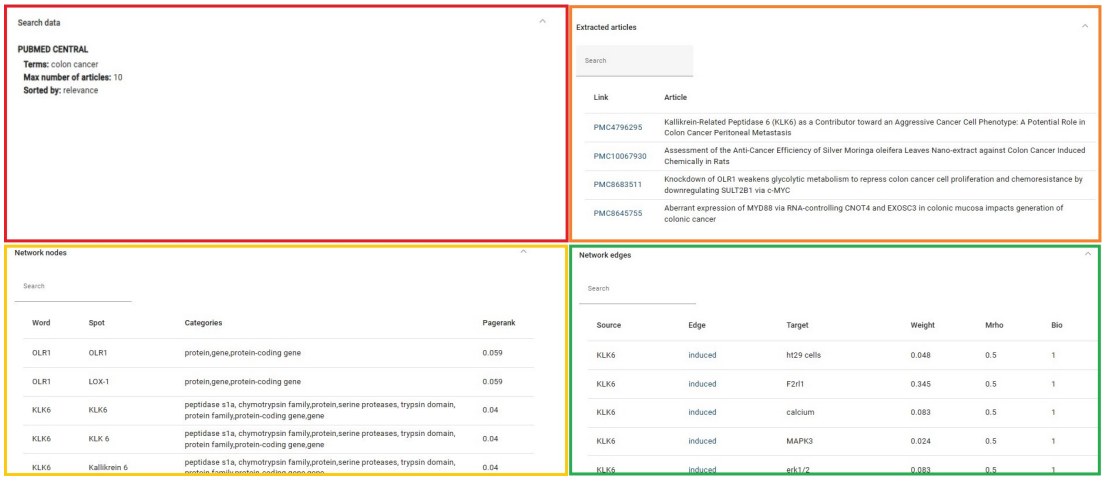


Figure 22: Bottom panel of the NetME GUI. Top left is "Search Data", top right is "Extracted Articles", bottom left is "Network Nodes", bottom right is "Network Edges"

It is split into five panels. Figure 22 shows four, as the fifth one (sentence analysis) is a NetME 2.0 exclusive, and will be explained in detail in Section 5.4.

- The *"Search Data"* panel shows the recap of all the query's details. It shows the type of query performed, the textual prompt, and any attached user-defined property.
- The *"Extracted Article"* panel shows, for PM and PMC queries, the titles of the papers used to build the network, along with their IDs and link to their respective PM and PMC pages.
- The *"Network Nodes"* panel shows a table with each node in the network. Each row contains the following information: (i) Word, the title of the Wikidata page and the name of the node; (ii) Spots, the list of mentions found in the query associated with this node; (iii) Categories, the list of categories associated to this node; (iv) PageRank, the Personalized PageRank float value of the node, computed as described in Section 5.4.
- The *"Network Edges"* panel shows a table with each edge in the network. Each row contains the following information: (i) Source, the source node of the relationship; (ii) Target, the target node of the relationship; (iii) Edge, the textual edge label (e.g., "is activated"), and clicking on it will open the NetME sentence details panel like the one in Figure 20); (iv) Weight, the edge weight; (v) Mrho, the edge Mrho; (vi) Bio, the bio-score. The last three floating point parameters are computed as described in Section 5.2.1.

NETME PROVIDES THREE DOWNLOAD BUTTONS below the network panel, which are:

- *Download graph (SVG)*: provides a direct download link to the image of the graph, as displayed through Cytoscape in the network panel. The image is provided for download in an SVG format.
- *Download graph (CSV)*: downloads two files, nodes.csv and edges.csv. These files contain the fields that are shown in the bottom panels for nodes and edges.
- *Download all infos (JSON)*: downloads the full dump JSON file used by NetME. It contains all the information about nodes, edges, and the sentences retrieved.

5.2 NETME 1.0

NetME's first version was published in an Applied Network Science paper [10]. This version had fewer features than the current release, but we still show here the core design principles and the case studies envisioned to test the quality of the networks. All the functionalities that are exclusive to NetME 2.0 and all the improvements with respect to this version will be described in Section 5.4. Figure 23 shows the architecture of NetME 1.0.

As discussed in Section 5.1, we allow three types of queries. For PM and PMC queries, we use the Entrez eSearch and eFetch APIs [88] to query the two services and download the abstracts and full-texts respectively.

For the extraction of nodes, we use OntoTagME (version 1.0) as the entity linker. The full set of NetME's biological DBs and ontologies is the same one of OntoTagME 1.0, and the full list is provided in Section 4.2.

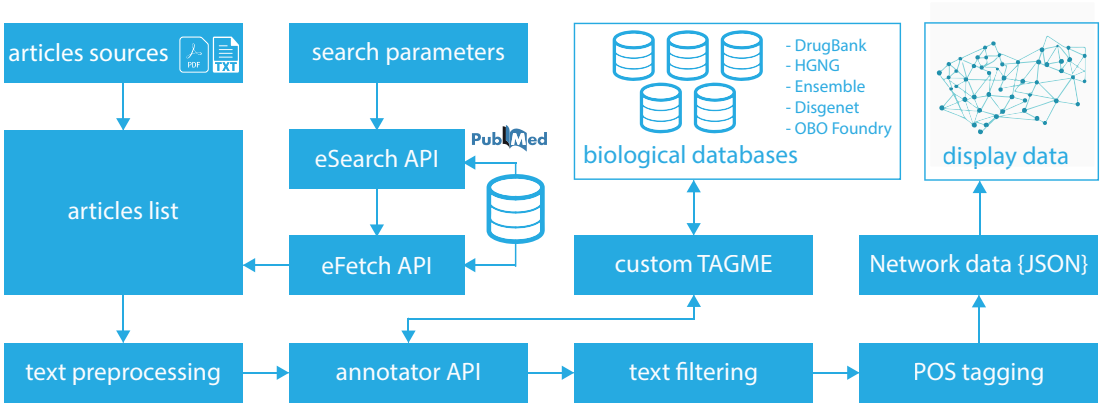


Figure 23: NetME 1.0 architecture

5.2.1 *Phrase Engine*

Once the network nodes have been extracted (with OntoTagME) the next step is the one of phrase processing, in order to understand whether any potential valuable relationship exists between couples of entities.

The key idea was to look for biological relationships only where a specific set of verbs appeared. For this reason, we referred to an expert in biology, who provided us with a list of *biologically relevant verbs*. The full list is shown in Table 8. As we can see, verbs range from very common ones (e.g., "find") to very specific ones (e.g., "upregulates").

Verb Forms		
activate	downregulate	reduce
affect	enhance	regulate
associates	express	release
block	find	reveal
cause	inactivate	stimulate
contain	increase	trigger
control	induce	ubiquitination
decrease	interacts	upregulates
detect	overexpress	
display	produce	

Table 8: List of biological verb forms used by NetME

The next step is the one of dependency extraction. This is done to ensure that we properly parse the phrase, and choose which entities are referring to the given biological verb, along with the proper direction of this edge.

To solve this task, we use PoS tagging (more details in Section 2.2.2). We will then verify the coherence of the extracted elements through syntactic analysis.

Indeed, sentences have an internal organization that can be represented using a tree. Solving a syntax analysis problem for a sentence consists of looking for predefined syntactic forms that, like a tree, branch out from the single words. The main syntactic form is the sentence (S), which contains noun phrases (NP) or verb phrases (VP) that are formed by further elementary syntactic forms such as nouns (N), verbs (V), and determiners (DET). All this information will be used in the textual analysis phase to infer relations between them.

In order to extract the mutual dependencies of entities we use the dependency parsing module of spaCy. As described in Section 2.2.2, spaCy employs a transition-based dependency parser to check the syntactic coherence, used to then build the syntactic tree. The dependency parser component inside the spaCy library jointly learns sentence segmentation and labeled dependency parsing. The parser uses a variant of the non-monotonic arc-eager transition system, with the addition of a break transition to perform the sentence segmentation. Nivre's pseudo-projective dependency transformation is also used to allow the parser to predict non-projective parses.

Irrelevant PoS are filtered out (stop-words, URLs, etc.), and we keep only the useful verb forms and the nodes that correspond to the noun parts.

A final pruning phase is also executed in which we use:

- PoS tags and dependency labels to check if the syntactic link between the verb form and the annotations is correct and consistent.
- A dictionary of biological verb forms to check if they are pertinent. The surviving nodes and verb forms will allow the generation of network edges.

We define a set of edge scores and weights for the purpose of ranking edges. Each edge $e = (a, b)$ is weighted with three parameters: the term frequency and inverse document frequency (TF IDF, more in Section 2.2.1), the average relatedness (mrho) and the biological degree (bio). In particular:

- *edge weight*: it is the TF IDF of that edge, computed considering as the dictionary the set of documents provided by the user, and as a *token* the triple $(\text{source}, \text{target}, \text{verb})$. The full TF IDF is shown and explained in Section 2.2.1.
- mrho: measures the relatedness of the labels starting from the ρ value assigned by OntoTagME to the two annotations involved. This is done by averaging the two individual rho values, i.e., $\text{mrho}(e) = (\rho_a * \rho_b)/2$.
- bio: the degree of similarity (having a value ranging from $\emptyset$ to 1) between the inferred relationship and a set of biological verb forms (see Table 8). This is a normalized Levenshtein Distance [132] (also called edit distance) computation.

An example of a full annotation process on the sentence *"...CD147 regulates several VEGF isoforms and placental growth factor (PLGF), and it has unique effects on trophoblastic function..."* is shown in Figures 24 and 25. Through OntoTagME we detect the mentions [*"BSG"*, *"VEGFA"*, *"PGF"*]. Then the PoS tagging and dependency parsing phase is deployed (Figure 24), where three noun parts are identified (the phrase mentions, highlighted via orange segments): two of them ("VEGF" and "PLGF") have a joint relationship with "CD147". The verbal part is the label of the edges for the two pairs of nouns ("CD147" - "VEGF"), (CD147 - "PLGF"). Finally, two valid edges are

detected, according to the biological verbs we model: [*"BSG"*, *"regulate"*, *"VEGFA"*] and [*"BSG"*, *"regulate"*, *"PGF"*]. Note that "regulate" is a biological verb form and its bio score is equal to $\emptyset$, which corresponds to the maximum possible value.

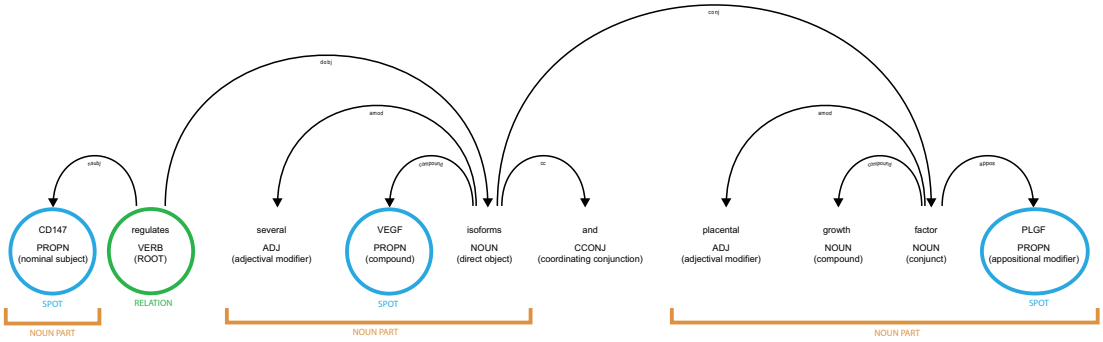


Figure 24: example of PoS extraction and coherence checking

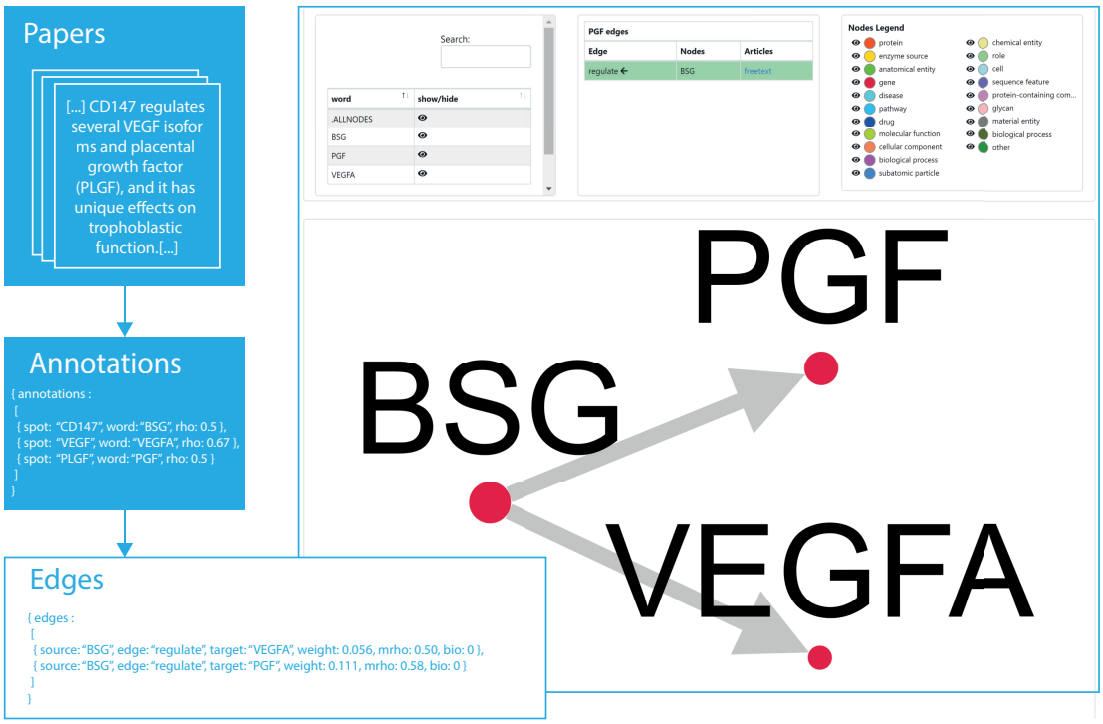


Figure 25: NetME example annotation, full process

5.3 NETME 1.0 - EVALUATION

We performed three case studies to analyze the reliability of the generated KGs. The first one (Section 5.3.1) aimed at providing a comprehensive analysis of NetME's performance by checking its ability to predict known relations between genes drawn from Kyoto Encyclopedia of Genes and Genomes - KEGG [126, 133, 134] or REACTOME [121–123] pathways and, on the other hand, its ability to avoid inferring false connections by using the Negatome 2.0 DB [135, 136]. The second case study (Section 5.3.2) is more specific and focuses on building a network based on some selected

publications that contain valuable information specific to the CD147 gene. Such a network is then compared against a manually-curated one derived from the same papers by a bio-expert. In both cases, the performance of NetME has been measured in terms of a precision/recall curve. The third case study focuses on DisGeNET's GDAs, and measures NetME's ability to catch them under supervision on the relevant set of articles.

5.3.1 *Case Study 1 - PubMed queries*

The first case study focuses on assessing NetME's performance through its capability to recover known gene interactions. For this purpose, we selected a subset of gene-gene interactions from KEGG/REACTOME by making use of STRING API. More precisely, such interactions were obtained by selecting 100 random gene-gene interactions for each of the following STRING text-mining score intervals: $500-600$, $600-700$, $700-800$, $800-900$, $\geqslant 900$. These interactions form the true-positive set.

Next, we selected 100 random pairs of non-interacting genes from the Negatome 2.0 DB as a true-negative set. For each interacting gene pair, we queried NetME with the papers used by STRING to infer the interactions. On the other hand, to annotate non-interacting genes, we queried NetME with the pair of genes of interest, selecting the top 20 papers from PM. Accuracy, sensitivity, specificity, and PPV values, detected by NetME, are listed in Table 9 The results clearly show that NetME produces reliable results when the annotations are performed on top of relevant literature (STRING text-mining score higher than 700). On the other hand, when the STRING text-mining score is lower than 700, NetME's performances degrade in accordance with STRING predicted confidence as highlighted by their score. The reason behind such a behavior is due to: (i) not enough literature about these interactions; (ii) the interactions have been inferred by human curators as a combination of other interactions occurring in the text. Furthermore, when the text-mining score is small, STRING predictions could be wrong. In fact, as reported in [16], a score of 500 would indicate that roughly every second term of an interaction might be erroneous (i.e., a false positive). Therefore, the computed values of accuracy, sensitivity, specificity, and PPV could be incorrect.

text-mining score	accuracy	sensitivity	specificity	PPV
$500-600$	58.5%	31%	86%	68.8%
$600-700$	66.5%	47%	86%	77.1%
$700-800$	72.5%	59%	86%	80.8%
$800-900$	73.5%	61%	86%	81.3%
$\geqslant 900$	84%	82%	86%	85.4%

Table 9: NetME 1.0, Case Study 1, Metrics

5.3.2 *Case Study 2 - Query from papers*

Many tools [125] and computational models rely on existing network DBs, such as KEGG and REACTOME. However, despite the enormous amount of available data, these DBs are still incomplete and therefore have partial information [137]. As an example, KEGG includes approximately one-third of the known genes.

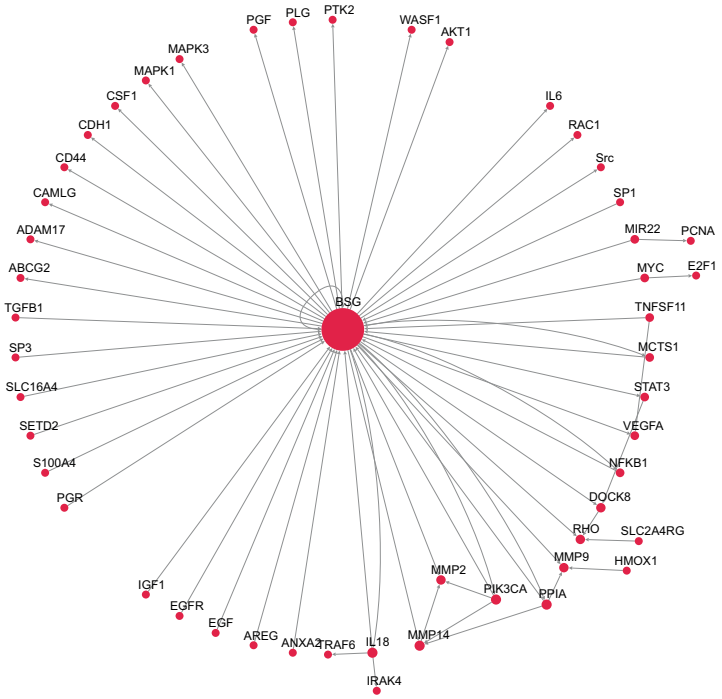
In this case study, we have chosen CD147, also known as Basigin, BSG, or EMMPRIN. This gene represented an example of a biological element that should have been supplemented to the KEGG network since, at the time of performing these tests, it was not described in their pathways. Among the bibliography consulted to build the network manually, we have carefully selected 10 papers containing a significant amount of helpful information for our purpose.

CD147 is a transmembrane glycoprotein of the immunoglobulin superfamily, expressed in many tissues and cells, which is known to participate in several high biological and clinical relevance processes and is a crucial molecule in the pathogenesis of several human diseases [127]. Recently, Wang et al. [138] discovered an interaction between host cell receptor CD147 and SARS-CoV-2 spike protein, together with Angiotensin-Converting Enzyme 2 (ACE2), as an entry point for SARS-CoV-2. For this reason, CD147 is an example of how a missing crucial gene within a biological network can compromise scientists' efforts to understand certain molecular phenomena.

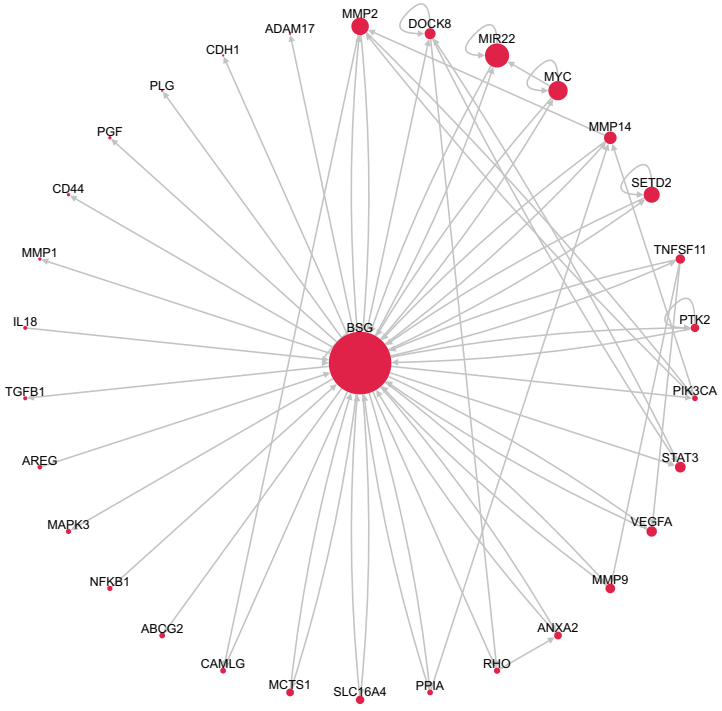
Figure 26 shows the comparison between the two networks. More specifically, Figure 26a depicts the pathway constructed by hand by a biological expert from the selected papers [127, 139–147], with CD147 (BSG) as the central node. Figure 26b shows the network synthesized by NetME using as input the same set of papers.

The biological expert states that NetME is able to show that CD147 is a potent inducer of metalloproteinases (MMPs) such as MMP2, MMP14, and MMP9 as reported in [127, 143, 144]. Furthermore, the overexpression of CD147, which results in increased phosphorylation of PI3K(PIK3CA), and Akt(AKT1), leads to the secretion of vascular endothelial growth factor (VEGFA) in several biological contexts such as KSHV infection [127, 143]. In addition to its ability to induce MMPs, CD147 regulates spermatogenesis, lymphocyte reactivity, and MCT system, in particular, MCT1 and MCT4 (MCTS1 and SLC16A4) expression [127, 147]. Our results also show that CD147 can increase the expression of ATP-binding cassette transporter G2 (ABCG2) protein, regulating its function as a drug transporter, as mentioned by Xiong, Edwards, and Zhou [127] for MCF-7 cells. NetME identifies also BSG as an upstream activator of STAT3, highlighting its involvement in tumor development in agreement with the literature [146]. As summarized by our knowledge network, CD147 is regulated by various inflammatory mediators, such as RANKL (TNFSF11), denoting its involvement in inflammatory processes [142, 143]. Among the potential activators of BSG, NetME also finds the transcription factor c-Myc (MYC) [140].

Our second case study shows a practical example of how NetME can create valuable networks by analyzing quickly and automatically some publications. This process resulted in 50 genes and 64 interactions. Next, by using the same set of papers, we run NetME with no upstream filter. The automatically generated network consisted of 86 genes and 139 edges. As the manually curated network consists of genes and proteins, only elements from these two categories were selected for the evalua-



(a) Handmade BSG Network



(b) NetME's BSG network

Figure 26: Comparison between the handmade and NetME's BSG networks on the 10 papers

tion. This was performed by considering edges with the lowest "bio" score for each node pair (which corresponds to the set of most biologically relevant verbs). Qualitatively, this network includes most of the interconnections mentioned in the papers, thus providing a reliable and comprehensive overview of the molecular function of

Basigin. Quantitatively, NetME achieved an accuracy of 98.99%, a sensitivity of 100%, a specificity of 98.98%, and a positive predicted value of 46.32%.

5.3.3 *Case Study 3 - 100 random DisGeNET associations*

Finally, we randomly sampled 100 DisGeNET GDAs with gene CD147, and we let NetME generate networks using the same PM abstracts used by DisGeNET for inferring those interactions. NetME detected 63 True Positive values out of 100, revealing a sensitivity of 63%. This case study will be repeated for NetME 2.0 in Section 5.5.3 to showcase the power of the new engine.

5.4 NETME 2.0

In order to improve the performance of our on-the-fly network-building tool, we decided to update several of the modules. Changes were performed both to the front-end and back-end of the system, and they were deemed significant enough to justify a new major release.

This new version of NetME, now called NetME 2.0 [13], has a new pipeline associated with it, as shown in Figure 27.

The core idea and design principles are the same ones as NetME 1.0, but the resulting networks achieved a significant boost in the quality of the bio-annotations, quality of the retrieved edges, noise reduction techniques, and usability of the tool. The new case studies performed to ensure the quality of the networks are in Section 5.5.

The first main upgrade is the switch from OntoTagME 1.0, described in Section 4.2, to OntoTagME 2.0, described in Section 4.4. Its differences have been described in detail in the respective sections. Additionally, NetME 2.0 comes with some GUI improvements, such as the new graph-RAG module and the data-science algorithms, which were not present in NetME 1.0.

The new phrase engine works as follows: once OntoTagME has extracted the entities (nodes) from a list of full-text documents, the edge inference module comes into play to infer any verbal relationship between entity pairs by means of the following seven steps.

STEP 1 (SENTENCE SPLITTING). Each input document is split into a sequence of sentences. This step is carried out through the spaCy pipeline that segments sentences via a set of rules based on punctuation and statistical language-based models, which take into account the probability of a specific word marking the end of a sentence based on a spaCy corpus. The sentence splitting is important because many NLP tools perform their text analysis and processing on a sentence-by-sentence basis, allowing more/less granular and targeted NLP operations.

STEP 2 (NODE TAGGING AND EXTRACTION). Each sentence is divided into individual tokens (words or sub-words) which are then assigned a grammatical tag, indicating its PoS (e.g., noun, verb, determiner). Then they are assigned a canonical form (lemma) based on its PoS. For example, the term "increased" is transformed into "increase". Lemmatization offers several benefits in NLP tasks because it reduces

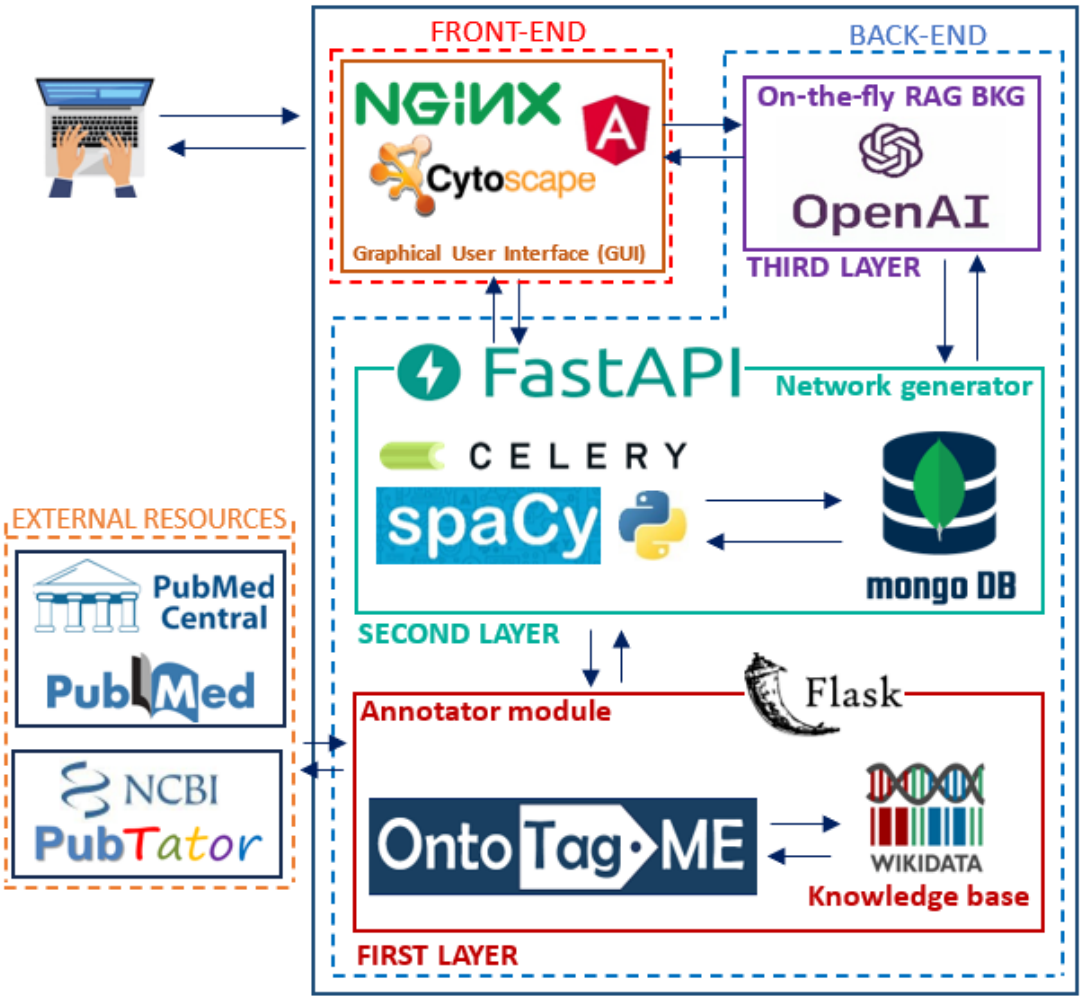


Figure 27: NetME 2.0 architecture

inflectional and derivational variations to facilitate easier comparison and analysis of textual data. Finally, they are assigned a Named Entity, if any: such as person name, organization, location, and date. Table 10 shows the output from spaCy after the two steps described above for the sentence "TP53 expression increased in colon cancer".

token	lemma	PoS tag
TP53	TP53	NNP
expression	expression	NN
increased	increase	VBD
in	in	IN
colon	colon	NN
cancer	cancer	NN

Table 10: SpaCy tokens and their PoS tagging for the sentence "TP53 expression increased in colon cancer"

STEP 3 (DEPENDENCY PARSING). Once the tokens have been detected and analyzed, the spaCy dependency parse tree builder module is adopted to determine the syntactic hierarchical dependencies between entity pairs. Figure 28 illustrates the dependency parse tree generated by spaCy for the sentence *"TP53 expression increased in colon cancer"*. The tree has the node labeled "increased" as its root, which depends on the token "expression" (as its *subject*), and the token "cancer" (as its *object*). The token "expression" depends on the token "TP53" (as its *compound*), while the token "cancer" depends on the token "colon" (as its *compound*). Irrelevant PoS are then filtered out (such as stop-words, URLs, etc). The final substep consists of keeping only the tokens that constitute mentions to biomedical entities (except verbs and adverbs that will be used to create edges) by adopting the annotations provided by OntoTagME.

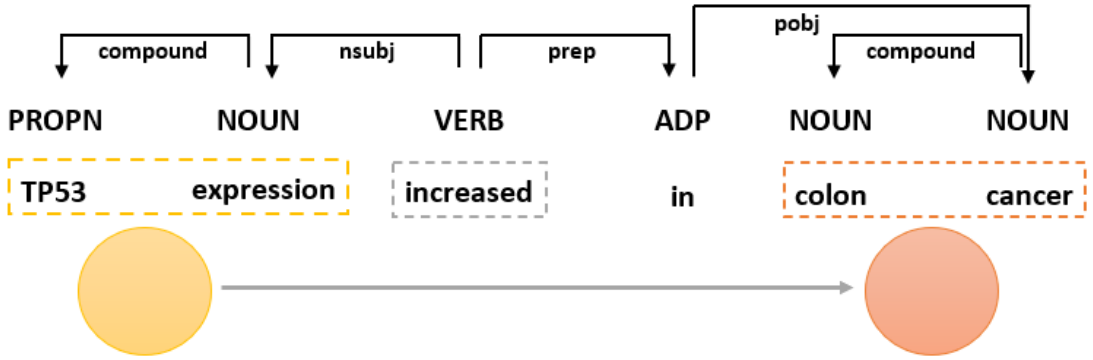


Figure 28: spaCy tree example on phrase: "TP53 expression increased in colon cancer"

STEP 4 (EDGE LABELING AND EXTRACTION). NetME 2.0 implements a relationship extraction module on top of spaCy in order to transform every action node of the dependency-parse tree into a relationship (edge of the graph) between two biomedical entities. More precisely, given a source and a target node in the dependency-parse tree, if a (undirected) tree path containing one or more actions (verbs) connecting these nodes does exist, then edges labeled with those actions are generated. For example, the sentence in Figure 28 generates an edge labeled with the verb "increased" connecting "TP53 expression" and "colon cancer" because of the existence of the (undirected) tree path from these two nodes. When we have just one action between the source and the target node, the edge label corresponds directly to that action.

Conversely, if the number of actions is more than one, the edge label is formed by making a semi-column concatenation of the actions between the source and the target nodes. This is useful because in certain cases, OntoTagME fails to annotate one or more biological entities. For example, if we analyze the sentence in Figure 29, OntoTagME does not annotate the terms *"cell viability"*, *"cell motility"*, and *"circPIP5K1A overexpression"*. Consequently, relationships with *"circPIP5K1A expression"* as the source node will lack the target node because both *"cell viability"* and *"cell motility"* are absent. Furthermore, the relationship with *"colon cancer"* as the target node will lack the source node due to the absence of annotation for *"circPIP5K1A overexpression"*. To ensure that no edges are missing in such a sentence, the module will establish a connection between *"circPIP5K1A"* and *"colon cancer"* with label *"attenuates; reduces; facilitates"* by an appropriate traversal and exploration of the syntactic tree. Multiple actions are exclusive to NetME 2.0, and they were not present in the

1.0 version. Additionally, the user can highlight the missing annotation in the GUI (as shown in Figure 30), thus notifying us. This allows us to manually check the requests, and periodically re-index OntoTagME adding this entity to the set of modeled ones.

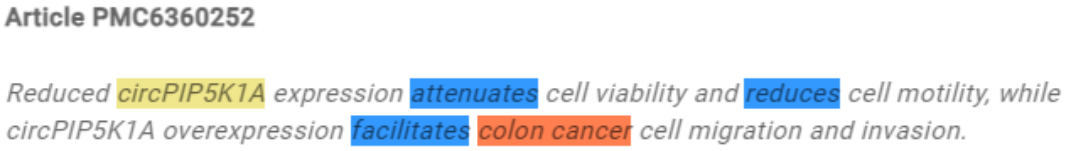
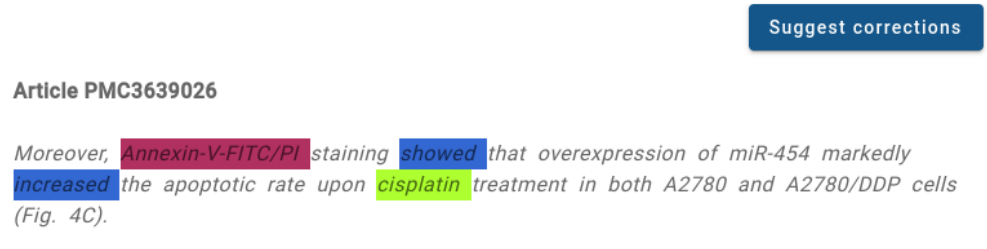
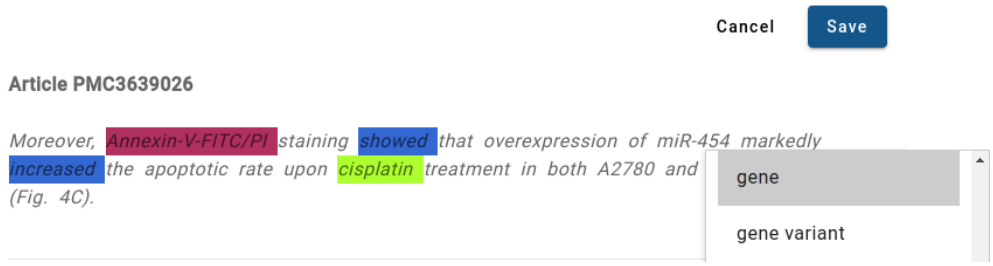


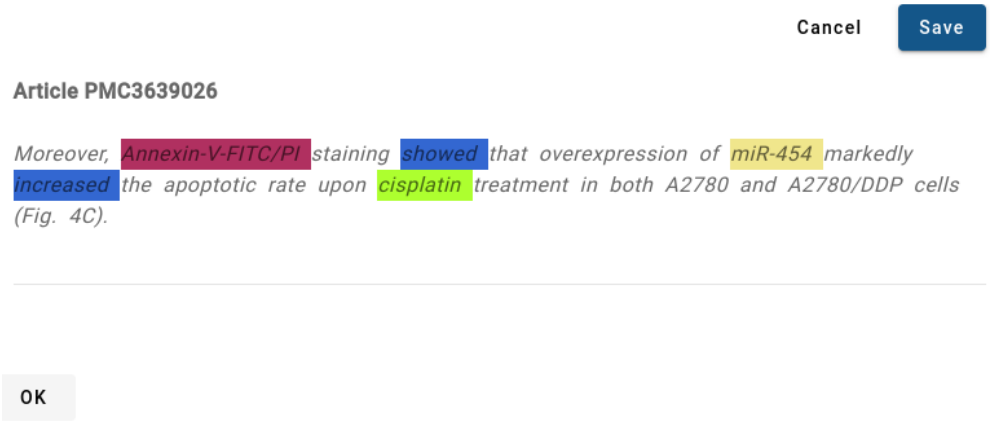
Figure 29: Example of multiple actions in NetME 2.0



(a) "Suggest corrections" button



(b) Selection of "miR-454" and its category (gene)



(c) "OK" button for sending every suggestion

Figure 30: Process guiding a user to propose the inclusion of the absent biological term "miR-454"

STEP 5 (EDGE SCORING). Each edge $e = (a, b)$, connecting two entities a and b is labeled with three numerical values: the TF IDF of the edge in the input documents, a measure of its biological coherence (*bio*), and a measure of the ambiguity of connecting a with b. More specifically, TF IDF (more in Section 2.2.1) quantifies how relevant a term is in a document collection, here adapted to work on labeled edges (hence, triples) rather than terms. Given the edge e and a document d in the input collection D of size N, we define:

$$\text{TF IDF}(e, d, D) = \text{TF}(e, d) * \text{IDF}(e, D)$$

where $\text{TF}(e, d)$ is the number of times the edge e has been detected in document d, and $\text{IDF}(e, D) = \log N/N_e$ is the so-called inverse document frequency, here specialized to work on edges, with N_e denoting the number of input documents including e. The *bio*-parameter measures the *normalized edit distance* (having a value ranging from $\emptyset$ to 1) between the label of the inferred relationship and a set of biological verb forms. The list of biological verbs is unchanged from NetME 1.0 and is shown in Table 8. The *ambiguity*-parameter measures how ambiguous an edge label is by computing the inverse of the number of actions labeling that edge. We remark that the choice of adding this third parameter comes from the experimental observation that OntoTagME might incur some false positives when annotating the relationships between biomedical entities, and thus we wish to make the user take into account those potentially erroneous situations by weighting properly the corresponding edges. Therefore, in the first example (Figure 28), the *ambiguity*-parameter is equal to 1, whereas in the second example (Figure 29), the value is 1/3. Finally, we mention that the GUI simplifies the management of the three parameters by just multiplying TF IDF and ambiguity into one single score, called *edge weight*.

STEP 6 (NODE CENTRALITY). After gathering nodes and edges, NetME computes an approximation of the *personalized PageRank* score over the resulting network in order to estimate the relevance of each of its nodes with respect to the one specified by the user's query. More details on the algorithm are in Section 2.1.2.

STEP 7 (NETWORK CONSTRUCTION). Nodes, edges, sentences, and all the other information extracted in the previous steps are used to generate the Biomedical KG, which is then stored as a JSON file (named with the user process-id, assigned at the user's request) and passed to the front-end GUI for visualization.

OTHER CHANGES FROM NETME 1.0. Some other upgrades have been put in place from the first version of NetME. In NetME 1.0, the edge inference process was generating too many noisy edge labels because, for a given sequence of tokens like "$entity_1$ $action_1$ $entity_2$ $action_2$ $entity_3$", the inferred edges were ($entity_1$)-[$action_1$]->($entity_2$) and ($entity_1$)-[$action_1$]->($entity_3$). All of those edges were considered equally important. This is a problem because NetME 1.0 was not considering the closeness of entities. This was inducing a lot of noise (in the previous example, the second edge could be a false positive). To mitigate this issue, NetME 2.0 assigns lower weights to edges that connect "far" entities so that these edges can be pruned at the time the graph is rendered via the GUI.

Moreover, NetME 2.0 changes the verb-centric approach to edge labeling to a more sophisticated approach that is based on paths occurring within the dependency-parsing tree of the sentence to be analyzed. In fact, edge relationships and labels are not generated by combining the entities around the detected verbs, thus possibly creating many false positive relations, but they are generated only if there exists a path in the dependency-parsing tree between those two entities. This has the twofold benefit of improving runtime performances and reducing noise.

Some other upgrades are more on the technical side. NetME 1.0 used PHP and JavaScript for the front-end, whereas Python was used for the back end. In NetME 2.0, the front-end has been re-implemented by using Angular JS, and its back-end has also been re-implemented by employing the Python FastAPI library[5] for the Rest API, and the Python Celery library[6] for the orchestration of the back-end logic. Celery's functionality empowers horizontal distribution across several servers and vertical scaling across different cores, thus speeding up the back-end operations. In addition, a docker-compose file has been written to facilitate the NetME 2.0 deployment.

RAG, NETME, AND OPENAI INTEGRATION. As we commented in the previous sections, the Biomedical KG built on the fly by NetME, as a result of a user query posted on PM is naturally suitable to express complex context information about Biomedical entities and relationships present in the query results, and so to provide more context information to help LLM better understand the relationship between biomedical entities and improve their expression and reasoning abilities.

The biomedical KG built on the fly by NetME turns out to be a natural candidate for designing a Graph-RAG application that better "understands" biomedical entities and their relationships. We call this novel approach *on-the-fly Graph-RAG* and refer the reader to Figure 31 for an illustration of its algorithmic structure. The red box contains the back-end modules, while the blue box contains the front-end modules. A user sends a query regarding a biomedical term using the NetME 2.0 GUI. Next, the KG is generated in accordance with the user's query, and visualized via the GUI. The user can then select a set of nodes of interest, which are passed to the Sentences Retrieval module to extract some sentences associated with the paths connecting such nodes. These sentences are then transmitted to OpenAI to generate a condensed and summarized text explaining the relationships among those biomedical entities.

For this reason, we have designed and implemented a graph RAG system on top of NetME 2.0 which was not present in the 1.0 version. The GUI of NetME 2.0 offers also the possibility to generate well-formed sentences that explain a user-selected set of biomedical entities (nodes) starting from suitably generated texts that are derived from the relationships modeled with the (edges of the) network constructed on the fly by NetME 2.0. This is the reason why we call this approach *on-the-fly Graph RAG* and, as far as we know, this is novel (more in Section 2.2.5).

This function is the fifth bottom panel in the network view GUI. More specifically, as shown in Figure 32, the user can select two or more nodes (entities) from the constructed network via a simple search box. NetME 2.0: (i) computes the set of all paths of a given maximum length (defined by the user) between all pairs of nodes in the selected set; (ii) it evaluates their score as the average of the weights of their

5 https://fastapi.tiangolo.com/
6 https://github.com/celery/celery

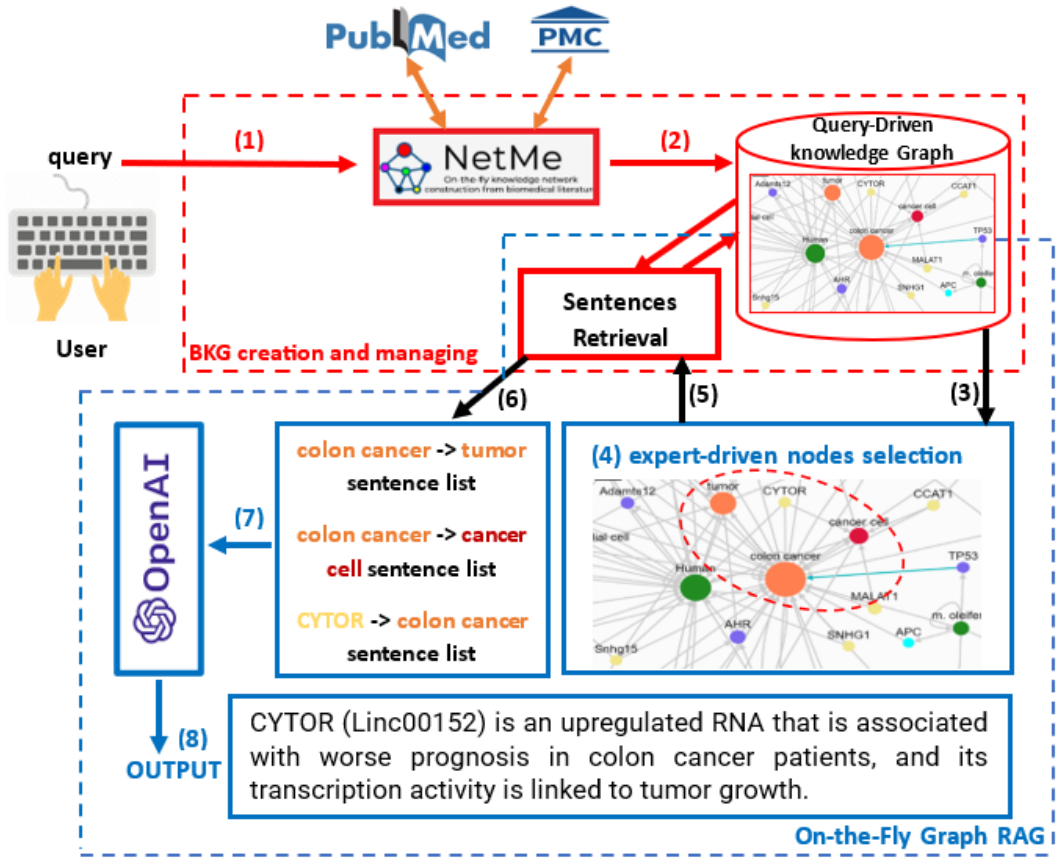


Figure 31: The algorithmic structure of the on-the-fly Graph RAG approach

composing (path-)edges; (iii) it then partitions the paths according to their scores in three ranges: $[0, 0.35[$, $[0.35, 0.7[$, and $[0.7, 1]$; and, finally, (iv) it selects the top paths from each such range. Of course, the most valuable range is the one with the top weight (i.e., $[0.7, 1]$). For this reason, the number of pulled edges is higher for higher-weight ranges and lower for lower-weight ones (in order to minimize the amount of noise). We decided to employ these ranges (in opposition to simply taking the top-k ones) to maximize variety, as we noticed that the top-k edges often shared the same small set of paths.

Figure 33 shows the sentence set (underlined in red and orange within the "Sentences" section) bracket with edges connecting "COVID-19" and "SLC16A1" nodes. Furthermore, in the "Articles" section, the user can find the document IDs where such sentences were extracted. The "Nodes" section, instead, lists the node set involved in the path between the "COVID-19" and "SLC16A1" nodes. When a user clicks on the "Start search" button, the sentence set is sent to OpenAI, which will return a summarized text. Each individual NetME graph page allows a user to submit up to five requests to OpenAI at most. If more are needed, then the users have to use their own API key. In such cases, the user's OpenAI requests do not involve NetME 2.0 back-end, ensuring the privacy of the API key, as required by OpenAI developers.

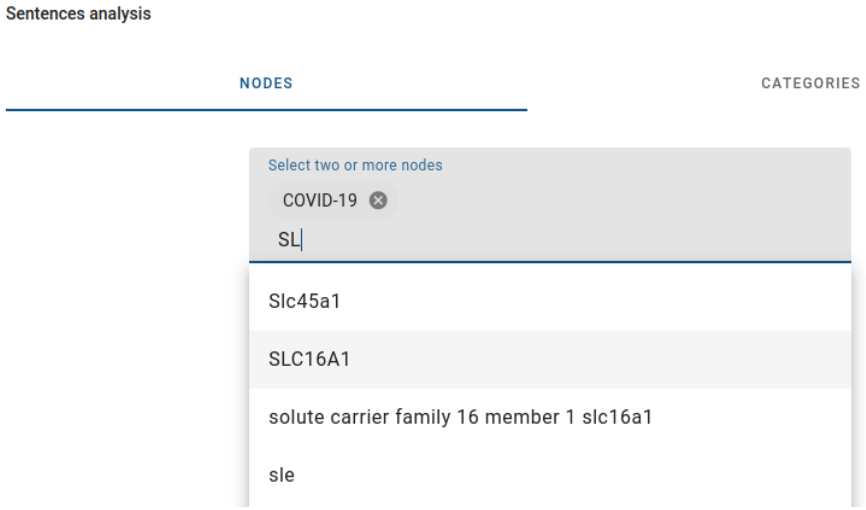


Figure 32: Nodes selection in order to generate a summarized text that talks about their associated entities: i.e., COVID-19 and SLC16A1

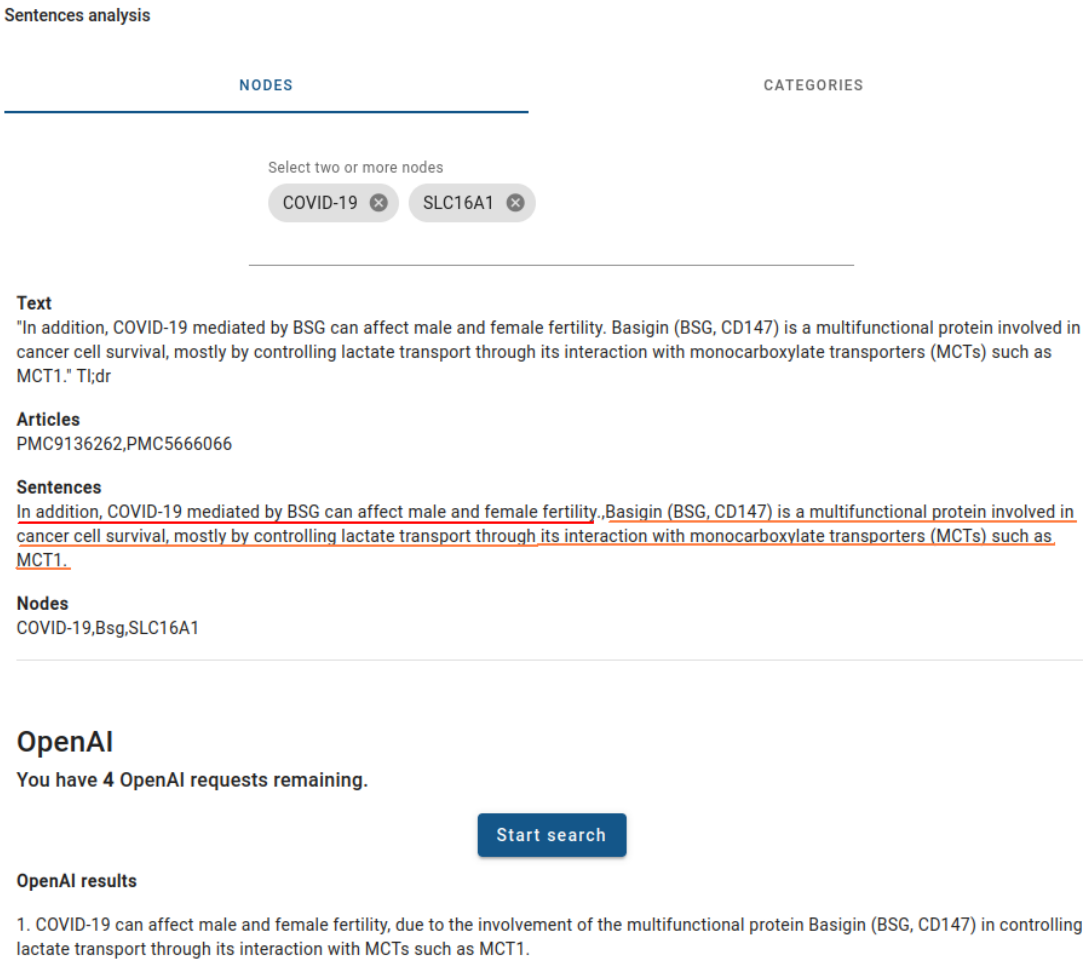


Figure 33: OpenAI summary example between nodes “COVID-19” and “SLC16A1”

5.5 NETME 2.0 - EVALUATION

In order to evaluate the quality of the networks built by NetME 2.0, we decided to focus on GDAs for three main reasons:

- *DisGeNET* offers a set of curated GDAs, attached to a set of relevant papers "testifying" that relation.
- *Genes* are the most complex entity types to model for NetME and OntoTagME. This is due to their many different names, aliases, and naming standards. More details on this are in Section 2.4.1.
- *GDAs* are usually considered as one of the most important biomedical relationship types.

Starting from these motivations, we designed the following case studies:

- The first one revolves around *guided* queries, and its goal is to measure the effectiveness of NetME in extracting knowledge *from a set of papers* given as input to NetME. This aims at looking at the ability of NetME to find the DisGeNET GDAs in those papers.
- The second one focuses on *open* queries, and its goal is to measure the effectiveness of NetME in extracting knowledge about some *genes* given in input to NetME, by looking at its ability to find the DisGeNET GDAs found by experts for those genes.
- The third one is done purely to show the improvement over NetME's first version. It is similar to the first case study but it focuses on 100 random GDAs for gene BSG.
- Finally, we compare, both qualitatively and quantitatively, the networks generated by NetME 2.0 and the set of biomedical KGs presented in Section 2.5.2.

We evaluated the quality of the retrieval process by computing the *recall* metric. This is due to the fact that NetME additionally finds GDAs that are not in the curated set. This is normal and expected, as for example, gene APP has 485 links on DisGeNET, but only 76 of those are curated.

5.5.1 *Case Study 1 - PubMed queries*

For the first case study, we selected three genes (i.e., BRCA1, APP, BRAF) with enough curated links in DisGeNET and with several related types of different diseases. We then took the full list of their associations on DisGeNET, and filtered out all those that do not comply with the following three filtering properties:

- The *Evidence Level* (EL_{gda}) must not be labeled as *limited* (associated to GDA with low certainty) or *disputed* (which means that a human reviewer labeled the GDA as wrong).

- The *Gene-Disease Association Score* ($\text{Score}_{\text{gda}}$) must not be below a variable threshold. This score is a value where $0 \leqslant \text{Score}_{\text{gda}} \leqslant 1$, where a bigger score means that the edge has stronger evidence in the literature. For our experiments, we fixed three thresholds (0.3, 0.4, 0.5), and we show how the results vary according to this.

- The *number of supporting PM IDs* for the GDA must not be less than three unless the evidence level for said GDA is *strong* or *definitive*.

These filters were added in order to remove potentially noisy or erroneous GDAs, and to guarantee at least a couple of PM IDs for NetME to be able to effectively retrieve this edge. The filter on the number of supporting PM IDs is very crucial. Most of the papers associated with a GDA either lack one of the two entities entirely or lack a phrase connecting them in any way. Additionally, many of the papers associated with the GDAs are not open access (roughly only one in four papers is open access).

Finally, we took all the remaining curated GDAs and merged the ones that belong to the same group (e.g., ankle reflex and knee reflex were all put under *reflex - generic*). This left us with 20 GDAs for gene APP, 10 for gene BRCA1, and 12 for gene BRAF (by setting the $\text{Score}_{\text{gda}}$ threshold as 0.3).

Then, for each GDA, we took the list of PM IDs deemed relevant for it by DisGeNET and built a network on those PM abstracts. If NetME was able to correctly find the edge, we would label that interaction as *found*; otherwise, we downloaded and annotated the full-texts of the available corresponding PMC IDs for all the open access ones. Finally, we built the network on these full-texts and checked whether we found the original GDA.

At this point, there are two possible outcomes:

- In the case of a correctly identified edge, we simply label it as *found* and track whether we found it within the abstracts or in the full-texts of PM papers.

- Otherwise, there might still be the situation that NetME did not have any *paragraph* in the downloaded full-texts where it would have been able to correctly retrieve that GDA. Therefore, we checked the list of PM papers associated with the missed gene-disease pairs to see whether these pairs were present in them: if present, these errors were called: *true misses*; otherwise *fake misses*, as NetME could not have been able to find that in any way.

For the purpose of the computation of the Recall metric, we count every miss (i.e., fake and true ones summed up). As shown in Table 11, our system is able to effectively capture most of the curated edges in DisGeNET. True Misses and Fake Misses are specified as TM and FM respectively.

More specifically only 2 out of 46 total edges are true misses by NetME (APP-Dementia and BRAF-LeopardSyndrome); 8 misses are not caused by NetME, but they come from the fact that the GDA did not transpire from the PM IDs put by researchers as evidence.

Going more in-depth for the three genes, and their respective mistakes, Tables 12, 13, 14 show the results regarding genes APP, BRAF, and BRCA1 respectively. We bolded all the edges missed by NetME in the table for ease of readability. For each curated edge in DisGeNET, we show the number of supporting PM IDs (it must

Gene	Threshold	Recall	Edges	Correct	TM	FM
BRCA1	$S_{gda} \geqslant 0.3$	0.714	14	10	0	4
	$S_{gda} \geqslant 0.4$	1	5	5	0	0
	$S_{gda} \geqslant 0.5$	1	5	5	0	0
APP	$S_{gda} \geqslant 0.3$	0.762	21	16	1	4
	$S_{gda} \geqslant 0.4$	0.8	10	8	1	1
	$S_{gda} \geqslant 0.5$	0.667	6	4	1	1
BRAF	$S_{gda} \geqslant 0.3$	0.833	12	10	1	1
	$S_{gda} \geqslant 0.4$	0.889	9	8	1	0
	$S_{gda} \geqslant 0.5$	0.889	9	8	1	0

Table 11: Case Study 1 - Metrics over the three genes

be noted that only a fraction of those have a free open access full-text paper available) and a column showing the result from NetME. The values shown for this column are:

- ABSTRACT: NetME found the edge on the abstracts network.
- FULL-TEXT: the abstract network was not enough, but NetME found the relationship on the full-text network.
- FAKE MISS: it means that we manually checked all the available abstracts and full-texts and there was no phrase or paragraph where such an association could be retrieved. This means that NetME had no way to retrieve this edge.
- TRUE MISS: this means that NetME did not find the edge.

Results for gene APP (Table 12) show that most of the mistakes happen with edges with a low PM support. Three of those mistakes were fake misses, with only *APP-dementia* being problematic to find per NetME. The number of full-texts where APP and dementia co-occur are 40 444, but this is mostly due to papers like [148, 149], that describe mobile and tablet applications to aid Dementia patients. This not only skews the number of co-occurrences up, but it also impacts the results of PM open queries, as they are often not coherent with the real query. Regardless, we also queried NetME for a match with a composite query to check if our system could be able to retrieve the GDA under better circumstances. Indeed, we obtain the missed edge when doing the composite query (syntax is *"(APP[All Fields]) AND (dementia[All Fields])"*) on the top 20 most relevant papers on PMC.

Inspecting the results of gene BRAF (Table 13), we can notice two misses. One is a fake miss (Syringocystadenoma Papilliferum), but NetME is not able to find *Leopard Syndrome*. This is likely because NetME sometimes labels the nodes as *"leopard (Specie)"*. Also in this case, like in the case of APP-Dementia from before, our tool can find the link BRAF-LeopardSyndrome using a composite query on PMC (syntax is *"(BRAF[All Fields]) AND (leopard Syndrome[All Fields]"*) on the top 20 results.

Disease	N. PM IDs	Output
Alzheimer's Disease	63	ABSTRACT
Familial Alzheimer Disease (FAD)	35	ABSTRACT
Presenile dementia	35	ABSTRACT
Acute Confusional Senile Dementia	35	ABSTRACT
Memory impairment	16	ABSTRACT
Memory Loss	16	ABSTRACT
Memory Disorders	16	ABSTRACT
Age-Related Memory Disorders	16	ABSTRACT
Nerve Degeneration	11	ABSTRACT
Adult Learning Disorders	8	ABSTRACT
Learning Disturbance	8	ABSTRACT
Learning Disabilities	8	FULL-TEXT
Learning Disorders	8	FULL-TEXT
Cerebral Amyloid Angiopathy	7	ABSTRACT
Paralysed	5	ABSTRACT
Mental Depression	**5**	**FAKE MISS**
Depressive Disorder	**5**	**FAKE MISS**
Todd Paralysis	**5**	**FAKE MISS**
Dementia	**3**	**TRUE MISS**
Cerebral hemorrhage	3	ABSTRACT
Cerebrovascular accident	1	ABSTRACT

Table 12: Case Study 1 - gene APP

Finally, on gene BRCA1, NetME can find all the links, except for those that are effectively impossible for the tool to retrieve.

5.5.2 *Case Study 2 - Open queries*

For the second case study, we focused on open queries about genes. We downloaded the TSV file including all *DisGeNET curated* GDAs updated up to the year 2022. This file contains 84 038 unique GDAs. We performed open queries on genes to check which of the edges referring to the queried one and present in DisGeNET were found by NetME. The open query relies on PMC, which effectively means that the quality of the network will be dependent on the quality of the top-k most relevant papers retrieved by NetME.

Opposite to the first case study, the above file does not offer the list of PM IDs associated with each GDA. Therefore we took two genes with a sufficient variety and more than 10 *associated diseases*: namely ABAT and CACNA1A. Then we made an

Disease	N. PM IDs	Output
Melanoma	24	ABSTRACT
Cardio-facio-cutaneous syndrome	14	FULL-TEXT
Papillary thyroid carcinoma	8	ABSTRACT
Noonan Syndrome	8	FULL-TEXT
Colorectal Carcinoma	8	FULL-TEXT
Thyroid Neoplasm	6	ABSTRACT
Leopard Syndrome	**6**	**TRUE MISS**
Thyroid Gland Follicular Adenoma	6	ABSTRACT
Noonan Syndrome 7	5	ABSTRACT
Colorectal Neoplasms	4	FULL-TEXT
Pilomyxoid astrocytoma	3	ABSTRACT
Syringocystadenoma Papilliferum	**1**	**FAKE MISS**

Table 13: Case Study 1 - gene BRAF

Disease	N. PM IDs	Output
Malignant neoplasm of breast	34	ABSTRACT
Breast-ovarian cancer	16	ABSTRACT
Breast Cancer	12	ABSTRACT
Mammary Neoplasms	12	ABSTRACT
Ovarian Cancer	10	ABSTRACT
Malignant neoplasm of ovary	9	ABSTRACT
Fanconi Anemia	7	ABSTRACT
Hereditary Breast and Ovarian Cancer	5	ABSTRACT
Mental Depression	5	ABSTRACT
Depressive disorder	5	ABSTRACT
Seizures	**1**	**FAKE MISS**
Ischemic stroke	**1**	**FAKE MISS**
Brain hemorrhage	**1**	**FAKE MISS**
Leukoencephalopathy	**1**	**FAKE MISS**

Table 14: Case Study 1 - gene BRCA1

open query for each of the two genes on NetME and generated a network on the top-50 most relevant full-texts for said query according to PMC.

Finally, we took all the GDAs found by NetME for those two genes. This left us with 72 GDAs for gene ABAT and 59 for gene CACNA1A. We finally ranked the results based on the edge weight assigned by NetME and evaluated the overall Recall metric and the one at the first 20 results.

Gene	Recall	Edges	Correct	Correct @ top-20
ABAT	0.769	13	10	9
CACNA1A	0.583	12	7	5

Table 15: Case Study 2 - Recall

Table 15 shows the metrics obtained by NetME on the genes ABAT and CACNA1A. As we can see, the results (Recall-wise) are slightly worse than in the first case study, and that is expected, as this example is less guided and thus more complex for NetME.
The detailed results for genes ABAT and CACNA1A are shown in Tables 16 and 17 respectively.

Disease / Disorder	Rank
Autistic Disorder	TOP 20
Gastroesophageal Reflux	TOP 20
Seizure	TOP 20
Letargy	TOP 20
Liver Cirrhosis	**FAKE MISS**
Hypotonia	TOP 20
Hyperreflexia	TOP 20
Hyporeflexia	TOP 20
Psychomotor Disorders	TOP 20
Epileptic drop attack	TOP 40
GABA transaminase deficiency	TOP 20
Developmental delay	**FAKE MISS**
Convulsions	**FAKE MISS**

Table 16: Case Study 2 - gene ABAT

Regarding gene ABAT, NetME was able to correctly identify 10 out of 13 GDAs, with only one out of 10 not being present in the top 20 most relevant ones according to the weight score. We then checked all the 3 highlighted misses by looking at the abstracts and full-texts in PM and PMC where those diseases and the gene ABAT co-occurred. More specifically:

- *ABAT - liver cirrhosis* co-occurred in 6 full-text papers, and never in abstracts.
- *ABAT - developmental delay* co-occurred in 3 full-text papers, and never in abstracts.
- *ABAT - convulsions* co-occurred in 10 full-text papers, and never in abstracts.

We manually checked those papers and found no paragraphs where those GDAs could be extracted, thus rendering those misses *fake* ones.

Disease / Disorder	Rank
Akinetic petict	TOP 110
Ataxia	TOP 5
Bipolar Disorder	**TRUE MISS**
Dystonia	TOP 10
Epilepsy	TOP 20
Exfoliation Syndrome	**TRUE MISS**
Alternating hemiplegia of childhood	TOP 5
Abnormal Coordination	**TRUE MISS**
Tremor	**TRUE MISS**
Hemiplegic migraine	TOP 5
Paroxysmal Torticollis	**TRUE MISS**
Absence Seizure Disorder	TOP 110

Table 17: Case Study 2 - gene CACNA1A

For the gene CACNA1A, NetME found 7 out of 12 edges. Out of the 7 correct ones, 2 were not in the top 20 results, while 5 were. This is by far the worst result obtained by NetME thus far. We then checked whether a better PM search would be able to aid in the extraction of the associations. Indeed, we performed 5 different composite queries, and in each of those, NetME was able to find all 5 missing GDAs. This shows how much of the network extraction process depends on the quality of the recommendation algorithm of PM and PMC.

5.5.3 *Case Study 3 - 100 random DisGeNET associations*

In the third case study, we replicated the test done for NetME 1.0 in Section 5.3.3.

We took the same set of 100 random GDAs from DisGeNET regarding gene BSG. Those associations have a pointer to a set of papers where the association was retrieved, and their cardinality varies from a minimum of 1 to a maximum of 82 papers. Then, for each of the 100 associations, we built a NetME network on the abstracts of those papers provided by DisGeNET, and we considered the edge as a true positive if we were able to find the said edge in the target network.

Table 18 shows the performance of our two systems, where it is clear that NetME 2.0 obtains a significant increase in true positive rate, which proves the strength of the new back-end we have developed.

	Correct	Missed
NetME 1.0	63	37
NetME 2.0	87	13

Table 18: Case Study 3 - Metrics

5.5.4 *Evaluating NetME against other known biomedical KGs*

We are left with an evaluation of NetME against the best known biomedical KGs, which we described in Section 2.5.2, and whose main characteristics are summarized in Table 2.

NetME is not a *static* (pre-computed) biomedical KG, as the many other ones. However, we consider those other static KGs because they can answer the same user queries as NetME via a sub-graph extraction step. Say, for example, a user wants to build the interaction network of gene BRCA1. NetME dynamically queries PM and PMC to retrieve up-to-date information about this gene and then builds the network for it on the fly. The static KGs may dynamically extract a suitable sub-graph that includes BRCA1 and some of its neighborhood nodes and edges.

In the following list, we describe in detail the main differences between NetME and all the other biomedical KGs, highlighting the strong points in favor of its design.

- *Pre-computation*: The most immediate difference is in the fact that static KGs are pre-computed offline, while NetME needs to construct a network from scratch for each query. Of course, the running time of a *sub-graph* extraction operation is smaller than the time required to mine papers and build a KG, as NetME does, which is a clear con for NetME. But, on the flip side, keeping a large KG up-to-date requires large-scale update operations, which NetME avoids by guaranteeing immediate access to the most recent publications on PM and PMC.

- *Full-text support*: At the time of writing this thesis, NetME is the only biomedical knowledge network builder able to work with PMC full-texts.

- *Strong categorization capabilities*: NetME is the only tool that provides two layers of categorization. Macro-categories correspond to the main types of bio-entities we model. On top of that, NetME proposes another set of categories that are either sub-categories of the previously mentioned macro-categories or offer other qualitative information. These categories can be handy at query time (e.g., find the diseases that share most connections with "tumor marker" gene nodes).

- *Weighted and/or labeled edges*: In the context of KGs, edges can be weighted and/or labeled. Out of the two options, labeled edges are considerably more powerful, because they carry valuable information on the specific type of relationship between the two connected bio-entities. Saying that one entity "up-regulates" another is vastly more interesting than just saying they are related, even if a weight is attached to it. All the discussed KGs, including NetME, offer labeled edges except for DARLING [21] and BioTagME [8]. NetME's edges, additionally, are both labeled and weighted, where the weights were discussed in Section 5.2.

- *Number of nodes/edges*: One of the main differences between KGs is in their number of nodes and edges (and their labels). However, having a large number of nodes/edges does not imply that the KG is of greater quality. Let's take *BIOS* [104] as an example: Its latest release (i.e., V2.2) contains about 27M nodes, but about 28% of them do not have a category attached; and out of 70M edges, about 69M have the label "is a", whereas most of the remaining edges use verbs

that solely relate to associations with drugs. This means that BIOS may work well only when dealing with chemicals and drugs.

For other quantitative and qualitative information regarding the analyzed KGs, and NetME, we refer the reader to Table 2.

The last thing to discuss is related to the accuracy performance of NetME and the other biomedical KGs. For this purpose, we performed a test along the lines of Case Study 3 shown in the previous Section 5.5.3. The test consisted of 100 random GDAs of gene "BSG" from DisGeNET, and their associated PM IDs. Now, since the various KGs builders work differently we needed to design three different experiments.

Since DARLING builds a co-occurrence network between entities in papers, we evaluated its accuracy by taking the abstracts of the PMs containing the term "BSG" (they were 697), building the co-occurrence network, and then checking how many of the 100 randomly chosen GDAs were found among the network edges (with no filtering). For the other (static) KGs, we took the neighboring diseases of node BSG and counted how many GDAs out of the 100 randomly sampled ones were caught. The final test on NetME was implemented by building one network per GDA, using the PM IDs provided by DisGeNET and associated with the checked GDA. We counted the number of edges that were found overall out of the 100 randomly sampled ones, exactly as in Case Study 3.

The results of this test are shown in Table 19.

Type	Tool	Correct	Details
on-the-fly with labeled (and weighted) edges	NetME 1.0 [this thesis]	63	PMC mined edges (full-text)
	NetME 2.0 [this thesis]	**87**	PMC mined edges (full-text)
precomputed with labeled (and weighted) edges	BIOS [104]	0	PM mined edges
	SPOKE [101]	0	ontological edges
	BioKG [22]	48	ontological edges
	Hetionet [100]	2	ontological edges
precomputed with weighted edges	DARLING [21]	85	co-occurrences in PM abstracts
	BioTagME [this thesis]	65	mix of PM mined and ontological edges

Table 19: NetME's performance comparison with other static bio-KGs. The "Correct" column shows the number of correctly identified BSG-disease associations by the tool out of the 100 randomly sampled ones from DisGeNET. Details are provided in the text

We notice that Clinical KG [107], RTX-KG2 [108] and the KG of Xu et al. [110] are not present in the table. For the first two, the reason is that their KGs use DisGeNET as their ontology, so the comparison would be unfair. On the other hand, we discarded

the KG of Xu et al. [110] because it does not provide any relation between bio-entities, since it focuses on affiliations and funding.

As Table 19 clearly shows, NetME 2.0 is the best performer, the second one is DARLING, which however links entities based only on their co-occurrence in abstracts, thus providing no "explanation" about KG's edges. Conversely, the closest tool offering similar features to NetME's KG (i.e., BioKG [22]) is worse in Recall by an absolute 39%.

6 CONCLUSION

In the rapidly evolving landscape of Natural Language Processing, Knowledge Graphs, and Artificial Intelligence, this thesis has focused on the challenges and opportunities raised by the biomedical domain. In particular, the main driving motivation was the impressive growth of the publishing rate of papers in biomedical repositories such as PM and PMC, making it impossible for researchers to be fully up-to-date with the most recent scientific discoveries for their studies. This has highlighted the critical need for computational tools that can scan through these vast paper repositories, extract relevant knowledge, and represent it meaningfully.

The goal of this thesis has been therefore to design and implement robust platforms, namely BioTagME and NetME, aimed at constructing comprehensive and accurate KGs tailored to biomedical data.

The design of BioTagME focused on building a large KG integrating biomedical ontologies and knowledge mined from PM abstracts. This latter was extracted by using TagME, a general-purpose entity linker, to find entities in the input text, and then connected them to form a graph based on the frequency of their co-occurrence. Then, to improve the performance of the entity linker, we designed OntoTagME, which builds on TagME and deploys a biomedical view of Wikidata as a back-end KB. We used OntoTagME as the entity linker for NetME, the first biomedical network generator that works on the fly on full-text papers. NetME integrates OntoTagME and state-of-the-art edge extraction modules to create high-quality networks based on user queries.

The results obtained by the evaluation of the KBs built by BioTagME and NetME against established benchmarks, curated datasets, and other known KBs-generators have been significant. With high sensitivity and specificity metrics, these platforms have demonstrated their efficacy in capturing intricate relationships among biomedical entities. Moreover, the enhancements introduced in NetME 2.0, including the integration of cutting-edge NLP techniques and the Graph RAG engine, have paved the way for some future research topics that we consider very interesting:

- NetME uses TagME as a back-end entity linker. However, for full-text processing applications, the University of Pisa has developed SWAT [55], whose focus on longer, more well-formed texts might better fit the needs of annotating long scientific papers. However, adapting SWAT to the biomedical field would require a significant re-adaptation of its architecture, along with the creation of a training dataset on biomedical papers.

- OntoTagME 1.0 successfully integrated a large set of biomedical ontologies, while OntoTagME 2.0 switched from ontologies to filtering the biomedical entities present in Wikidata. It would be interesting to evaluate a hybrid approach in which the large information available in Wikidata is supplemented with ontological knowledge.

- The Case Study 2 commented in Section 5.5.2 showed how much the PM ranking function impacts the networks built by NetME. Ebeid [150] has recently

proposed a new ranking function for PM papers that seems to perform better than the default one. It would be therefore interesting to evaluate its impact in the formation of NetME's KG.

- The final and most ambitious future work is, in our opinion, to apply NetME not just on a bunch of papers selected on the fly from PM or PMC repositories but apply it on their whole set of open-access papers which, at the time of writing this thesis, surpasses the 9M documents. This would create the first very-large biomedical KG constructed on the full text of PMC's papers, with an information richness and quality superior to the other static KGs, as we commented in Table 19.

A

APPENDIX

A.1 RELEVANT LINKS

ONTOTAGME

- REST API endpoint: https://ontotagme-sobigdata.d4science.org/
- REST API docs: https://sobigdata.d4science.org/web/tagme/ontotagme-api
- Repository: https://github.com/LorenzoBellomo/OntoTagME-SoBigData
- Image on Docker Hub: lorenzobellomosns/ontotagme-sobigdata:latest

BIOTAGME

- Web Application: https://biotagme.eu
- Full Network, Zenodo Download: https://zenodo.org/record/6325345
- Back-end Repository: https://github.com/knowmics-lab/biotagme_pipeline
- Front-End Repository: https://github.com/Anto188bas/biotagme_laravel

NETME

- Web Application: https://netme.click
- Repository: https://github.com/LorenzoBellomo/NetME-SoBigData

TOPICALTAGME

Repository: https://github.com/LorenzoBellomo/TopicalTagME

BIOLOGICAL WIKIDATA EXTRACTOR

Repository: https://github.com/LorenzoBellomo/simple-wikidata-db
(forked from https://github.com/neelguha/simple-wikidata-db)

BIBLIOGRAPHY

[1] Rita González-Márquez, Luca Schmidt, Benjamin M. Schmidt, Philipp Berens, and Dmitry Kobak. "The landscape of biomedical research." In: *bioRxiv* (2023). DOI: 10.1101/2023.04.10.536208. eprint: https://www.biorxiv.org/content/early/2023/05/25/2023.04.10.536208.full.pdf. URL: https://www.biorxiv.org/content/early/2023/05/25/2023.04.10.536208.

[2] Will Douglas Heaven. *AI is dreaming up drugs that no one has ever seen. Now we've got to see if they work.* https://www.technologyreview.com/2023/02/15/1067904/ai-automation-drug-development/. Feb. 2023.

[3] Gartner. *Market Guide for Graph Database Management Systems*. https://www.gartner.com/en/documents/4018220. Aug. 2023.

[4] L. M. Schriml et al. "Human Disease Ontology 2018 update: classification, content and workflow expansion." In: *Nucleic Acids Research* 47.D1 (Nov. 2018), pp. D955–D962. DOI: 10.1093/nar/gky1032. URL: https://doi.org/10.1093/nar/gky1032.

[5] David S Wishart et al. "PathBank: a comprehensive pathway database for model organisms." In: *Nucleic Acids Research* 48.D1 (Oct. 2019), pp. D470–D478. ISSN: 0305-1048. DOI: 10.1093/nar/gkz861. eprint: https://academic.oup.com/nar/article-pdf/48/D1/D470/31697631/gkz861.pdf. URL: https://doi.org/10.1093/nar/gkz861.

[6] D. S. Wishart et al. "DrugBank 5.0: a major update to the DrugBank database for 2018." In: *Nucleic Acids Research* 46.D1 (Nov. 2017), pp. D1074–D1082. DOI: 10.1093/nar/gkx1037. URL: https://doi.org/10.1093/nar/gkx1037.

[7] Paolo Ferragina and Ugo Scaiella. "Fast and Accurate Annotation of Short Texts with Wikipedia Pages." In: *IEEE Software* 29.1 (2012), pp. 70–75. DOI: 10.1109/MS.2011.122.

[8] Antonio Di Maria, Salvatore Alaimo, Lorenzo Bellomo, Fabrizio Billeci, Paolo Ferragina, Alfredo Ferro, and Alfredo Pulvirenti. "BioTAGME: A Comprehensive Platform for Biological Knowledge Network Analysis." In: *Frontiers in Genetics* 13 (2022). ISSN: 1664-8021. DOI: 10.3389/fgene.2022.855739. URL: https://www.frontiersin.org/articles/10.3389/fgene.2022.855739.

[9] Antonio Di Maria, Salvatore Alaimo, Lorenzo Bellomo, Fabrizio Billeci, Paolo Ferragina, Alfredo Ferro, and Alfredo Pulvirenti. *BioTAGME: A comprehensive platform for biological knowledge network analysis*. en. 2022. DOI: 10.5281/ZENODO.6325345. URL: https://zenodo.org/record/6325345.

[10] Alessandro Muscolino, Antonio Di Maria, Rosaria Valentina Rapicavoli, Salvatore Alaimo, Lorenzo Bellomo, Fabrizio Billeci, Stefano Borzì, Paolo Ferragina, Alfredo Ferro, and Alfredo Pulvirenti. "NETME: on-the-fly knowledge network construction from biomedical literature." In: *Applied Network Science* 7.1 (Jan. 2022), p. 1. ISSN: 2364-8228. DOI: 10.1007/s41109-021-00435-x. URL: https://doi.org/10.1007/s41109-021-00435-x.

[11] Gene Ontology Consortium. "The Gene Ontology (GO) database and informatics resource." In: *Nucleic Acids Research* 32.90001 (Jan. 2004), pp. 258D–261. DOI: 10.1093/nar/gkh036. URL: https://doi.org/10.1093/nar/gkh036.

[12] V. Petri et al. "The pathway ontology – updates and applications." In: *Journal of Biomedical Semantics* 5.1 (2014), p. 7. DOI: 10.1186/2041-1480-5-7. URL: https://doi.org/10.1186/2041-1480-5-7.

[13] Antonio Di Maria, Lorenzo Bellomo, Fabrizio Billeci, Alfio Cardillo, Salvatore Alaimo, Paolo Ferragina, Alfredo Ferro, and Alfredo Pulvirenti. "NetMe 2.0: A web-based platform for extracting and modeling knowledge as labeled graphs from biomedical literature." In: *Bioinformatics* (Submitted).

[14] C. H. Wei, H. Y. Kao, and Z. Lu. "PubTator: a web-based text mining tool for assisting biocuration." In: *Nucleic Acids Res* 41.Web Server issue (July 2013), W518–522.

[15] Chih-Hsuan Wei, Alexis Allot, Robert Leaman, and Zhiyong Lu. "PubTator central: automated concept annotation for biomedical full text articles." In: *Nucleic Acids Research* 47.W1 (May 2019), W587–W593. ISSN: 0305-1048. DOI: 10.1093/nar/gkz389. eprint: https://academic.oup.com/nar/article-pdf/47/W1/W587/28880193/gkz389.pdf. URL: https://doi.org/10.1093/nar/gkz389.

[16] Damian Szklarczyk et al. "The STRING database in 2017: quality-controlled protein-protein association networks, made broadly accessible." In: *Nucleic acids research* 45 (Oct. 2016). DOI: 10.1093/nar/gkw937.

[17] Janet Piñero, Juan Manuel Ramírez-Anguita, Josep Saüch-Pitarch, Francesco Ronzano, Emilio Centeno, Ferran Sanz, and Laura I Furlong. "The DisGeNET knowledge platform for disease genomics: 2019 update." In: *Nucleic Acids Research* 48.D1 (Nov. 2019), pp. D845–D855. ISSN: 0305-1048. DOI: 10.1093/nar/gkz1021. eprint: https://academic.oup.com/nar/article-pdf/48/D1/D845/31697865/gkz1021.pdf. URL: https://doi.org/10.1093/nar/gkz1021.

[18] Deng Cai, Yan Wang, Lemao Liu, and Shuming Shi. "Recent Advances in Retrieval-Augmented Text Generation." In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '22. 2022, 3417–3419. DOI: 10.1145/3477495.3532682.

[19] Huayang Li, Yixuan Su, Deng Cai, Yan Wang, and Lemao Liu. "A Survey on Retrieval-Augmented Text Generation." In: *CoRR* abs/2202.01110 (2022). arXiv: 2202.01110. URL: https://arxiv.org/abs/2202.01110.

[20] OpenAI. *GPT-4 Technical Report*. 2023. arXiv: 2303.08774 [cs.CL].

[21] Evangelos Karatzas, Fotis A. Baltoumas, Ioannis Kasionis, Despina Sanoudou, Aristides G. Eliopoulos, Theodosios Theodosiou, Ioannis Iliopoulos, and Georgios A. Pavlopoulos. "Darling: A Web Application for Detecting Disease-Related Biomedical Entity Associations with Literature Mining." In: *Biomolecules* 12.4 (2022). ISSN: 2218-273X. DOI: 10.3390/biom12040520. URL: https://www.mdpi.com/2218-273X/12/4/520.

[22] Brian Walsh, Sameh K. Mohamed, and Vít Nováček. "BioKG: A Knowledge Graph for Relational Learning On Biological Data." In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. CIKM '20. Virtual Event, Ireland: Association for Computing Machinery, 2020, 3173–3180. ISBN: 9781450368599. DOI: 10.1145/3340531.3412776. URL: https://doi.org/10.1145/3340531.3412776.

[23] Aidan Hogan et al. "Knowledge Graphs." In: *ACM Comput. Surv.* 54.4 (July 2021). ISSN: 0360-0300. DOI: 10.1145/3447772. URL: https://doi.org/10.1145/3447772.

[24] Google. *Introducing the Knowledge Graph: things, not strings*. https://blog.google/products/search/introducing-knowledge-graph-things-not/. 2012.

[25] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report 1999-66. Previous number = SIDL-WP-1999-0120. Stanford InfoLab, Nov. 1999. URL: http://ilpubs.stanford.edu:8090/422/.

[26] S. Alaimo, A. Pulvirenti, R. Giugno, and A. Ferro. "Drug-target interaction prediction through domain-tuned network-based inference." In: *Bioinformatics* 29.16 (Aug. 2013), pp. 2004–2008.

[27] Xiaohua Zhou, Xiaodan Zhang, and Xiaohua Hu. "MaxMatcher: Biological Concept Extraction Using Approximate Dictionary Lookup." In: *PRICAI 2006: Trends in Artificial Intelligence*. Ed. by Qiang Yang and Geoff Webb. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1145–1149. ISBN: 978-3-540-36668-3.

[28] DB-Engines. *DB-Engines Ranking of Graph DBMS*. https://db-engines.com/en/ranking/graph+dbms. Nov. 2023.

[29] Neo4j. *Neo4j - The World's Leading Graph Database*. 2012. URL: http://neo4j.org/.

[30] Neo4j. *Cypher Introduction Manual*. https://neo4j.com/docs/cypher-manual/current/introduction/.

[31] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL].

[32] Gerard Salton and Michael J McGill. "Introduction to modern information retrieval." In: (1986).

[33] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd. *spaCy: Industrial-strength Natural Language Processing in Python*. 2020. DOI: 10.5281/zenodo.1212303. URL: https://doi.org/10.5281/zenodo.1212303.

[34] Matthew Honnibal and Mark Johnson. "An Improved Non-monotonic Transition System for Dependency Parsing." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, 2015, pp. 1373–1378. DOI: 10.18653/v1/D15-1162. URL: https://www.aclweb.org/anthology/D15-1162.

[35] Ikuya Yamada, Akari Asai, Jin Sakuma, Hiroyuki Shindo, Hideaki Takeda, Yoshiyasu Takefuji, and Yuji Matsumoto. "Wikipedia2Vec: An Efficient Toolkit for Learning and Visualizing the Embeddings of Words and Entities from Wikipedia." In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 2020, pp. 23–30.

[36] E. Asgari and M. R. Mofrad. "Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics." In: *PLoS One* 10.11 (2015), e0141287.

[37] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[38] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. "Improving language understanding by generative pre-training." In: (2018).

[39] Sosuke Kobayashi. *Homemade BookCorpus*. https://github.com/soskek/bookcorpus. 2018.

[40] The New York Times Cade Metz. *Chatbots May 'Hallucinate' More Often Than Many Realize*. https://www.nytimes.com/2023/11/06/technology/chatbots-hallucination-rates.html. 2023.

[41] Sébastien Bubeck et al. *Sparks of Artificial General Intelligence: Early experiments with GPT-4*. 2023. arXiv: 2303.12712 [cs.CL].

[42] Hugo Touvron et al. "Llama 2: Open Foundation and Fine-Tuned Chat Models." In: abs/2307.09288 (2023). arXiv: 2307.09288. URL: https://arxiv.org/abs/2307.09288.

[43] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung-Yeung Shum, and Jian Guo. "Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph." In: *CoRR* abs/2307.07697 (2023). arXiv: 2307.07697. URL: https://arxiv.org/abs/2307.07697.

[44] NLP-progress. *Entity Linking - NLP-progress*. http://nlpprogress.com/english/entity_linking.html. 2023.

[45] Ruben Verborgh, Michael Röder, Ricardo Usbeck, and Axel-Cyrille Ngonga Ngomo. "GERBIL – Benchmarking Named Entity Recognition and Linking Consistently." In: *Semant. Web* 9.5 (Jan. 2018), 605–625. ISSN: 1570-0844. DOI: 10.3233/SW-170286. URL: https://doi.org/10.3233/SW-170286.

[46] Johannes M. van Hulst, Faegheh Hasibi, Koen Dercksen, Krisztian Balog, and Arjen P. de Vries. "REL: An Entity Linker Standing on the Shoulders of Giants." In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '20. New York, NY, USA: Association for Computing Machinery, 2020, 2197–2200. ISBN: 9781450380164. DOI: 10.1145/3397271.3401416. URL: https://doi.org/10.1145/3397271.3401416.

[47] Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. "End-to-End Neural Entity Linking." In: *Proceedings of the 22nd Conference on Computational Natural Language Learning*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 519–529. DOI: 10.18653/v1/K18-1050. URL: https://aclanthology.org/K18-1050.

[48] Manoj Prabhakar Kannan Ravi, Kuldeep Singh, Isaiah Onando Mulang', Saeedeh Shekarpour, Johannes Hoffart, and Jens Lehmann. "CHOLAN: A Modular Approach for Neural Entity Linking on Wikipedia and Wikidata." In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Ed. by Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty. Online: Association for Computational Linguistics, Apr. 2021, pp. 504–514. DOI: 10.18653/v1/2021.eacl-main.40. URL: https://aclanthology.org/2021.eacl-main.40.

[49] Francesco Piccinno and Paolo Ferragina. "From TagME to WAT: A New Entity Annotator." In: *Proceedings of the First International Workshop on Entity Recognition & Disambiguation*. ERD '14. Gold Coast, Queensland, Australia: Association for Computing Machinery, 2014, 55–62. ISBN: 9781450330237. DOI: 10.1145/2633211.2634350. URL: https://doi.org/10.1145/2633211.2634350.

[50] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. "Robust Disambiguation of Named Entities in Text." In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Ed. by Regina Barzilay and Mark Johnson. Edinburgh, Scotland, UK.: Association for Computational Linguistics, July 2011, pp. 782–792. URL: https://aclanthology.org/D11-1072.

[51] Ian Witten and David Milne. "An Effective, Low-Cost Measure of Semantic Relatedness Obtained from Wikipedia Links." In: (June 2010).

[52] Olena Medelyan, David Milne, Catherine Legg, and Ian H. Witten. "Mining meaning from Wikipedia." In: *International Journal of Human-Computer Studies* 67.9 (2009), pp. 716–754. ISSN: 1071-5819. DOI: https://doi.org/10.1016/j.ijhcs.2009.05.004. URL: https://www.sciencedirect.com/science/article/pii/S1071581909000561.

[53] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. "FLAIR: An easy-to-use framework for state-of-the-art NLP." In: *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. 2019, pp. 54–59.

[54] Octavian-Eugen Ganea and Thomas Hofmann. "Deep Joint Entity Disambiguation with Local Neural Attention." In: *CoRR* abs/1704.04920 (2017). arXiv: 1704.04920. URL: http://arxiv.org/abs/1704.04920.

[55] Marco Ponza, Paolo Ferragina, and Francesco Piccinno. "Swat: A system for detecting salient Wikipedia entities in texts." In: *Comput. Intell.* 35.4 (2019), pp. 858–890. DOI: 10.1111/coin.12216. URL: https://doi.org/10.1111/coin.12216.

[56] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. "The Stanford CoreNLP Natural Language Processing Toolkit." In: *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Ed. by Kalina Bontcheva and Jingbo Zhu. Baltimore, Maryland: Association for Computational Linguistics, June 2014, pp. 55–60. DOI: 10.3115/v1/P14-5010. URL: https://aclanthology.org/P14-5010.

[57] Rada Mihalcea and Paul Tarau. "TextRank: Bringing Order into Text." In: *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Ed. by Dekang Lin and Dekai Wu. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 404–411. URL: https://aclanthology.org/W04-3252.

[58] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. "DeepWalk: Online Learning of Social Representations." In: *CoRR* abs/1403.6652 (2014). arXiv: 1403.6652. URL: http://arxiv.org/abs/1403.6652.

[59] Mohamed Ali Hadj Taieb, Mohamed Ben Aouicha, and Abdelmajid Ben Hamadou. "Computing semantic relatedness using Wikipedia features." In: *Knowledge-Based Systems* 50 (2013), pp. 260–278. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2013.06.015. URL: https://www.sciencedirect.com/science/article/pii/S0950705113001913.

[60] Adam Pauls and Dan Klein. "Faster and Smaller N-Gram Language Models." In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Ed. by Dekang Lin, Yuji Matsumoto, and Rada Mihalcea. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 258–267. URL: https://aclanthology.org/P11-1027.

[61] Matthew Hoffman, Francis Bach, and David Blei. "Online Learning for Latent Dirichlet Allocation." In: *Advances in Neural Information Processing Systems*. Ed. by J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta. Vol. 23. Curran Associates, Inc., 2010. URL: https://proceedings.neurips.cc/paper_files/paper/2010/file/71f6278d140af599e06ad9bf1ba03cb0-Paper.pdf.

[62] Marco Ponza, Paolo Ferragina, and Soumen Chakrabarti. "A Two-Stage Framework for Computing Entity Relatedness in Wikipedia." In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. CIKM '17. New York, NY, USA: Association for Computing Machinery, 2017, 1867–1876. ISBN: 9781450349185. DOI: 10.1145/3132847.3132890. URL: https://doi.org/10.1145/3132847.3132890.

[63] Taher H. Haveliwala. "Topic-Sensitive PageRank." In: *Proceedings of the 11th International Conference on World Wide Web*. WWW '02. New York, NY, USA: Association for Computing Machinery, 2002, 517–526. ISBN: 1581134495. DOI: 10.1145/511446.511513. URL: https://doi.org/10.1145/511446.511513.

[64] Sascha Rothe and Hinrich Schütze. "CoSimRank: A Flexible & Efficient Graph-Theoretic Similarity Measure." In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Kristina Toutanova and Hua Wu. Baltimore, Maryland: Association for Computational

Linguistics, June 2014, pp. 1392–1402. DOI: 10.3115/v1/P14-1131. URL: https://aclanthology.org/P14-1131.

[65] Eric Yeh, Daniel Ramage, Christopher D. Manning, Eneko Agirre, and Aitor Soroa. "WikiWalk: Random walks on Wikipedia for Semantic Relatedness." In: *Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing (TextGraphs-4)*. Ed. by Monojit Choudhury, Samer Hassan, Animesh Mukherjee, and Smaranda Muresan. Suntec, Singapore: Association for Computational Linguistics, Aug. 2009, pp. 41–49. URL: https://aclanthology.org/W09-3206.

[66] F. Fouss, A. Pirotte, and M. Saerens. "A novel way of computing similarities between nodes of a graph, with application to collaborative recommendation." In: *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*. 2005, pp. 550–556. DOI: 10.1109/WI.2005.9.

[67] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. "LINE: Large-scale Information Network Embedding." In: *Proceedings of the 24th International Conference on World Wide Web*. WWW '15. International World Wide Web Conferences Steering Committee, May 2015. DOI: 10.1145/2736277.2741093. URL: http://dx.doi.org/10.1145/2736277.2741093.

[68] Wenhao Gu, Xiao Yang, Minhao Yang, Kun Han, Wenying Pan, and Zexuan Zhu. "MarkerGenie: an NLP-enabled text-mining system for biomedical entity relation extraction." In: *Bioinformatics Advances* 2.1 (May 2022), vbac035. ISSN: 2635-0041. DOI: 10.1093/bioadv/vbac035. eprint: https://academic.oup.com/bioinformaticsadvances/article-pdf/2/1/vbac035/47084031/vbac035.pdf. URL: https://doi.org/10.1093/bioadv/vbac035.

[69] J. G. Zheng, D. Howsmon, B. Zhang, J. Hahn, D. McGuinness, J. Hendler, and H. Ji. "Entity linking for biomedical literature." In: *BMC Med Inform Decis Mak* 15 Suppl 1.Suppl 1 (2015), S4.

[70] Mamatjan Abdurxit, Turdi Tohti, and Askar Hamdulla. "An Efficient Method for Biomedical Entity Linking Based on Inter- and Intra-Entity Attention." In: *Applied Sciences* 12.6 (2022). ISSN: 2076-3417. DOI: 10.3390/app12063191. URL: https://www.mdpi.com/2076-3417/12/6/3191.

[71] S. Sohn, D. C. Comeau, W. Kim, and W. J. Wilbur. "Abbreviation definition identification based on automatic precision estimates." In: *BMC Bioinformatics* 9 (Sept. 2008), p. 402.

[72] Lihu Chen, Gaël Varoquaux, and Fabian M. Suchanek. "A Lightweight Neural Model for Biomedical Entity Linking." In: *AAAI Conference on Artificial Intelligence*. 2020. URL: https://api.semanticscholar.org/CorpusID:229221320.

[73] K. A. Gray, R. L. Seal, S. Tweedie, M. W. Wright, and E. A. Bruford. "A review of the new HGNC gene family resource." In: *Human Genomics* 10.1 (Feb. 2016). DOI: 10.1186/s40246-016-0062-6. URL: https://doi.org/10.1186/s40246-016-0062-6.

[74] J. A. Blake et al. "Mouse Genome Database (MGD): Knowledgebase for mouse-human comparative biology." In: *Nucleic Acids Res* 49.D1 (Jan. 2021), pp. D981–D987.

[75] L. S. Gramates et al. "FlyBase: a guided tour of highlighted features." In: *Genetics* 220.4 (Apr. 2022).

[76] Renqian Luo, Liai Sun, Yingce Xia, Tao Qin, Sheng Zhang, Hoifung Poon, and Tie-Yan Liu. "BioGPT: generative pre-trained transformer for biomedical text generation and mining." In: *Briefings in Bioinformatics* 23.6 (Sept. 2022), bbac409. ISSN: 1477-4054. DOI: 10.1093/bib/bbac409. eprint: https://academic.oup.com/bib/article-pdf/23/6/bbac409/47144271/bbac409.pdf. URL: https://doi.org/10.1093/bib/bbac409.

[77] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. "BioBERT: a pre-trained biomedical language representation model for biomedical text mining." In: *Bioinformatics* 36.4 (Sept. 2019), pp. 1234–1240. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btz682. eprint: https://academic.oup.com/bioinformatics/article-pdf/36/4/1234/48983216/bioinformatics_36_4_1234.pdf. URL: https://doi.org/10.1093/bioinformatics/btz682.

[78] Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. *Domain-Specific Language Model Pretraining for Biomedical Natural Language Processing*. 2020. eprint: arXiv:2007.15779.

[79] Iz Beltagy, Kyle Lo, and Arman Cohan. "SciBERT: Pretrained Language Model for Scientific Text." In: *EMNLP*. 2019. eprint: arXiv:1903.10676.

[80] Ruiqing Ding, Xiao Han, and Leye Wang. *A Unified Knowledge Graph Augmentation Service for Boosting Domain-specific NLP Tasks*. 2023. arXiv: 2212.05251 [cs.CL].

[81] Toyofumi Fujiwara. "Open Agile text mining for bioinformatics: the PubAnnotation ecosystem." In: *Bioinformatics* 35 (Apr. 2019). DOI: 10.1093/bioinformatics/btz227.

[82] Robert Leaman, Rezarta Islamaj Doğan, and Zhiyong Lu. "DNorm: disease name normalization with pairwise learning to rank." In: *Bioinformatics* 29.22 (Aug. 2013), pp. 2909–2917. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btt474. eprint: https://academic.oup.com/bioinformatics/article-pdf/29/22/2909/48891951/bioinformatics_29_22_2909.pdf. URL: https://doi.org/10.1093/bioinformatics/btt474.

[83] R. Leaman and Z. Lu. "TaggerOne: joint named entity recognition and normalization with semi-Markov Models." In: *Bioinformatics* 32.18 (Sept. 2016), pp. 2839–2846.

[84] Mujeen Sung, Minbyul Jeong, Yonghwa Choi, Donghyeon Kim, Jinhyuk Lee, and Jaewoo Kang. "BERN2: an advanced neural biomedical named entity recognition and normalization tool." In: *Bioinformatics* 38.20 (Sept. 2022), pp. 4837–4839. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btac598. eprint: https://academic.oup.com/bioinformatics/article-pdf/38/20/4837/46535173/btac598.pdf. URL: https://doi.org/10.1093/bioinformatics/btac598.

[85] Patrick Lewis, Myle Ott, Jingfei Du, and Veselin Stoyanov. "Pretrained Language Models for Biomedical and Clinical Tasks: Understanding and Extending the State-of-the-Art." In: *Proceedings of the 3rd Clinical Natural Language Processing Workshop*. Ed. by Anna Rumshisky, Kirk Roberts, Steven Bethard, and Tristan Naumann. Online: Association for Computational Linguistics, Nov. 2020, pp. 146–157. DOI: 10.18653/v1/2020.clinicalnlp-1.17. URL: https://aclanthology.org/2020.clinicalnlp-1.17.

[86] L. Smith et al. "Overview of BioCreative II gene mention recognition." In: *Genome Biology* 9 (Sept. 2008).

[87] Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. "Special Report: NCBI Disease Corpus: A Resource for Disease Name Recognition and Concept Normalization." In: *J. of Biomedical Informatics* 47 (Feb. 2014), 1–10. ISSN: 1532-0464.

[88] Eric Sayers. *A General Introduction to the E-utilities*. https://www.ncbi.nlm.nih.gov/books/NBK25497/. 2009.

[89] E. Birney. "An Overview of Ensembl." In: *Genome Research* 14.5 (May 2004), pp. 925–928. DOI: 10.1101/gr.1860604. URL: https://doi.org/10.1101/gr.1860604.

[90] M. Griffith et al. "CIViC is a community knowledgebase for expert crowdsourcing the clinical interpretation of variants in cancer." In: *Nature Genetics* 49.2 (Jan. 2017), pp. 170–174. DOI: 10.1038/ng.3774. URL: https://doi.org/10.1038/ng.3774.

[91] M. Whirl-Carrillo, E. M. McDonagh, J. M. Hebert, L. Gong, K. Sangkuhl, C. F. Thorn, R. B. Altman, and T. E. Klein. "Pharmacogenomics Knowledge for Personalized Medicine." In: *Clinical Pharmacology & Therapeutics* 92.4 (Oct. 2012), pp. 414–417. DOI: 10.1038/clpt.2012.96. URL: https://doi.org/10.1038/clpt.2012.96.

[92] B. Smith et al. "The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration." In: *Nature Biotechnology* 25.11 (Nov. 2007), pp. 1251–1255. DOI: 10.1038/nbt1346. URL: https://doi.org/10.1038/nbt1346.

[93] D. A. Natale et al. "Protein Ontology (PRO): enhancing and scaling up the representation of protein entities." In: *Nucleic Acids Research* 45.D1 (Nov. 2016), pp. D339–D346. DOI: 10.1093/nar/gkw1075. URL: https://doi.org/10.1093/nar/gkw1075.

[94] M. Gremse, A. Chang, I. Schomburg, A. Grote, M. Scheer, C. Ebeling, and D. Schomburg. "The BRENDA Tissue Ontology (BTO): the first all-integrating ontology of all organisms for enzyme sources." In: *Nucleic Acids Research* 39.Database (Oct. 2010), pp. D507–D513. DOI: 10.1093/nar/gkq968. URL: https://doi.org/10.1093/nar/gkq968.

[95] J. B. L. Bard. "The AEO, an Ontology of Anatomical Entities for Classifying Animal Tissues and Organs." In: *Frontiers in Genetics* 3 (2012). DOI: 10.3389/fgene.2012.00018. URL: https://doi.org/10.3389/fgene.2012.00018.

[96] OBO Technical WG. *Phenotype And Trait Ontology*. URL: http://obofoundry.org/ontology/pato.html.

[97] Alexander D. Diehl et al. "The Cell Ontology 2016: enhanced content, modularization, and ontology interoperability." In: *Journal of Biomedical Semantics* 7.1 (July 2016). DOI: 10.1186/s13326-016-0088-7. URL: https://doi.org/10.1186/s13326-016-0088-7.

[98] Sirarat Sarntivijai et al. "CLO: The cell line ontology." In: *Journal of Biomedical Semantics* 5.1 (2014), p. 37. DOI: 10.1186/2041-1480-5-37. URL: https://doi.org/10.1186/2041-1480-5-37.

[99] The UniProt Consortium. "UniProt: the universal protein knowledgebase." In: *Nucleic Acids Research* 45.D1 (Nov. 2016), pp. D158–D169. ISSN: 0305-1048. DOI: 10.1093/nar/gkw1099. eprint: https://academic.oup.com/nar/article-pdf/45/D1/D158/23819877/gkw1099.pdf. URL: https://doi.org/10.1093/nar/gkw1099.

[100] Daniel Himmelstein and Sergio Baranzini. "Heterogeneous Network Edge Prediction: A Data Integration Approach to Prioritize Disease-Associated Genes." In: (Dec. 2014). DOI: 10.1101/011569.

[101] John H Morris et al. "The scalable precision medicine open knowledge engine (SPOKE): a massive knowledge graph of biomedical information." In: *Bioinformatics* 39.2 (Feb. 2023). ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btad080. eprint: https://academic.oup.com/bioinformatics/article-pdf/39/2/btad080/49278346/btad080.pdf. URL: https://doi.org/10.1093/bioinformatics/btad080.

[102] David N. Nicholson, Daniel S. Himmelstein, and Casey S. Greene. "Expanding a database-derived biomedical knowledge graph via multi-relation extraction from biomedical abstracts." In: *BioData Mining* 15.1 (Oct. 2022), p. 26. ISSN: 1756-0381. DOI: 10.1186/s13040-022-00311-z. URL: https://doi.org/10.1186/s13040-022-00311-z.

[103] David N. Nicholson and Casey S. Greene. "Constructing knowledge graphs and their biomedical applications." In: *Computational and Structural Biotechnology Journal* 18 (2020), pp. 1414–1428. ISSN: 2001-0370. DOI: https://doi.org/10.1016/j.csbj.2020.05.017. URL: https://www.sciencedirect.com/science/article/pii/S2001037020302804.

[104] Sheng Yu et al. "BIOS: An Algorithmically Generated Biomedical Knowledge Graph." In: *ArXiv* abs/2203.09975 (2022). URL: https://api.semanticscholar.org/CorpusID:247594837.

[105] Kang Zhou, Yuepei Li, and Qi Li. "Distantly Supervised Named Entity Recognition via Confidence-Based Multi-Class Positive and Unlabeled Learning." In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Smaranda Muresan, Preslav Nakov, and Aline Villavicencio. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 7198–7211. DOI: 10.18653/v1/2022.acl-long.498. URL: https://aclanthology.org/2022.acl-long.498.

[106] O. Bodenreider. "The Unified Medical Language System (UMLS): integrating biomedical terminology." In: *Nucleic Acids Res* 32.Database issue (Jan. 2004), pp. D267–270.

[107] Alberto Santos et al. "A knowledge graph to interpret clinical proteomics data." In: *Nature Biotechnology* 40.5 (May 2022), pp. 692–702. ISSN: 1546-1696. DOI: 10.1038/s41587-021-01145-6. URL: https://doi.org/10.1038/s41587-021-01145-6.

[108] E. C. Wood et al. "RTX-KG2: a system for building a semantically standardized knowledge graph for translational biomedicine." In: *BMC Bioinformatics* 23.1 (Sept. 2022). DOI: 10.1186/s12859-022-04932-3. URL: https://doi.org/10.1186/s12859-022-04932-3.

[109] D. R. Unni et al. "Biolink Model: A universal schema for knowledge graphs in clinical, biomedical, and translational science." In: *Clin Transl Sci* 15.8 (Aug. 2022), pp. 1848–1855.

[110] Jian Xu et al. "Building a PubMed knowledge graph." In: *Scientific Data* 7.1 (June 2020), p. 205. ISSN: 2052-4463. DOI: 10.1038/s41597-020-0543-2. URL: https://doi.org/10.1038/s41597-020-0543-2.

[111] Evangelos Pafilis, Pier Luigi Buttigieg, Barbra Ferrell, Emiliano Pereira, Julia Schnetzer, Christos Arvanitidis, and Lars Juhl Jensen. "EXTRACT: interactive extraction of environment metadata and term suggestion for metagenomic sample annotation." In: *Database* 2016 (2016), baw005.

[112] Jianbo Yuan, Zhiwei Jin, Han Guo, Hongxia Jin, Xianchao Zhang, Tristram Smith, and Jiebo Luo. "Constructing biomedical domain-specific knowledge graph with minimum supervision." In: *Knowledge and Information Systems* 62 (Jan. 2020), pp. 1–20. DOI: 10.1007/s10115-019-01351-4.

[113] Mark Wilkinson et al. "The FAIR Guiding Principles for scientific data management and stewardship." In: *Scientific Data* 3 (Mar. 2016). DOI: 10.1038/sdata.2016.18.

[114] Halil Kilicoglu, Dongwook Shin, Marcelo Fiszman, Graciela Rosemblat, and Thomas C. Rindflesch. "SemMedDB: a PubMed-scale repository of biomedical semantic predications." In: *Bioinformatics* 28.23 (Oct. 2012), pp. 3158–3160. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bts591. eprint: https://academic.oup.com/bioinformatics/article-pdf/28/23/3158/6143885/bts591.pdf. URL: https://doi.org/10.1093/bioinformatics/bts591.

[115] Guanxiong Zhang et al. "DiseaseEnhancer: a resource of human disease-associated enhancer catalog." In: *Nucleic Acids Research* 46.D1 (Oct. 2017), pp. D78–D84. ISSN: 0305-1048. DOI: 10.1093/nar/gkx920. URL: https://doi.org/10.1093/nar/gkx920.

[116] Pieter-Jan Volders, Kenny Helsens, Xiaowei Wang, Björn Menten, Lennart Martens, Kris Gevaert, Jo Vandesompele, and Pieter Mestdagh. "LNCipedia: a database for annotated human lncRNA transcript sequences and structures." In: *Nucleic Acids Research* 41.D1 (Oct. 2012), pp. D246–D251. DOI: 10.1093/nar/gks915. URL: https://doi.org/10.1093/nar/gks915.

[117] Ashwini Jeggari, Debora S Marks, and Erik Larsson. "miRcode: a map of putative microRNA target sites in the long non-coding transcriptome." In: *Bioinformatics* 28.15 (May 2012), pp. 2062–2063. DOI: 10.1093/bioinformatics/bts344. URL: https://doi.org/10.1093/bioinformatics/bts344.

[118] Ana Kozomara, Maria Birgaoanu, and Sam Griffiths-Jones. "miRBase: from microRNA sequences to function." In: *Nucleic Acids Research* 47.D1 (Nov. 2018), pp. D155–D162. ISSN: 0305-1048. DOI: 10.1093/nar/gky1141. URL: https://doi.org/10.1093/nar/gky1141.

[119] Hsi-Yuan Huang et al. "miRTarBase 2020: updates to the experimentally validated microRNA–target interaction database." In: *Nucleic Acids Research* 48.D1 (Oct. 2019), pp. D148–D154. ISSN: 0305-1048. DOI: 10.1093/nar/gkz896. eprint: https://academic.oup.com/nar/article-pdf/48/D1/D148/31697874/gkz896.pdf. URL: https://doi.org/10.1093/nar/gkz896.

[120] B. Xie, Q. Ding, H. Han, and D. Wu. "miRCancer: a microRNA-cancer association database constructed by text mining on literature." In: *Bioinformatics* 29.5 (Jan. 2013), pp. 638–644. DOI: 10.1093/bioinformatics/btt014. URL: https://doi.org/10.1093/bioinformatics/btt014.

[121] D. Croft et al. "Reactome: a database of reactions, pathways and biological processes." In: *Nucleic Acids Research* 39.Database (Nov. 2010), pp. D691–D697. DOI: 10.1093/nar/gkq1018. URL: https://doi.org/10.1093/nar/gkq1018.

[122] G. Joshi-Tope. "Reactome: a knowledgebase of biological pathways." In: *Nucleic Acids Research* 33.Database issue (Dec. 2004), pp. D428–D432. DOI: 10.1093/nar/gki072. URL: https://doi.org/10.1093/nar/gki072.

[123] David Croft et al. "The Reactome pathway knowledgebase." In: *Nucleic Acids Research* 42.D1 (Nov. 2013), pp. D472–D477. DOI: 10.1093/nar/gkt1102. URL: https://doi.org/10.1093/nar/gkt1102.

[124] Max Franz, Christian T. Lopes, Gerardo Huck, Yue Dong, Onur Sumer, and Gary D. Bader. "Cytoscape.js: a graph theory library for visualisation and analysis." In: *Bioinformatics* (Sept. 2015), btv557. DOI: 10.1093/bioinformatics/btv557. URL: https://doi.org/10.1093/bioinformatics/btv557.

[125] S. Alaimo, R. V. Rapicavoli, G. P. Marceca, A. La Ferlita, O. B. Serebrennikova, P. N. Tsichlis, B. Mishra, A. Pulvirenti, and A. Ferro. "PHENSIM: Phenotype Simulator." In: *bioRxiv* (2020). DOI: 10.1101/2020.01.20.912279. eprint: https://www.biorxiv.org/content/early/2020/10/06/2020.01.20.912279.full.pdf. URL: https://www.biorxiv.org/content/early/2020/10/06/2020.01.20.912279.

[126] Minoru Kanehisa and Susumu Goto. "KEGG: Kyoto Encyclopedia of Genes and Genomes." In: *Nucleic Acids Research* 28.1 (Jan. 2000), pp. 27–30. ISSN: 0305-1048. DOI: 10.1093/nar/28.1.27. URL: https://doi.org/10.1093/nar/28.1.27.

[127] Lijuan Xiong, Carl Edwards, and Lijun Zhou. "The Biological Function and Clinical Utilization of CD147 in Human Diseases: A Review of the Current Scientific Literature." In: *International Journal of Molecular Sciences* 15.10 (Sept. 2014), pp. 17411–17441. DOI: 10.3390/ijms151017411. URL: https://doi.org/10.3390/ijms151017411.

[128] Dai-Hung Ngo, Thanh-Sang Vo, Dai-Nghiep Ngo, Isuru Wijesekara, and Se-Kwon Kim. "Biological activities and potential health benefits of bioactive peptides derived from marine organisms." In: *International Journal of Biological Macromolecules* 51.4 (Nov. 2012), pp. 378–383. DOI: 10.1016/j.ijbiomac.2012.06.001. URL: https://doi.org/10.1016/j.ijbiomac.2012.06.001.

[129] Jiao Wang et al. "The role of phosphatidylserine on the membrane in immunity and blood coagulation." In: *Biomarker Research* 10.1 (Jan. 2022). DOI: 10.1186/s40364-021-00346-0. URL: https://doi.org/10.1186/s40364-021-00346-0.

[130] Richard C. Becker, Deepak Voora, and Svati H. Shah. "Hemostasis and Thrombosis." In: *Genomic and Personalized Medicine*. Elsevier, 2013, pp. 602–611. DOI: 10.1016/b978-0-12-382227-7.00052-5. URL: https://doi.org/10.1016/b978-0-12-382227-7.00052-5.

[131] The Guardian Libby Brooks and Alex Hern. *Shock an aw: US teenager wrote huge slice of Scots Wikipedia*. https://www.theguardian.com/uk-news/2020/aug/26/shock-an-aw-us-teenager-wrote-huge-slice-of-scots-wikipedia. Aug. 2020.

[132] Gonzalo Navarro. "A Guided Tour to Approximate String Matching." In: *ACM Comput. Surv.* 33.1 (Mar. 2001), 31–88. ISSN: 0360-0300. DOI: 10.1145/375360.375365. URL: https://doi.org/10.1145/375360.375365.

[133] Minoru Kanehisa. "Toward understanding the origin and evolution of cellular organisms." In: *Protein Science* 28.11 (Sept. 2019), pp. 1947–1951. DOI: 10.1002/pro.3715. URL: https://doi.org/10.1002/pro.3715.

[134] M. Kanehisa. "KEGG: Kyoto Encyclopedia of Genes and Genomes." In: *Nucleic Acids Research* 28.1 (Jan. 2000), pp. 27–30. DOI: 10.1093/nar/28.1.27. URL: https://doi.org/10.1093/nar/28.1.27.

[135] P. Blohm, G. Frishman, P. Smialowski, F. Goebels, B. Wachinger, A. Ruepp, and D. Frishman. "Negatome 2.0: a database of non-interacting proteins derived by literature mining, manual annotation and protein structure analysis." In: *Nucleic Acids Research* 42.D1 (Nov. 2013), pp. D396–D400. DOI: 10.1093/nar/gkt1079. URL: https://doi.org/10.1093/nar/gkt1079.

[136] Pawel Smialowski et al. "The Negatome database: a reference set of non-interacting protein pairs." In: *Nucleic Acids Research* 38.suppl_1 (Nov. 2009), pp. D540–D544. DOI: 10.1093/nar/gkp1026. URL: https://doi.org/10.1093/nar/gkp1026.

[137] J. Menche, A. Sharma, M. Kitsak, S. D. Ghiassian, M. Vidal, J. Loscalzo, and A.-L. Barabasi. "Uncovering disease-disease relationships through the incomplete interactome." In: *Science* 347.6224 (Feb. 2015), pp. 1257601–1257601. DOI: 10.1126/science.1257601. URL: https://doi.org/10.1126/science.1257601.

[138] Ke Wang et al. "CD147-spike protein is a novel route for SARS-CoV-2 infection to host cells." In: *Signal Transduction and Targeted Therapy* 5.1 (Dec. 2020). DOI: 10.1038/s41392-020-00426-x. URL: https://doi.org/10.1038/s41392-020-00426-x.

[139] Zhi Jiang, Shuijun Hu, Dong HUua, Jianlong Ni, Lan Xu, Yan Ge, Yinghui Zhou, Zhihong Cheng, and Shiliang Wu. "β3GnT8 plays an important role in CD147 signal transduction as an upstream modulator of MMP production in tumor cells." In: *Oncology Reports* 32.3 (June 2014), pp. 1156–1162. DOI: 10.3892/or.2014.3280. URL: https://doi.org/10.3892/or.2014.3280.

[140] Ling-Min Kong, Cheng-Gong Liao, Yang Zhang, Jing Xu, Yu Li, Wan Huang, Yi Zhang, Huijie Bian, and Zhi-Nan Chen. "A Regulatory Loop Involving miR-22, Sp1, and c-Myc Modulates CD147 Expression in Breast Cancer Invasion and Metastasis." In: *Cancer Research* 74.14 (June 2014), pp. 3764–3778. DOI: 10.1158/0008-5472.can-13-3555. URL: https://doi.org/10.1158/0008-5472.can-13-3555.

[141] Xia Ke, Fei Fei, Yanke Chen, Li Xu, Zheng Zhang, Qichao Huang, Hongxin Zhang, Hushan Yang, Zhinan Chen, and Jinliang Xing. "Hypoxia upregulates CD147 through a combined effect of HIF-1α and Sp1 to promote glycolysis and tumor progression in epithelial solid tumors." In: *Carcinogenesis* 33.8 (June 2012), pp. 1598–1607. DOI: 10.1093/carcin/bgs196. URL: https://doi.org/10.1093/carcin/bgs196.

[142] G. D. Grass and B. P. Toole. "How, with whom and when: an overview of CD147-mediated regulatory networks influencing matrix metalloproteinase activity." In: *Bioscience Reports* 36.1 (Jan. 2016). DOI: 10.1042/bsr20150256. URL: https://doi.org/10.1042/bsr20150256.

[143] N. Rucci, D. Millimaggi, M. Mari, A. Del Fattore, M. Bologna, A. Teti, A. Angelucci, and V. Dolo. "Receptor Activator of NFKB Ligand Enhances Breast Cancer– Induced Osteolytic Lesions through Upregulation of Extracellular Matrix Metalloproteinase Inducer CD147." In: *Cancer Research* 70.15 (July 2010), pp. 6150–6160. DOI: 10.1158/0008-5472.can-09-2758. URL: https://doi.org/10.1158/0008-5472.can-09-2758.

[144] P. Ding, X. Zhang, S. Jin, B. Duan, P. Chu, Y. Zhang, Z. Chen, B. Xia, and F. Song. "CD147 functions as the signaling receptor for extracellular divalent copper in hepatocellular carcinoma cells." In: *Oncotarget* 8.31 (May 2017), pp. 51151–51163. DOI: 10.18632/oncotarget.17712. URL: https://doi.org/10.18632/oncotarget.17712.

[145] Henning Ulrich and Micheli M. Pillat. "CD147 as a Target for COVID-19 Treatment: Suggested Effects of Azithromycin and Stem Cell Engagement." In: *Stem Cell Reviews and Reports* 16.3 (Apr. 2020), pp. 434–440. DOI: 10.1007/s12015-020-09976-7. URL: https://doi.org/10.1007/s12015-020-09976-7.

[146] Shi-Jie Wang, Hong-Yong Cui, Yan-Mei Liu, Pu Zhao, Yang Zhang, Zhi-Guang Fu, Zhi-Nan Chen, and Jian-Li Jiang. "CD147 promotes Src-dependent activation of Rac1 signaling through STAT3/DOCK8 during the motility of hepatocellular carcinoma cells." In: *Oncotarget* 6.1 (Nov. 2014), pp. 243–257. DOI: 10.18632/oncotarget.2801. URL: https://doi.org/10.18632/oncotarget.2801.

[147] P. Kirk, M.C. Wilson, C. Heddle, M.H. Brown, A.N. Barclay, and A.P. Halestrap. "CD147 is tightly associated with lactate transporters MCT1 and MCT4 and facilitates their cell surface expression." In: *The EMBO Journal* 19.15 (Aug. 2000),

pp. 3896–3904. DOI: 10.1093/emboj/19.15.3896. URL: https://doi.org/10.1093/emboj/19.15.3896.

[148] L. ksnebjerg, B. Woods, and G. Waldemar. "Designing the ReACT App to Support Self-Management of People with Dementia: An Iterative User-Involving Process." In: *Gerontology* 65.6 (2019), pp. 673–685.

[149] H. K. Rai, J. Schneider, and M. Orrell. "An Individual Cognitive Stimulation Therapy App for People with Dementia and Carers: Results from a Feasibility Randomized Controlled Trial (RCT)." In: *Clin Interv Aging* 16 (2021), pp. 2079–2094.

[150] I. A. Ebeid. "MedGraph: A semantic biomedical information retrieval framework using knowledge graph embedding for PubMed." In: *Front Big Data* 5 (2022), p. 965619.