ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Natural Language Processing

AUTOMATIC TERMINOLOGY CODING FOR THE BIOMEDICAL DOMAIN

Candidate
Emmanuele Bollino

Supervisor
Prof. Paolo Torroni

Co-Supervisors
Prof. Andrea Galassi
Vieri Emiliani

Academic year 2022-2023

Session 2nd

A nonna Lucia

Abstract

The biomedical sector, rich in unstructured data from sources like clinical notes and health records, presents a prime opportunity for Natural Language Processing (NLP) applications. Especially pivotal is the task of entity linking, wherein textual mentions are mapped to medical concepts within a knowledge base, in this case, represented by the Unified Medical Language System (UMLS) Metathesaurus. Within this realm, the Italian language faces resource constraints (only 4% of UMLS 4M concepts have a label in the Italian language). Current systems like MAPS Group's Clinika software lean on label matching to link the extracted facts to the corresponding UMLS concepts. This dissertation deals with the design of a new Clinika component aimed at enhancing entity linking for Italian terms against UMLS, even in the absence of direct Italian labels. Employing transformer-based multilingual embeddings, a novel 'concept guesser' architecture was developed to tackle the linking challenge intelligently, maximizing the level of exploitation of the currently available knowledge. This innovation not only enhances Clinika's effectiveness but also paves the way for advanced multilingual clinical decision support systems.

Contents

Chapter 1

Introduction

The project that is the subject of this thesis was performed during a curricular internship period at MAPS S.p.A. within the research and development area. The principal objective was to enhance the capabilities of the Clinika software regarding the entity linking of clinical terms against the UMLS metathesaurus. This led to an exploration of the domain, an assessment of the available resources, an examination of the state-of-the-art techniques, the development of an architecture that integrates various components to perform the task, and an evaluation of the results.

1.1 Context and Domain

The biomedical domain is relevant for healthcare services, clinical research, and medical education. Within this complex landscape, the profusion of medical documentation, in the form of electronic medical records (EMR), clinical notes, medical publications, or prescriptions, forms the basis of clinical and administrative decision-making. Nonetheless, the unstructured disposition of this voluminous documentation constitutes a limitation to its effective utilization. A significant portion of valuable information within this domain is in the form of textual narratives that necessitate systematic organization and analysis to unlock their potential in serving the diverse tasks in healthcare.

One of the pivotal tasks toward the effective utilization of information in these documents is the clinical terminology coding [Blundell, 2023]. Coding refers to the process of mapping free-text clinical terms to standardized terms or codes from established medical terminologies. This endeavor provides a structured and standardized representation of medical information, which is instrumental for a multitude of applications ranging from clinical decision support systems, healthcare analytics, to research and development in medicine.

The Unified Medical Language System (UMLS) [Bodenreider, 2004] stands as a comprehensive resource that aggregates numerous health and biomedical vocabularies, providing a unified, standardized lexicon along with a semantic network to elucidate relationships between medical concepts. Mapping clinical terms to such a metathesaurus allows the effective utilization of the extracted knowledge by reasoning upon it in a systematic and reliable way.

In the context of a multilingual and multicultural world, the endeavor of encoding assumes an additional layer of complexity. Languages such as Italian, with a lack of resources compared to English, face an uphill challenge in the precise mapping of clinical terms to established knowledge bases like UMLS. The absence of direct translations or equivalents makes the challenge more difficult, seeking innovative approaches to face these issues.

Once clinical terms have been encoded against the Unified Medical Language System (UMLS), a wide array of tasks can be facilitated within the biomedical domain. Here are some of the prominent tasks:

Clinical Decision Support Systems CDS Systems [Wasylewicz and Scheepers-Hoeks, 2019] can rely on encoded terms to provide robust and accurate results based on standardized and structured data, that can be analyzed to support clinicians.

Healthcare Analytics Healthcare organizations can perform analytics to monitor and improve the quality of care, manage operations efficiently, and identify areas for cost savings or operational improvements.

Regulatory Compliance and Reporting Standardized coding of clinical terms assists in meeting regulatory requirements and streamlined reporting to various stakeholders, including government health departments, insurance companies, and accreditation organizations. In Italy, the legislation [Italian Government, 2008] forces the codification of some information in the hospital discharge records against the International Classification of Diseases (ICD9CM).

Interoperability Encoding clinical terms fosters interoperability among different healthcare information systems by enabling the seamless exchange and interpretation of medical data.

Semantic Search and Retrieval Enables more effective and semantically enriched search and retrieval of medical information, which is crucial for clinicians, researchers, and healthcare administrators.

1.2 Clinika

Clinika, a patented software developed by MAPS S.p.A. [Emiliani, 2017], serves as an automated analytical tool for processing medical documents. It uses a semantic engine to extract and code relevant concepts in clinical texts for automated processes and decision support systems in clinical and administrative settings. Since 2013, several Italian local health units have adopted Clinika for their operational needs. In particular, Clinika VAP (Automatic Prescription Verification) is capable of examining the unstructured free text of medical prescriptions and extracting structured knowledge that supports strategic decisions. The objective is to evaluate and verify the adequateness of the prescriptions made by healthcare professionals. The insights provided by Clinika enable the directors of a health facility to strategically orchestrate resource allocation and identify critical areas where enhanced awareness among the doctors is needed.

Specifically, Clinika executes Named Entity Recognition (NER) on an array of medical documents including medical reports, prescriptions, and discharge letters. It is notable that all documents processed are articulated in the Italian language. Clinika ambitiously aims to map interesting targets to a pre-established terminology system encompassing millions of potential terms. Among the well-established and reputable terminology systems in the biomedical field, MAPS S.p.A. has opted for UMLS as the primary reference metathesaurus.

The legacy version of Clinika preceding this work relied on the exact matching of Italian labels to associate targets with UMLS concepts, a methodology that, while straightforward, posed limitations in terms of flexibility and comprehensiveness. This approach necessitated a precise correspondence between the labels within the documents and those existing within the UMLS metathesaurus, potentially overlooking or misrepresenting nuanced or varied terminological expressions prevalent in clinical discourse.

This thesis relies on and integrates the work of the existing Clinika infrastructure and the work of the other interns. The most relevant component is the data extraction pipeline. It integrates a series of SpaCy components designed to discern significant tokens within an input clinical text. The Dependency Parser is the most relevant component, that is in charge of determining the syntactic structure of the sentences with all their linguistic relations. This task is quite difficult because of the nature of clinical documents that are often written in a nominal way deviating from the standard logical and grammar conventions. This pipeline yields a list of mentions with eventually some attached metadata, that are intended to be the relevant terms that need to be coded against the UMLS knowledge base.

1.3 Objectives

This thesis delves into the relevant task of encoding clinical terms, particularly focusing on bridging the linguistic gap faced by the Italian language, by mapping them to the UMLS metathesaurus. By exploiting the advancements in Natural Language Processing (NLP), machine learning, and biomedical knowledge basis, this thesis aims to contribute toward enhancing the effectiveness and accessibility of medical information across linguistic boundaries. By referring to Clinika software developed by MAPS S.p.A., the subsequent chapters will unfold the design, implementation, and evaluation of a new architecture aimed at automatic terminology coding for the biomedical domain.

Overall, the objective is to build a system that reliably performs entity-linking between extracted Italian mentions and the UMLS concepts. The main challenges to deal with are about the scarcity of Italian labels in the UMLS metathesaurus, the disambiguation of similar but unrelated terms, and the alignment to the most pertinent and specific concept. Each of these challenges requires a detailed and clear approach to build a strong linking framework that can manage the complex language of the biomedical field. By solving these challenges, this project aims to greatly improve the precision of Clinika's entity-linking system and the amount of coded terms, thereby increasing the correctness of information collected and supporting better clinical and administrative decisions.

The main challenge is to correctly link an Italian biomedical term to the most pertinent UMLS concept even in the absence of a corresponding Italian label. For instance, the text 'collo omerale'[1] is not present in UMLS, while there is the word 'collo'[2] and 'omero'[3]. The word 'collo'[2] is also ambiguous. So, associating the right CUI to this piece of text is not a straightforward task.

[1]Italian for 'humeral neck'
[2]Italian for 'neck'
[3]Italian for 'humerus'

Chapter 2

Background

This chapter delves into a comprehensive examination of the existing resources that allow the achievement of the goals of this project, with a great focus on the NLP techniques pivotal for the entity linking task.

2.1 UMLS

A medical terminology system is a specialized language or standardized vocabulary used in the healthcare industry to accurately describe the processes, procedures, symptoms, diseases, and conditions pertaining to medical practice and research. These terminologies enable clear communication and shared understanding among healthcare professionals, researchers, and administrators, ensuring consistency and reducing ambiguity in the documentation and communication of medical information. The main advantages of using a medical terminology system include standardization. Coding and classification allow for the easy grouping and identification of diseases, conditions, and treatments, and enable the systematic classification of health information crucial for analytical purposes. Examples of medical terminology systems include the Systematized Nomenclature of Medicine – Clinical

Terms[1] (SNOMED CT), the International Classification of Diseases[2] (ICD), the Logical Observation Identifiers Names and Codes[3] (LOINC), and the Unified Medical Language System (UMLS) which integrates several standardized medical terminologies into a single, comprehensive system. Medical terminology systems may also include relational and hierarchical structures that delineate the relationships among diverse concepts, facilitating more streamlined information retrieval and analysis.

The Unified Medical Language System[4] (UMLS) [Bodenreider, 2004] is a set of databases and tools developed by the U.S. National Library of Medicine[5] (NLM). UMLS provides a common framework that bridges various vocabularies and standards used in the healthcare domain to enable computer systems to better process medical information. The UMLS reference version for this thesis is the 2020AB. UMLS consists of three knowledge sources: metathesaurus, semantic network, and lexicon and lexical tools.

Metathesaurus The UMLS Metathesaurus is the primary component of the Unified Medical Language System (UMLS). It is essentially a large, multipurpose, and multilingual database containing biomedical and health-related concepts. It aggregates and interlinks a wide variety of different terminologies and classifications used across various healthcare systems, research domains, and standard-setting organizations.

Conceptual Organization Various elements play crucial roles in forming a coherent and comprehensive terminology system. At the core are ‘concepts’, which represent distinct meanings or ideas in the biomedical or healthcare domain. Each concept aggregates various terms or phrases that share the same meaning, from different source vocabularies, into a unified entity. These

[1]`https://www.snomed.org/`
[2]`https://www.who.int/standards/classifications/classification-of-diseases`
[3]`https://loinc.org/`
[4]`https://www.nlm.nih.gov/research/umls/`
[5]`https://www.nlm.nih.gov/`

terms or phrases are referred to as ‘atoms’. Each atom within a concept essentially represents a different expression for the same underlying idea. These atoms are derived from different ‘dictionaries’ or source vocabularies, each with a specific associated ‘string’ and language. The dictionaries encompass a range of specialized vocabularies, standards, or coding systems utilized across different sectors within healthcare and biomedical domains. On another layer, ‘relationships’ form the backbone of the conceptual structure by interlinking atoms, thus also concepts, in a meaningful manner. Relationships can be hierarchical, where a parent-child relationship between concepts is established, or associative, where non-hierarchical connections are made, such as the relationship between a disease and its common treatments.
UMLS includes a vocabulary ranking system[6] that assigns a rank score to the combination of SAB (vocabulary) and TTY (term type), enabling a dependable ranking of concept terms.

Identifiers Each component is identified with a unique code. From a bottom-up perspective, each source dictionary is identified by its Source Abbreviation (SAB). The atoms derived from a dictionary are identified by the Atom Unique Identifier (AUI). Language associated with the atoms is identified by a Language of Terms (LAT). The strings are identified by the String Unique Identifier (SUI). The unique identifier of a string in a source is the CODE (dictionary dependent). The concepts, atoms aggregators, are identified by a Concept Unique Identifier (CUI). Then, each relationship is identified by a Relationship Unique Identifier (RUI) and its type is the REL with optionally a standardized specification, the Relationship Attribute (RELA). Finally, the semantic type of a concept is identified by a Type Unique Identifier (TUI). There are more identifiers and components but the ones described are enough for the sake of this thesis.

[6]Table MRRANK in UMLS

Semantic Network The Semantic Network acts as a categorization mechanism for all concepts encapsulated in the Metathesaurus. It organizes these concepts into broad, generally understood classes, and delineates relationships among them. A fundamental element of the Semantic Network is the 'Semantic Type'. Each concept in the Metathesaurus is assigned at least one semantic type, that represents its category. Each semantic type possesses a unique definition and a designated place in the network's hierarchy, with examples including 'Disease or Syndrome', 'Pharmacologic Substance', and 'Diagnostic Procedure'. Beyond the semantic types, 'Semantic Relationships' are defined; these relationships elucidate how different concepts interrelate within the biomedical domain, with common semantic relationships being "is_a" (a hierarchical relationship) or "treats" (a non-hierarchical relationship). To facilitate an even higher-level organization, 'Semantic Groups' aggregate semantic types that share logical associations. These groups further organize the semantic types into broader categories, for instance, all diseases and pathological conditions might be congregated into the 'Disorders' group.

SPECIALIST Lexicon and Lexical Tools These resources are useful for some medical natural language processing (NLP) tasks. The SPECIALIST Lexicon is a comprehensive lexical database with biomedical and general English vocabulary, crucial for supporting linguistic applications in processing biomedical texts. On the other hand, the Lexical Tools, leveraging the SPECIALIST Lexicon, facilitate various lexical operations for NLP tasks such as morphological analysis and synonym identification.

2.1.1 Quantitative Analysis

UMLS is an important and rich resource and in order to understand it better and make data-driven decisions, it is important to perform a quantitative analysis of the metathesaurus, by looking at the distribution of its components.

Languages

UMLS lacks solid multi-lingual support, there are 25 represented languages. However, its English coverage is quite impressive but the difference between English and any other language is quite large. For the sake of simplicity, a concept is said to belong to a language if it exists at least one label of that language. There are only 1.151 concepts without an English label. As shown in table 2.1 even if the Italian language is the third most-covered language in terms of concepts, the ratio between the number of Italian concepts and English concepts is less than 4%. It also emerges that Latin and Germanic (western) languages are the most represented, with an exceptional number of Spanish atoms. On the other side, oriental languages are poorly represented. Surprisingly Japan is the language with more atoms per concept, while English is not the one with the highest ratio. This may suggest that there can be many alone atoms poorly associated with concepts.

Language (LAT)	**Concepts**	**Atoms**	**Atoms per Concept**
ENG	4,261,033	9,421,201	2.21
SPA	463,552	1,172,015	2.53
ITA	164,584	239,011	1.45
DUT	155,108	282,772	1.82
FRE	148,352	418,938	2.82
POR	144,020	400,373	2.78
RUS	131,446	281,319	2.14
GER	114,192	223,292	1.96
CZE	90,174	188,790	2.09
KOR	85,199	137,432	1.61
CHI	78,259	78,300	1.00
JPN	70,811	297,224	4.20
TOTAL	4,262,184	13,507,570	3.17

Table 2.1: Quantitative analysis for not suppressed top represented languages

Semantic Types

It is important to know the distribution of semantic types in UMLS to understand the coverage of the types of interest. There are 127 sematic types, so it is evident that grouping is paramount. As shown in table 2.2 and table 2.3, There are so many concepts belonging to categories that are not so useful for the context of interest of the original objective. Indeed, the living beings semantic group will be rarely used. Nevertheless, the coverage of interesting types like disorders, procedures, and anatomies, is still relevant.

Semantic Type (STY)	Concepts
Eukaryote	979.954
Bacterium	420.586
Finding	306.126
Therapeutic or Preventive Procedure	301.753
Organic Chemical	244.965
Plant	223.147
Pharmacologic Substance	175.109
Fungus	160.240
Amino Acid, Peptide, or Protein	158.155
Clinical Drug	128.848
Injury or Poisoning	109.705
Disease or Syndrome	109.660
Clinical Attribute	99.025
Body Part, Organ, or Organ Component	91.063
Gene or Genome	79.825

Table 2.2: Concepts per semantic types

Relations

Relations represent a crucial part of UMLS. There are 11 relationship types[7] and 974 relationship attribute types [Tables 2.4, 2.5]. There are in total 84,350,564 relationships. Even if it might seem a big number, it is important to take into consideration the fact that relations are unidirectional but their symmetric version is always present. Moreover, relations are defined on atoms, so the same relation between concepts can appear multiple times. Sibling relations are the most numerous even if they can be easily derived. So, skipping siblings and halving the relations, leads to 22,534,518 relations. The ratio of

[7]REL abbreviations description `https://www.nlm.nih.gov/research/umls/knowledge_sources/metathesaurus/release/abbreviations.html#mrdoc_REL`

Semantic Group	Concepts
Living Beings	1.951.563
Chemicals & Drugs	929.935
Disorders	642.428
Procedures	435.413
Physiology	169.310
Anatomy	153.633
Concepts & Ideas	92.476
Genes & Molecular Sequences	80.501
Devices	68.811
Objects	24.770
Phenomena	15.035
Activities & Behaviors	5.741
Geographic Areas	4.633
Organizations	4.050
Occupations	2.074

Table 2.3: Concepts per semantic types

relations per concept is 5.29. Considering the complexity of the biomedical domain, this ratio is not so high.

REL	Count
SIB	39,281,528 (19,640,764)
RO	17,834,406 (8,917,203)
SY	7,129,688 (3,564,844)
CHD / PAR	6,077,711
RQ	2,947,678 (1,473,839)
RB / RN	1,856,128
QB / AQ	613,457
RL	62,672 (31,336)

Table 2.4: Relation types count

Sources

The reliability of UMLS derives directly from its source vocabularies. There are some vocabularies that provide more atoms than others and it is interesting to know how many synonyms each vocabulary provides for the same concept. The results of this inquiry are shown in table 2.6. Each vocabulary has a focus

RELA	Count
<not specified>	51,409,014
inverse_isa	2,612,975
isa	2,612,975
translation_of	1,697,294
has_translation	1,697,294
has_inactive_ingredient	1,452,046
inactive_ingredient_of	1,452,046
classified_as	1,242,870
classifies	1,242,870
member_of	1,161,041
… same_as	194,650

Table 2.5: Relevant relation attribute types count

on some particular semantic types and usually one language.

Vocabulary (SAB)	Concepts	Atoms	Atoms per Concept
NCBI	1,907,782	2,085,881	1.09
MSH	404,655	908,129	2.24
MEDCIN	358,221	945,560	2.64
SNOMEDCT_US	352,881	943,751	2.67
SCTSPA	343,591	854,528	2.49
MTH	234,078	235,349	1.01
LNC	216,799	587,633	2.71
ICD10PCS	190,672	269,665	1.41
NCI	161,477	386,147	2.39

Table 2.6: Quantitative concept analysis for the top represented vocabularies

Regarding the relations, the same observation is valid, in table 2.7 the sources that provide the highest number of relations are shown. There are sources that provide many atoms and a few relations and vice-versa.

2.2 Language Models

The Transformer architecture [Vaswani et al., 2017] has set the stage for a multitude of sophisticated models capable of handling a wide range of NLP tasks with significant efficacy. Among these models, Bidirectional Encoder Representations from Transformers (BERT) [Devlin et al., 2018], by Google,

Vocabulary (SAB)	**Relations**
SNOMEDCT_US	6,817,224
SCTSPA	5,993,222
MTHSPL	4,211,900
LNC	3,931,302
NCBI	3,822,748
MSH	3,152,576
RCD	2,929,222
GO	2,553,632
RXNORM	2,474,682
MDRJPN	2,367,490

Table 2.7: Quantitative relations analysis for the top represented vocabularies

has emerged as a notably influential and powerful model that has significantly advanced the state of the art in NLP.

At the heart of BERT is the Transformer architecture, which employs the self-attention mechanism to weigh the importance of different parts of the input text relative to each other. Unlike its predecessors, which processed text in a unidirectional manner, BERT operates bidirectionally. This bidirectional processing enables BERT to capture a more comprehensive understanding of the contextual relationships between words in a sentence.

The pre-training and fine-tuning strategy enables the ease of usage of BERT. Initially, BERT is pre-trained on a vast corpus of text, during which it learns to predict missing words in a sentence, the self-supervised masked language modeling (MLM) task. This pre-training phase allows BERT to learn a rich understanding of language, capturing semantics, relationships between words, and other linguistic nuances. However, the true potency of BERT is unveiled during the fine-tuning phase. Fine-tuning is the process wherein the pre-trained BERT model is adapted for a specific NLP task. During fine-tuning, the parameters of the BERT model are slightly adjusted (some layers can be frozen) on the task-specific data to achieve optimal performance for that particular task. This two-step process of pre-training and fine-tuning not only drastically reduces the amount of training data required for the target task but also leads to models that generalize well on a variety of tasks, showcasing the versatility and efficiency of BERT. It also allows for reduced costs considering that full training of such a big model can be heavily resource-demanding and fine-tuning is relatively lightweight.

The success of BERT led to the development of numerous variants and extensions, each designed to tackle unique challenges or to optimize performance in specific domains. For the sake of this project, the interest is focused on models capable of handling the Italian language and biomedical-specific terminology. Indeed, utilizing a general-purpose, pre-trained model directly for biomedical applications may result in suboptimal performance due to the pronounced distributional disparities between texts from general domains and those from biomedical realms. When there is a significant divergence between the target domain and the corpus on which the model was originally trained, as is the case in biomedicine, the model can greatly benefit from a domain-adaptive training phase, like the BioBERT model [Lee et al., 2020], but it has the drawback of being limited to the English language.

2.2.1 Embeddings

BERT's transformer-based architecture can be used to generate feature vector representations of words or sentences, namely embeddings. The BERT model first encodes the tokenized input text using a stack of encoder layers. Each encoder layer consists of a self-attention mechanism and a feed-forward network. The self-attention mechanism allows the model to learn the relationships between different words in the input sequence. The feed-forward network then applies a non-linear transformation to the output of the self-attention mechanism. to further refine the understanding and representation of each token. The output of the last encoder layer is a sequence of hidden state vectors, each a high-dimensional representation of a token. These hidden state vectors represent the contextual embeddings of the words in the input sequence.

BERT's architecture allows it to learn long-range dependencies in the input text sequence. This is important for understanding the meaning of text, as the meaning of a word or phrase can be affected by words that are far away in the sentence. It is also able to learn the relationships between different words in the input text sequence. This is important for tasks such as semantic similarity and text classification, where the model needs to understand the meaning of the entire sentence.

Pooling

Pooling is a technique employed to aggregate information from a sequence of vectors to produce a fixed-size vector, irrespective of the input sequence's length. This is particularly useful when handling text of varying lengths. In the context of BERT, pooling is often performed on the output embeddings of the tokens to create a single vector representation of the entire input sequence [Ma et al., 2019]. This feature vector can then be used for various downstream tasks. It is important to consider that the input string s is tokenized as [CLS], $s_0, \ldots, s_k$, [SEP] and the BERT model encodes s to a series of hidden states $\mathbf{h}_{\text{[CLS]}}, \mathbf{h}_0, \ldots, \mathbf{h}_k, \mathbf{h}_{\text{[SEP]}}$. Different pooling strategies can be used to obtain the embedding $\mathbf{e} \in \mathbb{R}^l$ with usually $l = 768$.

Mean Pooling It consists of taking the mean of the embeddings of all tokens or a selected group of tokens (e.g. excluding [CLS] and [SEP]): $\mathbf{e}_{\text{max}} = \text{MaxPool}(\mathbf{h}_{\text{[CLS]}}, \mathbf{h}_0, \ldots, \mathbf{h}_k, \mathbf{h}_{\text{[SEP]}})$.

Max Pooling For each dimension of the embedding space, the maximum value is selected across all the token embeddings in a given sequence. This operation results in a single vector that retains the highest activation across each dimension, from all tokens: $\mathbf{e}_{\text{avg}} = \text{AvgPool}(\mathbf{h}_{\text{[CLS]}}, \mathbf{h}_0, \ldots, \mathbf{h}_k, \mathbf{h}_{\text{[SEP]}})$.

CLS Token Pooling BERT pre-training involves training a [CLS] (classification) token whose embedding is intended to be used for classification tasks. Post-training, the embedding of the [CLS] token can be used as a pooled representation of the entire sequence:
$\mathbf{e} = \mathbf{e}_{[\text{CLS}]} = \mathbf{h}_{[\text{CLS}]}$.

2.2.2 Multilingual BERT

Multilingual BERT (mBERT) is a variant of the original BERT model, developed with the capability to process text from 104 different languages, including Italian, through a shared subword vocabulary constructed from the text of all included languages. Unlike having individual models for each language, mBERT operates on a single model architecture, trained on a multilingual corpus, to create a language-agnostic representation space. This unique approach facilitates zero-shot learning across different languages, allowing a model trained on a specific task in one language to be applied to the same task in other languages without requiring additional training data. This feature, known as cross-lingual transfer, is particularly beneficial in scenarios with scarce training data for certain languages.

The mBERT model is pre-trained using similar objectives to the original BERT, including masked language model (MLM) and next sentence prediction (NSP), but on a multilingual corpus, which helps the model learn language-agnostic representations. The applications of mBERT are diverse and significant, especially in cross-lingual NLP tasks such as document classification, named entity recognition, and question answering among others, where it has shown strong performance on various benchmarks.

2.2.3 MedBIT

MedBIT is a model published by the Italian Neuroscience and Rehabilitation Network (RIN) [Buonocore et al., 2023]. Its objective is to provide an effective model for the Italian biomedical context. It is a model based on BioBERT [Lee et al., 2020] trained on Italian biomedical corpora.

Training The model is trained against the self-supervised MLM task with essentially two types of data [Figure 2.1]. The first dataset is derived from the PubMed[8] abstracts undergone through the Google's Neural Machine Translation (NMT) System [Wu et al., 2016] translated from English to Italian, privileging quantity over quality. The authors refer to the model trained only with this machine-translated dataset as BioBIT. Then, MedBIT is obtained from BioBIT by additionally training it with an Italian biomedical corpus derived from reliable resources written by humans. However, the quantity of data is very small but for this training quality is privileged over quantity.

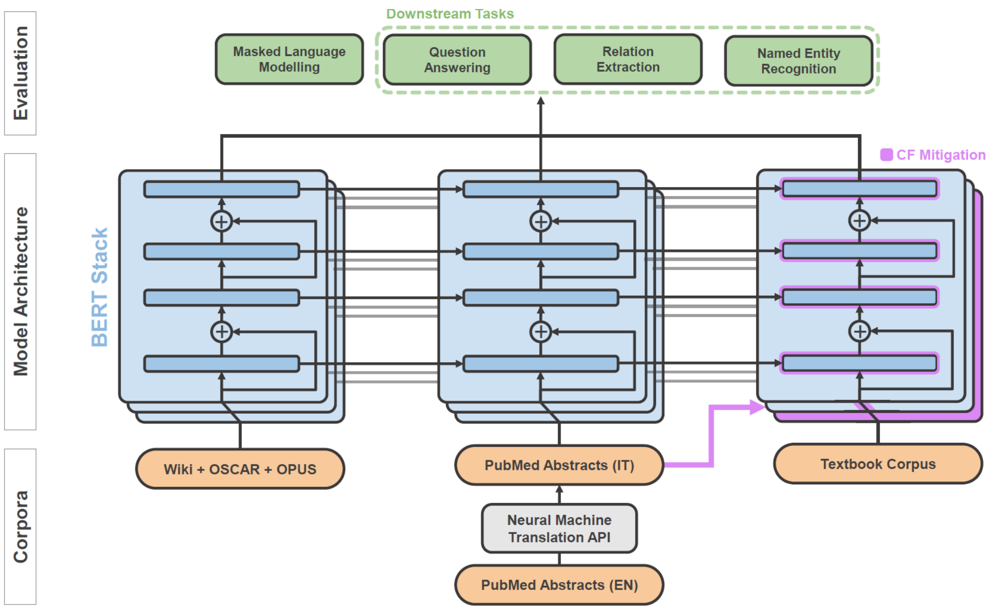


Figure 2.1: BaseBIT, BioBIT, and MedBIT training and evaluation pipeline [Buonocore et al., 2023]

[8] https://pubmed.ncbi.nlm.nih.gov/

Catastrophic Forgetting Mitigation MedBIT was trained with techniques to mitigate the catastrophic forgetting phenomenon, thus avoiding significant deviations from the previously learned parameters. The employed strategies are about learning regularization such as Layer-wise Learning Rate Decay (LLRD), Warmup, and Layer Freezing, alongside knowledge distillation techniques like Mixout and Experience Replay.

LLRD introduces a decay function to the learning rate on a per-layer basis, such that layers situated closer to the input nodes, often encoding more general information, experience a smaller learning rate. The learning rate schedule can be prefaced with a brief warm-up phase, facilitating a smoother transition into the learning process. Layer Freezing procedure nullifies the gradient of the network's earliest layers prior to the last training phase.

On the other hand, the Mixout approach adopts a stochastic procedure to mix the weights of the pre-trained checkpoint with those of the model currently undergoing training. This is intended to augment the stability of language model tuning. Instead, experience replay is realized by furnishing the model with an additional batch of data, sampled from the corpus used for the preceding pre-trained checkpoint (BioBIT), every n steps, the replay frequency.

These diversified techniques, harmoniously integrated, are aimed at ensuring the retention of crucial prior knowledge while adeptly adapting to the peculiarities of biomedical texts.

Evaluation The models have been evaluated with respect to the baseline represented by the BaseBIT model, trained with an Italian general corpus. All the models share the same dictionary, relying upon the BERT tokenization method. The metric to evaluate the MLM performance is the average pseudo-perplexity (PPPL) [Salazar et al., 2020] in equation 2.1, and the Mean Reciprocal Rank of the top five tokens T_5, as shown in equation 2.2. Let N denote the number of tokens within the corpus C being evaluated. Where N is the number of tokens of the corpus C under evaluation, w_t is the word of the sentence S of the corpus C in position t. Then $p(w_t|S_{\backslash t})$ is the conditional probability for the masked word w_t given the other words of the sentence $S_{\backslash t}$. Then $R_{w_{M}S}$ is the rank of the correct word w_M of the sentence S.

$$\mathrm{PPPL}(C) = \exp\left(-\frac{1}{N}\sum_{S\in C}\sum_{t=1}^{|S|}\log p(w_t|S_{\backslash t})\right) \tag{2.1}$$

$$\mathrm{MRR}(C) = \frac{1}{|C|}\sum_{S\in C}\begin{cases}\frac{1}{R_{w_{M}S}}, & \text{if } w_M \subset T_5\\ 0, & \text{otherwise}\end{cases} \tag{2.2}$$

The models are evaluated not only against the MLM task but also against other downstream tasks like Question Answering, Relation Extraction, and Named Entity Recognition, after a fine-tuning round. Tests by the authors show that MedBIT$_{\mathrm{R3}}$+ is one of the best for the cited downstream tasks.

2.2.4 CODER

Contrastive learning on knowledge graphs for cross-lingual medical term representation (CODER) [Yuan et al., 2022] is a model aiming to provide a dense representation in which terms of related biomedical concepts are close, in a multi-lingual manner. It exploits UMLS synonyms and relations [Figure 2.2] to provide high-quality features. For the sake of this thesis, only the CODER$_{\mathrm{ALL}}$ model based on mBERT is taken into consideration.

Based on our research, CODER is the only work that fully utilizes UMLS as a knowledge graph and in a multilingual way. Other models like SapBERT [Liu et al., 2020] utilize only synonyms from UMLS.

Term Normalization Referring to UMLS [Section 2.1], $\mathcal{D} = \{c_i\}_{i=1}^{|\mathcal{D}|}$ is the metathesaurus concept dictionary, and $\mathcal{R} = \{r_i\}_{i=1}^{|\mathcal{R}|}$ is the set of relationships among the concepts in $\mathcal{D}$, represented by triplets $\{(h, r, t)\}$ of an head concept $h \in \mathcal{D}$, a tail concept $t \in \mathcal{D}$, and a relation $r \in \mathcal{R}$. Each concept c_i is defined by a set of terms (represented with strings) $\mathcal{T}_i = \{s_i^j\}_{j=1}^{|\mathcal{T}_i|}$. The embedding of

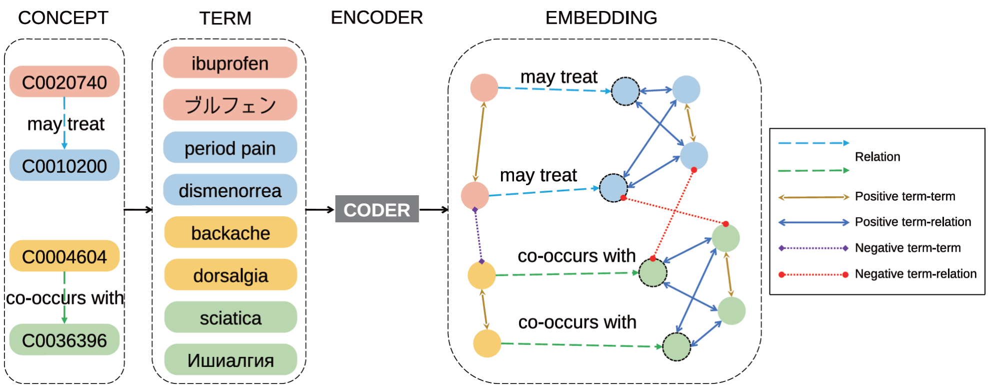


Figure 2.2: CODER term encoding and similarity [Yuan et al., 2022]

s_i^j is referred to as $\mathbf{e}_i^j \in \mathbb{R}^l$ with $l = 768$, obtained with the CLS token pooling [Section 2.2.1]. The objective is to predict the correct concept c of an input term s with embedding $\mathbf{e}$. The predicted concept $\hat{c}$ is given by the concept associated with the most similar term with respect to the given one taking into consideration the cosine similarity of the embeddings: $\hat{c} = c_{\arg\max_i(\cos(\mathbf{e}_i^j, \mathbf{e}))}$.

Training CODER bases the training process on a contrastive learning framework by maximizing the similarity between positive term-term pairs and term-relation-term pairs from the reference KG. At each training step the following steps are performed to compute the loss:

1. A batch of k relationship triplets $\{(h, r, t)\}_{i=1}^k$ is sampled from the KG. To have a uniform notation for concepts, denote $c_i = h_i$ and $c_{k+1} = t_i$. In this way, there are $2k$ possibly not-unique concepts.
2. For each concept c_i, given its set of terms $\mathcal{T}_i = \{s_i^j\}_{j=1}^{|\mathcal{T}_i|}$, only one term s_i is sampled. Thus, there are $2k$ terms.
3. The current CODER model is used to embed the set of terms $\{s_i\}_{i=1}^{2k}$ into the set of embeddings $\{\mathbf{e}_i\}_{i=1}^{2k}$.
4. The term-term pair label τ_{ij} between s_i and s_j is 1 if $c_i = c_j$, 0 otherwise. The similarity of the two terms is given by the cosine of the embeddings $S_{ij} = \cos(\mathbf{e}_i, \mathbf{e}_j)$.
5. The pair-based Multi-Similarity loss (MS-loss) [Wang et al., 2019] is

adopted as depicted in equation 2.3. Given an anchor, the MS-loss deals only with hard negative pairs $\mathscr{N}_i$ and hard positives $\mathscr{P}_i$.

$$\begin{aligned}
\mathscr{N}_i &= \{j | \tau_{ij} = 0, S_{ij} > \min_{\tau_{ik}=1} S_{ik} - \epsilon\}, \\
\mathscr{P}_i &= \{j | \tau_{ij} = 1, S_{ij} < \min_{\tau_{ik}=0} S_{ik} + \epsilon\}, \\
\mathcal{L}_{\text{MS}} &= \frac{1}{2k} \sum_{i=1}^{2k} \Big(\frac{1}{\alpha} \log(1 + \sum_{j \in \mathscr{P}_i} \exp(-\alpha(S_{ij} - \lambda)) \\
&\quad + \frac{1}{\beta} \log(1 + \sum_{j \in \mathscr{N}_i} \exp(\beta(S_{ij} - \lambda)) \Big)
\end{aligned} \tag{2.3}$$

6. The term-relation-term label τ_{ij}^{rel} between s_i and s_j is 1 if $(h_i, r_i, t_{j-k} \in$ KG, 0 otherwise. The similarity is given by the multiplication of the relation mapping matrix $S_{ij}^{\text{rel}} = \cos(\mathbf{M}_{r_i}^{\top} \mathbf{e}_i, \mathbf{e}_j)$, where $\mathbf{M}_{r_i} \in \mathbb{R}^{l \times l}$.
7. With the same principle at step 5 the MS-loss of term-relation-term $\mathcal{L}_{\text{MS}}^{\text{rel}}$ is computed.
8. The final loss is given by $\mathcal{L} = \mathcal{L}_{\text{MS}} + \mu \mathcal{L}_{\text{MS}}^{\text{rel}}$ with μ a hyperparameter.

Evaluation CODER was evaluated by the authors against some medical term normalization tasks with English datasets in a zero-shot setting. Based on their experiments comparing other popular BERT-based models, it is shown that for this task CODER$_{\text{ENG}}$ performs significantly better than CODER$_{\text{ALL}}$. Moreover, contextual pre-trained MLM embeddings do not succeed in achieving embedding-based normalization, and medical word embeddings do not provide improvements over the word embedding baseline. Another test was done with non-English datasets comparing CODER$_{\text{ALL}}$ with mBERT, resulting in the former outperforming by far the latter.

Some other tests, involving t-SNE visualization of different contextual embeddings, clearly show that CODER clusters and groups concepts concerning the semantic types.

CODER++

CODER employs contrastive learning to generate embeddings that bring closely related terms within a conceptual vicinity. While these embeddings have exhibited promising results in various applications, they are not without limitations. The main limitation is their insensitivity to minor textual differences, showing high similarities for non-synonymous but textually similar terms. This insensitivity can lead to suboptimal results when clustering biomedical terms, where precise distinctions are often crucial. An upgraded approach named CODER++ has been conceived [Zeng et al., 2022]. CODER++ addresses the need for fine-grained representations in the realm of term embeddings. It introduces sampling that dynamically selects hard positive and negative samples during contrastive learning. This strategic adjustment allows CODER++ to capture subtle nuances in term semantics, resulting in more accurate and effective biomedical term clustering. CODER++ was pre-trained by the authors only for the English language and at the moment there is no multilingual CODER++ version.

Hard Negative Sampling The advancement of CODER++ consists of the usage of dynamic hard negative samples during pre-training with the objective of generating better term clustering, by pushing further semantically different but syntactically similar terms. For each term s_i, k positive terms $p_{i_1}, \ldots, p_{i_k}$ with the same CUI of s_i are sampled. The textual difference between s_i and a positive sample p_i is an indicator of how hard the positive sample is. The most relevant strategy is about the negative sampling, indeed, for each term s_i, m negative hard terms $n_{i_1}, \ldots, p_{i_m}$ are drawn by looking at the current nearest neighbors of s_i. These neighbors if have a different CUI of s_i, constitute the hard negative samples. Because it is expensive to look for all the embeddings of all the terms in the KG, a vector library [Section 3.3] is used and an index of embeddings is updated at each epoch with the new results of the model under training. The loss, as in CODER, is the MS-loss. This aims at pushing further negative terms that the model considers as positives, in a more effective way.

2.3 Related Works

In literature, several attempts have been made towards the biomedical entity linking goal, often referred to as medical concept normalization. The employed techniques evolved as the technologies in NLP did. Among the scientific community, UMLS stands out as a highly regarded and dependable resource. Nevertheless, the field of research still awaits a substantial multilingual breakthrough, despite some noteworthy efforts that have been made in this direction.

MetaMap MetaMap [Aronson, 2001, Aronson and Lang, 2010] is a tool developed by the National Library of Medicine (NLM) that annotates biomedical texts against the UMLS metathesaurus. This tool conducts a series of linguistic analyses, including tokenization, part-of-speech tagging, lexical lookup, and syntactic parsing. It also generates variants of phrases, identifies candidate matches in the Metathesaurus, constructs mappings, and offers word sense disambiguation (WSD).
MetaMap's evaluation process relies on four linguistic measures: centrality, variation, coverage, and cohesiveness. It boasts a robust lexicon, a relaxed data model, and efficient processing options. However, MetaMap is limited to English text and has room for improvement in terms of disambiguating ambiguous terms.
Despite its strengths, MetaMap's reduced accuracy in handling ambiguity remains a challenge. That is because MetaMap can be considered obsolete nowadays with respect to NLP advancements. Purely syntactic methods are insufficient to reach high-level performances. It is now often used as a baseline for these types of problems.

Classification The entity linking problem can be framed as a classification one [Miftahutdinov and Tutubalina, 2019]. The classification problem addressed in this research revolves around the mapping of entity mentions from User-Generated Texts (UGTs) to specific medical concepts within UMLS. This linguistic-based system employs lexical lookups and variants to assign scores to phrases within a sentence. This process involves converting textual mentions into numerical representations using neural network models, capturing their underlying semantic meaning. To enhance this mapping, domain-specific knowledge from UMLS is incorporated, allowing for the computation of semantic similarity features. A classification algorithm then assigns probability scores to potential medical concepts for each mention, ultimately selecting the concept with the highest probability as the classification result. The aim is to automate the normalization of UGT entity mentions into standardized medical terminology, which holds significant promise for applications in healthcare and biomedical research.
Although state-of-the-art embedding mechanisms and language models such as BERT were used, the huge number of classes still represents an impairment for such an approach. Moreover, the classification approach is also inflexible with respect to classes modifications, requiring new training.

Ranking The ranking approach [Sung et al., 2020] focuses on ranking candidate concepts based on their similarity to the input term. Unlike the classification method, this approach uses a binary classifier, where positive samples consist of terms paired with their corresponding concept names, and negative samples consist of terms paired with non-corresponding concept names. The classifier output serves as a measure of similarity used to rank candidate concepts for normalization. In ranking methods, the goal is to assess the similarity between the input term and candidate target terms by training them as positive and negative pairs. For instance, DNorm [Leaman et al., 2013] learns to rank target terms by calculating similarities between TF-IDF vectors. While

CODER [Yuan et al., 2022, Zeng et al., 2022] [Section 2.2.4] uses pre-trained BERT-based embeddings and contrastive learning to generate a representation space for UMLS terms and then find the nearest neighbors of the queries.

Italian Annotator Relevant efforts have been made by the Italian community towards the annotation problem [Attardi et al., 2015]. This approach uses three specialized Named Entity Recognizers (NERs) for extracting mentions of body parts and treatments, other clinical entities, and measurements. Tanl NER [Attardi et al., 2009], a statistical sequence tagger, was employed. The tagger utilizes a Conditional Markov Model [McCallum et al., 2000] and offers various configuration options for classification algorithms and feature extraction templates. Different feature sets were experimented with, such as word shape features, dictionary features, prefixes, suffixes, bigrams, last words, first words, and frequent words extracted from the training data.
The issue of a lack of Italian annotated corpus forced to rely on automatically annotated data. Impressive results are shown with an F1-score above 96%. But, as the authors warn, it's important to note that these accuracy results are indicative, primarily due to the automatic annotation bias in the corpus.

Medical Conceptual Similarity Measure A metric for embeddings evaluation of their clusterization with respect to the semantic types has been devised [Choi et al., 2016]. Given a set of concepts C, a semantic type T, and the number of nearest-neighbors k MCSM is defined in equation 2.4.

$$\text{MCSM}(C, T, k) = \frac{1}{|C(T)|} \sum_{c \in C(T)} \sum_{i=1}^{k} \frac{1_T(c(i))}{\log_2(i+1)} \tag{2.4}$$

Where $C(T) \subset C$ is the set of concepts of semantic type T, $c(i)$ is the i^{th} closest neighbor of c, and 1_T is the indicator function of T. If $k \to \infty$, the measure becomes meaningless, and on the opposite, if k is excessively small, it leads to an excessive variance.

Chapter 3

Methodology

The inherent complexity of medical records, which deals with various non-standard names, abbreviations, and misspellings, necessitates the standardization of these terms. The process of term normalization can be approached either as a classification or ranking problem. Classification methods, while achieving impressive results, face limitations when dealing with the vast number of concepts within terminology systems like the UMLS. In contrast, ranking methodologies excel in handling extensive and flexible concept sets.

In the process of linking biomedical terms to the UMLS metathesaurus, several critical challenges need to be overcome. These challenges primarily revolve around ensuring accurate contextual comprehension, disambiguating terms that are lexically similar but differ in meaning, and addressing the scarcity of resources in the Italian language. Furthermore, developing a robust, trustworthy, and efficient system is crucial, especially considering its practical applications in various industries.

Therefore, this research focuses on the entity linking task, treating biomedical term linking against the UMLS as a ranking problem, with a particular emphasis on its seamless integration into the Clinika architecture pipeline, by exploiting the already-built functionalities. In particular by integrating the entity linking part of the current document annotation pipeline [Figure 3.1] [Barbieri, 2023].

The assertion of this study is that through intelligent utilization of the semantic relationships (eventually hierarchical) within UMLS and the syntactic relationships within the free documents, it becomes feasible to execute reliable and precise entity linking. Precision is prioritized because, for the context domain, it is extremely important not to have false positives.

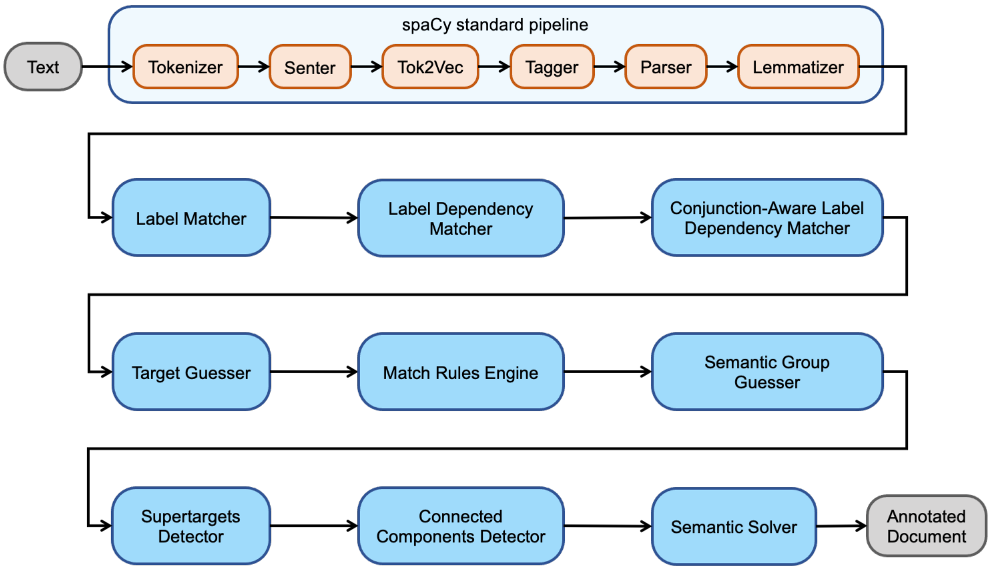


Figure 3.1: Clinika annotation pipeline [Barbieri, 2023]

3.1 Entity Extraction

The final objective is to produce an annotated document, with its set of associated UMLS concepts, derived from a medical text. The first requirement is to identify and extract relevant segments of text in the document that are worthy of being coded, the so-called mentions. They usually consist of disease names, treatments, body parts, medicines, medical devices, or surgeries. This task is performed by the entity extraction components (from the beginning to the match rules engine) of the Clinika annotation pipeline designed and implemented by other colleagues [Figure 3.1] [Barbieri, 2023]:

1. **SpaCy Standard Pipeline** SpaCy[1] is a popular Python library for NLP that offers a standard processing pipeline for various tasks. It consists of several components, that are mainly used here for syntactic analysis. These components work together in a sequential manner, with each one building on the output of the previous step.
   (a) **Tokenizer** It is in charge of breaking down the input text into individual tokens. It segments the input text into words, punctuation, and other meaningful units.
   (b) **Senter** The senter, short for sentence segmenter, is responsible for splitting the text into sentences. It identifies sentence boundaries based on punctuation marks and context.
   (c) **Tok2Vec** It is a feature extractor. It converts each token into a dense vector representation, capturing semantic information about the token in the context of the entire document.
   (d) **Tagger** It is responsible for assigning part-of-speech (POS) tags to each token in the text. Part-of-speech tagging helps identify the grammatical roles of words in sentences, such as nouns, verbs, adjectives, and adverbs.
   (e) **Dependency Parser** It performs dependency parsing by analyzing the grammatical structure of sentences and determining how words relate to each other within a sentence [Figure 3.2]. This leads to the creation of a dependency tree, where each word is linked to its syntactic head.

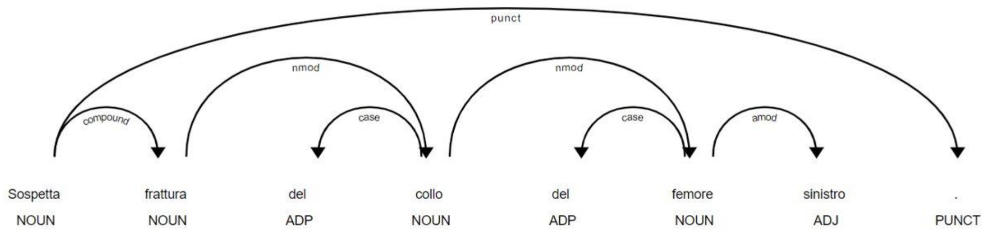


Figure 3.2: Dependencies for an Italian medical sentence [Barbieri, 2023]

[1]https://spacy.io/

   (f) **Lemmatizer** It reduces words to their base or dictionary form, by standardizing word forms and limiting inflectional variations.

2. **Label Matcher** It is a custom component in the pipeline designed to identify UMLS labels within a given medical text and assign the respective Concept Unique Identifiers (CUIs). This search is performed on both the original text and the lemmatized one. The Label Matcher leverages the spaCy PhraseMatcher component, which identifies a list of patterns within a document. UMLS labels have been converted into spaCy documents, in order to leverage the efficiency of the PhraseMatcher. This was done with exceptional attention to the tokenization.
3. **Label Dependency Matcher** It is designed to address a limitation of the Label Matcher. Indeed, it can only find labels that appear contiguously in the text. For instance, to find the label 'fracture of the femur' from the text 'fracture of the neck of femur' it's essential to have the more specific label in UMLS. It cannot discover the broader label if the specific one is missing. To overcome this limitation, the Label Dependency Matcher searches for labels within the text's syntactic tree.
4. **Conjunction-Aware Label Dependency Matcher** It is tailored for detecting labels linked by conjunctions within the syntactic tree of documents. This component unveils hidden syntactic relationships by modifying the input document's tree to make them explicit, allowing for the extraction of all associated labels. It enhances the Label Dependency Matcher's capabilities by handling conjunctions effectively and unifying the results with previously discovered labels. For instance, for the text 'fracture of the tibia and fibula', it is possible to generate the two mentions 'fracture of the tibia' and 'fracture of the fibula'.
5. **Target Guesser** To overcome these issues about sensitivity to typos, tokenization differences, and variations in labels, a custom component called the Target Guesser, based on spaCy's SpanCategorizer and a predictive model, identifies and categorizes relevant text spans. It deserves

deeper attention in the following paragraphs [Section 3.1.1].

6. **Match Rules Engine** The Match Rules Engine serves to address scenarios where matches found by previous components do not correspond to desired text spans or where some spans remain undetected due to not being present in the knowledge base. This component allows to specify custom rules for manipulating matches based on certain conditions, including token and span attributes like part-of-speech, lemma, or syntactic dependencies. Each rule applied by this component includes an 'action' field that dictates how it should affect the identified matches:
   (a) **Remove** Removes an undesired match.
   (b) **Set** Sets a specified concept for a match, replacing any existing concepts. This helps disambiguate terms with multiple meanings.
   (c) **Add** Adds specified concepts to a match, without removing existing concepts. It's useful for adding not present synonyms.

3.1.1 Target Guesser

The Target Guesser component [Barbieri, 2023] is crucial for detecting mentions that are not perfectly matched with UMLS. Indeed, the scope of the project of this thesis is to provide the best possible concept for a mention, even in the absence of a direct label. Without the target guesser, this wouldn't have been possible. It is based on the SpanCategorizer spaCy component. The SpanCategorizer is based on a predictive model that detects relevant portions of text and classifies them. For the sake of the target guesser, only one class is needed. It is similar to a Named Entity Recognizer (NER) but it is able to handle also overlapped portions of text. The spaCy SpanCategorizer is made up of a suggester and a classifier. The suggester is a customizable function that extracts all candidate spans. This function can be rule-based or use machine learning approaches. The classifier's role is to predict the correct class for the extracted spans provided by the suggester, considering their surrounding context. The classifier is composed of an embedded, a pooler, and a scorer. Although the single-class usage of the target guesser, the classifier becomes useful if used in a binary way to decide whether to emit or not a target. Indeed, only targets with a score higher than 0.5 are emitted. The custom SpanCategorizer is based mainly on LSTM networks [Hochreiter and Schmidhuber, 1997] in a bi-directional fashion.

Corpus Because of the lack of Italian resources for this specific context, a manually annotated corpus was used to train the SpanCategorizer. In particular, 468 medical reports were annotated with $6,000$ targets globally. The annotation tool used is Prodigy[2] which perfectly integrates with spaCy. The first 15% of the documents have been annotated completely manually, then an initial draft model was trained in order to automatically annotate the rest of the documents and manually correct the generated targets, instead of annotating everything from scratch.

Performances The evaluation metric is done in a word-based tokenized way, to provide a better reliability of the results. Indeed, if all the target text is considered, the model gets penalized even if the detected target is almost correct. In table 3.1 it is shown how a metric that considers tokens is more effective in the evaluation. The target guesser is considered to be highly reliable. Indeed, it reaches a 98.44% precision, a 96.75% recall, a 97.58% F1-score.

Text	the radiography shows a displaced fracture				
Tokens	the	radiography	shows	a	displaced	fracture
True Targets	✗	✓	✗	✗	✓	
Detected Targets	✗	✓	✗	✗	✗	✓
Standard metric	TN	TP	TN	TN	FN	FP
Tokenized metric	TN	TP	TN	TN	FN	TP

Table 3.1: Target Guesser evaluation metrics example

[2]`https://prodi.gy/`

3.1.2 Connected Components

The Clinika pipeline is able to provide relevant information about the text to be annotated. It first detects the components that are worth to be annotated and then determines the compatibilities among them. Using the rules it is possible to extract also mentions that are not directly present in UMLS (e.g. 'frattura del collo del femore'[3]) [Figure 3.3].

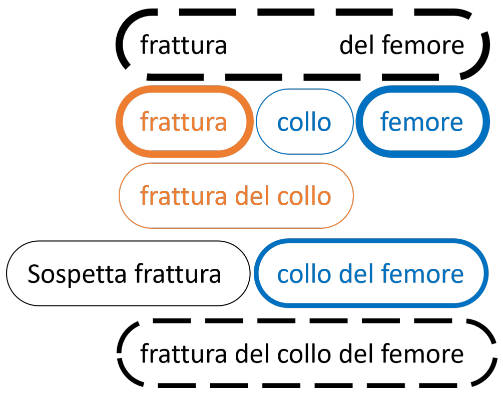


Figure 3.3: Mentions of sentence in figure 3.2 [Barbieri, 2023]

In the end, it is important to emit one or more concepts but not all the detected mentions have to be annotated. Indeed, the best combination of mentions should be found, e.g. it is not possible to emit both 'collo'[4] and 'collo del femore'[5], they are clearly incompatible. This concept of compatibility [Barbieri, 2023] is better formalized by the pipeline, declaring as incompatible mentions that are in the same conflict set [Figure 3.2]. Two or more matches are considered compatible if the text spans they cover do not have any tokens in common. It is then possible to concatenate the mentions of multiple compatible matches, producing a unified text string for generating a new mention. Concatenation options include using the original texts from the document, utilizing Italian preferred labels, or employing English preferred labels. In this way, it is possible to perform additional reasoning to emit a reliable solution.

[3] Italian for 'femoral neck fracture', even if this label exists, let's pretend that it does not
[4] Italian for 'neck'
[5] Italian for 'femural neck'

	frattura	collo	femore	frattura del collo	collo del femore	sospetta frattura	frattura del femore	frattura del collo del femore
frattura		✓	✓		✓			
collo	✓		✓			✓		
femore	✓	✓		✓		✓		
frattura del collo			✓					
collo del femore	✓					✓		
sospetta frattura		✓	✓		✓			
frattura del femore								
frattura del collo del femore								

Table 3.2: Compatibilities of sentence in figure 3.2

3.2 Vector Representation

Because of the great number of concepts and atoms, it is important to have a dense representation of them in a high-dimensional vector space $\mathbb{R}^l$, usually with $l = 768$. The embedding system has to be inductive, allowing for the representation of strings that are not part of the training set. Moreover, the inference has to be fast. To fulfill these needs, a BERT-based language model is perfectly suitable [Section 2.2].

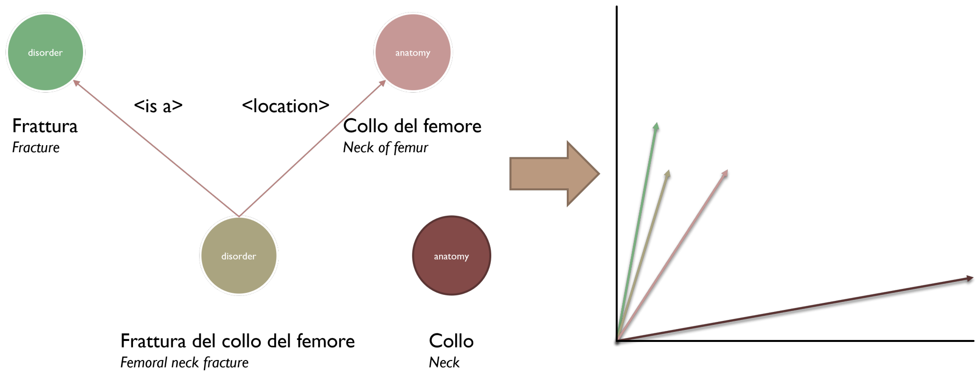


Figure 3.4: Embedding desired example with $l = 2$

The desired terms in the embedding space should ideally be grouped on their semantic types [Figure 3.5]. Synonyms should be closer and they should also capture hierarchical relations. To solve ambiguity issues, the embeddings should relate terms also on the basis of their context and the relations that are not hierarchical. For instance, as shown in figure 3.4, the ‘neck’ should be quite distant from the ‘neck of femur’ even if it shows a high lexical similarity. Moreover, supposing that the ‘femoral neck fracture’ is not in the knowledge (UMLS metathesaurus) but there is ‘fracture’ and ‘neck of femur’, the model is supposed to understand that the ‘femoral neck fracture’ is a specification of ‘fracture’ and its location is the ‘neck of femur’. The ‘neck’ alone should show a low similarity because it is an anatomical part distant from the ‘femur’ and it is not involved at all.

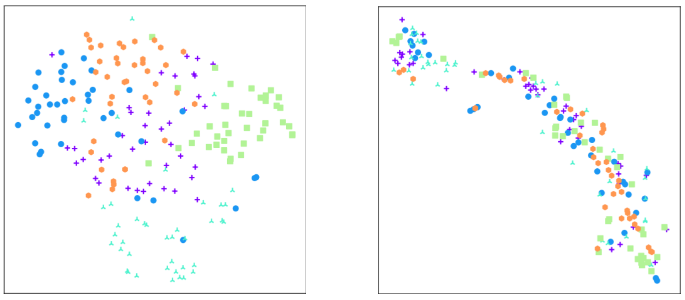

(a) CODER$_{\text{ENG}}$ (b) BioBERT

Figure 3.5: t-SNE for concepts of different semantic types [Yuan et al., 2022]

3.2.1 Vector Load and Storage

In order to build a ranking system, it is essential to have access to the knowledge base in a rapid way. For this reason, it is crucial to have a system that given a string returns the relative embedding with the smallest latency possible. Before storing the embeddings, they should be generated from the terms of the knowledge base. Even if this task might seem straightforward, it is not because of the huge amount of data in the knowledge base [Section 2.1.1]. The full UMLS weighs about 30 GB with 13,507,570 atoms. It is evident how it is difficult to work with in-memory solutions like Pandas[6].

Load UMLS has been deployed on a MySQL database by MAPS S.p.A. leading to the possibility of using SQL cursors. However, difficulties have

[6]`https://pandas.pydata.org/`

been experienced with the low reactivity of MySQL. The solution adopted is Dask[7] for Python. Its efficiency when dealing with large files is given by the following techniques:

1. **Parallelization and Scalability** It is designed to handle parallel and distributed computing. When reading large datasets, it can split the work across multiple CPU cores or across multiple nodes in a cluster.
2. **Lazy Evaluation and Out-of-Core Processing** It doesn't load the entire dataset into memory at once. Instead, it creates a computational graph representing the data processing steps to be performed. As a result, Dask can efficiently read and process data in small, manageable chunks, minimizing memory usage.
3. **Optimized Backends** It leverages optimized backend libraries for file reading, such as fsspec and fastparquet, which are designed for efficient I/O operations.
4. **Data Structures** It uses its own parallelized data structures, which are optimized for distributed computing.

Dask's ability to parallelize, lazily evaluate, and efficiently manage memory, along with its optimized backends and scalability options, make it a sensitive choice for reading and processing large datasets.

Storage Embeddings need to be stored in a compact and rapidly-retrievable way. Considering $l = 768$, a 8 bytes representation for numbers in $\mathbb{R}$, and $13,507,570$ atoms, the memory allocation would end up with an overall ~ 83 GB representation. However, embeddings are unique for each string and not for each atom, so it is possible to consider only the $11,674,270$ unique strings. With such high dimensional space, the contribution of precision between float32 and float64 representations is not so important. Thus, using 4 bits for real numbers and only the unique strings, the required memory drops to

[7]`https://www.dask.org/`

$\sim$ 36 GB. Further reductions could be performed with an NN-based embedding compressor or a memory-efficient storage system.
For the sake of this project, it is essential to have a low-latency reading embedding database. LevelDB[8] was chosen for this purpose. It is an open-source, high-performance key-value storage library developed by Google, designed for efficiency, reliability, and versatility. LevelDB leverages Bloom filters [Bloom, 1970] to reduce the number of disk reads when looking up a key. LevelDB organizes data on disk in a series of sorted files. Before reading a file, it checks the Bloom filter for that file. If the Bloom filter suggests that the key might exist in the file, LevelDB proceeds with a disk read operation. If the filter indicates that the key is definitely not in the file, LevelDB avoids reading the file, which can significantly reduce I/O operations and improve lookup performance. The Bloom filter is initialized with a fixed size, that influences the number of false positives. When an element is inserted into a Bloom filter, it goes through multiple hash functions, each of which produces a different index within the filter's array of bits. These bits are then set to 1 in the filter. Instead, when checking if an element exists in the set, it gets hashed with the same hash functions used during insertion. If all of the corresponding bits in the filter are set to 1, the filter indicates that the element might exist in the set. If any of the bits are 0, then the element definitely does not exist in the set. Bloom filters are fast for both insertion and membership tests because they involve simple bit manipulations and hash computations.

3.3 Vector Indexing

Once the embeddings of the knowledge base are generated and stored, a system able to perform searches on this dataset is paramount. There are several vector databases and libraries that suitable for this nearest neighbors search

[8]`https://github.com/google/leveldb`

task. Annoy[9] (Approximate Nearest Neighbors Oh Yeah) was chosen because it is almost as fast as the other solutions and its indexes are portable. Annoy is a popular open-source library designed by Spotify to efficiently search for approximate nearest neighbors in high-dimensional spaces. It was created to address the problem of finding approximate nearest neighbors quickly, which can be computationally expensive for large datasets and high-dimensional data using traditional methods. It is designed for efficiency and can perform ANN search significantly faster than brute-force methods. However, the results are approximated, depending on the chosen parameters the quality of the approximation may vary.

It also supports multiple distance metrics (Euclidean, Manhattan, Angular, Hamming, or Dot Product). The angular distance, is computed by the Euclidean distance of normalized vector, i.e. $d_{\text{ang}}(u, v) = \sqrt{2(1 - \cos(u, v))}$.

The Annoy workflow is the following:

1. **Forest** Annoy builds a data structure called a ‘forest’ or ‘forest of random projections’. This data structure consists of a collection of binary trees, each of which is responsible for partitioning the dataset into smaller subsets.
2. **Random Projections** Annoy employs random projections to divide the data into different regions. Random projections are a type of dimensionality reduction technique. They project high-dimensional data onto lower-dimensional subspaces in a randomized manner. By using random projections, Annoy creates a set of hyperplanes that divide the dataset into smaller parts.
3. **Construction** During the construction phase, Annoy builds the forest. Each tree is constructed independently. For each tree, a random subset of the data points is chosen, and then a binary tree is built based on the random projections. The number of trees and the size of the subsets are

[9] `https://github.com/spotify/annoy`

parameters that can be adjusted. These parameters determine at write-time the trade-off between speed and accuracy. Also, the distance metric must be chosen at write-time.

4. **Query** When it is needed to find the k-nearest-neighbors of a query point, Annoy traverses each tree in the forest to search for potential candidates. It uses the random projections to quickly eliminate large portions of the dataset that are not relevant to the query. It returns a set of candidate points that are likely to be approximate nearest neighbors. These candidates are selected based on their proximity to the query point in the lower-dimensional space defined by the random projections.
5. **Ranking** The library calculates the actual distances between the query point and the candidates in the original high-dimensional space. It ranks the candidates by their true distances, allowing you to retrieve the top-k nearest neighbors.

Index File Annoy uses a binary file format to store the data structure it constructs during the indexing phase. This binary file contains all the information needed to perform approximate nearest neighbor queries. This file format is platform-independent, making it suitable for cross-platform compatibility. It provides built-in functions to serialize (save) and deserialize (load) its data structures to and from files. Annoy's file portability makes it convenient for sharing pre-trained indexes without recomputing them.

Position Alignment The main issue to deal with when working with vector libraries like Annoy, is the key-alignment. Annoy indexes vectors based on a key called position. The position is a progressive integer number. Because of this, an external data structure that links positions with the actual embedded string is essential. To this purpose, it was decided to employ a MySQL database with a look-up table indexed on this position, containing also other information with respect to the UMLS knowledge base. Maintaining this table consistent with the Annoy index is a full responsibility of the developers.

Performance Parameters Tuning Annoy involves primarily two key parameters: the number of trees, denoted as 'n_trees', and the number of nodes to inspect during searching, known as 'search_k'. The number of returned neighbors will be referred to as k, but it does not affect performances. The choice of values for these parameters should align with the application's need for accuracy, index size, and query performance.

1. **Trees** The 'n_trees' parameter is specified during the index construction phase and has a significant impact on both the build time and the resulting index size. Indeed, Increasing the value of 'n_trees' improves the accuracy of query results by building a more refined and diversified set of trees. However, the drawback is that it also results in larger index files and longer build times.
2. **Nodes to Inspect** During the nearest neighbor search, the 'search_k' parameter determines the maximum number of nodes to examine for potential candidates. A higher value for 'search_k' increases the likelihood of finding accurate nearest neighbors but extends query times.

3.4 Ranking and Aggregation

A ranking system is built using the discussed components. When it is needed to find the nearest neighbors of a string, this string should be embedded with the same model used for the creation of the Annoy index. Then. the vector of this string can be directly passed to the Annoy query system to retrieve the positions of the k-nearest-neighbors and their distances. These positions should be decoded using the look-up knowledge base positioning table.
However, it is important to have a similarity score associated with each neighbor to ease the reasoning. Indeed, the first step of the ranking system is to convert the distances into scores, e.g. for the angular similarity:

$s_{\text{ang}} = -(d_{\text{ang}}^2/2) + 1$. Then, a temporary rank $\hat{r}$ is assigned by sorting the results by the similarity in descending order.

The results might likely be relative to more concepts. Because the desired output is a concept, it is possible to exploit the obtained ranking of labels to derive a ranking of concepts. Let k be the number of retrieved labels, n be the number of different concepts associated with the retrieved labels, L be the set of retrieved labels, c_i be the concept associated with label i, $\hat{s}_i$ be the score of label i, and be s_j the score of concept j. Several strategies can be adopted, each with different efficiency and drawbacks:

1. **Max** The retrieved labels are grouped by their CUI and the maximum score within the group is selected.
2. **Sum** The retrieved labels are grouped by their CUI and the within-group scores are summed up. Each label's score is treated as a vote for its associated concept. Then, each summation is divided by k. In this way, the score meaning is preserved, while a normalization to 1 would end up with a probability-like result.
3. **Weighted Sum** The sum is weighted on the basis of the ranking position $\hat{r}_i$ of the label. A new label score is computed $\hat{s}_{t_i} = \hat{s}_i(k - i + 2)^{\alpha}$ where α is a weighting parameter for the ranking relevance. Then, the new label scores $\hat{s}_{t_i}$ are summed up like at point 2.
4. **Hierarchical** If resulting concepts and labels are organized hierarchically, forming a taxonomy, label scores can be aggregated hierarchically, with parent concepts inheriting scores from their child labels. However, this assumption does not always hold in this context. This method provides context-aware rankings.
5. **Machine Learning** Advanced machine learning models, such as neural networks or gradient boosting, can be trained to predict concept scores based on label scores. Additional features, context, or metadata can also be incorporated into the model, like the information provided by the entity extraction pipeline. This approach is suitable for applications

where traditional methods may not capture the nuanced relationships between labels and concepts.

6. **Lexical Similarity** The lexical similarity between the target mention and the nearest neighbors can be used to assign an additional score. For instance, the Damerau–Levenshtein distance [Damerau, 1964] is a string similarity metric that measures how different two strings are by considering four basic operations: insertions, deletions, substitutions, and transpositions (swapping adjacent characters). It is calculated through a dynamic programming approach using a matrix where each cell represents the distance between the corresponding substrings of the two input strings. Initially, the matrix is filled with distances for individual characters. Then, by considering adjacent cells, the algorithm computes the minimum distance needed to transform one string into the other, accounting for the four operations. However, due to the sensitivity to minor typographical errors of lexical similarity measures, the scores are not highly reliable in this context.

It is important to underline that these ranking mechanisms depend on the number of labels per concept in the knowledge base. If there is a specific correct concept but with only a label, and there is a less specific concept with more labels, this last one could override the most correct one. However, the similarity measure should be higher for the more specific label. Thus, using a weighted sum rather than the normal sum, or other advanced techniques, is extremely relevant.

All these approaches should be equipped with a threshold that can be applied either to the label scores s_i or to the concept scores c_j, or even both. The adopted threshold is fixed to not add an additional overhead.

3.5 Graph Representation

The UMLS dataset contains valuable relational knowledge that enhances our inference process. Instead of solely relying on embedding values, we incorporate the semantics of the index results, utilizing relationships among atoms. These relationships encompass a predefined set of standardized relations, along with their respective specifications. Additionally, there exist asserted hierarchical relations within the dataset. To facilitate reasoning and analysis of this relational knowledge, we conducted a seamless migration of UMLS into Neo4J[10]. This migration allows to effortlessly traverse the graph structure and apply algorithms for further insights and applications.

Neo4j is a schema-free graph database management system. It is designed to efficiently store, manage, and query data in a graph format. Unlike traditional relational databases that use tables and rows, Neo4j is specifically optimized for working with data that has complex relationships. It has a native graph data model, which consists of nodes, relationships, and properties. Nodes represent entities, relationships connect nodes and define how they are related, and properties store additional information about nodes and relationships. Neo4j uses Cypher, a powerful and expressive query language specifically designed for querying graph data, allowing retrieval and manipulation of data patterns within the graph. Neo4j is ACID compliant, ensuring data integrity and reliability even in the face of concurrent transactions. Finally, Neo4j offers a variety of efficient graph algorithms that are commonly used in data science and analytics. It is worth mentioning that every relationship in Neo4J must be unidirectional, however, at read-time, it is possible to consider the relationships as undirected. Neo4j offers various indexing methods, including label indexing, property indexing, full-text indexing, and spatial indexing. Indexes help quickly locate nodes, relationships, or properties based on specific criteria, significantly improving query performance and making Neo4j

[10]`https://neo4j.com/`

suitable for complex graph data analysis.

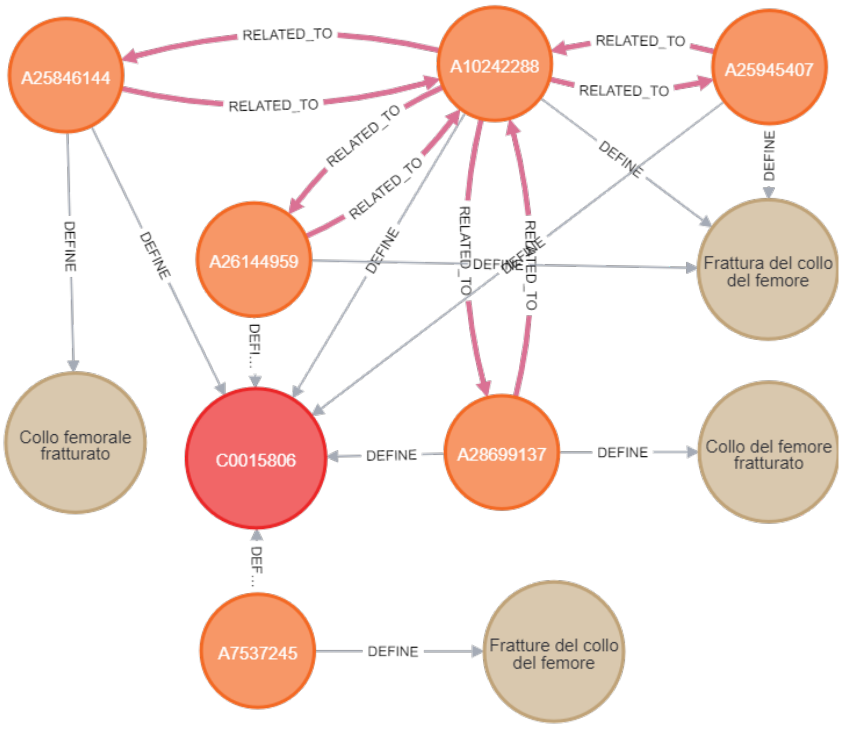

A25846144
A10242288
A25945407
A26144959
C0015806
A28699137
A7537245
Collo femorale fratturato
Frattura del collo del femore
Collo del femore fratturato
Fratture del collo del femore
RELATED_TO
DEFINE

Figure 3.6: Graph representation of a concept and its labels

For the migration, it is necessary to identify the nodes, the relationships, and their properties. Not everything of UMLS was migrated, but only the relevant part of the metathesaurus. The fundamental components are the atoms. Everything revolves around them. Each atom, identified by an AUI, defines a concept node and a string node. Concepts and strings can be defined by multiple atoms [Figure 3.6]. The relationships are defined between atoms and are symmetrically duplicated. Then, there are nodes for the possible semantic types and groups, concepts are associated with semantic types with a belonging relationship and so are the semantic types with the semantic groups. Finally, indices were added to the most queried properties and labels.

For read-optimization purposes, relationships are written redundantly on the concepts. In this way, it is possible to run algorithms directly on concepts without the overhead of traversing the atoms, significantly improving the response time.

In Neo4J, an essential tool is the shortest path algorithm, used to find the most concise route between two nodes in a graph. Neo4j provides the APOC (Awesome Procedures on Cypher) library, which extends the functionality of Neo4j, including the shortest path algorithm. It is possible to perform more advanced shortest-path queries, such as finding paths within a specific distance, considering node properties, or considering weighted relationships.

3.6 Architecture

The designed architecture is meant to be modular, expandable, pipelined, and easily accessible. The high-level architecture depicted in figure 3.7 shows the dependencies between the various components.

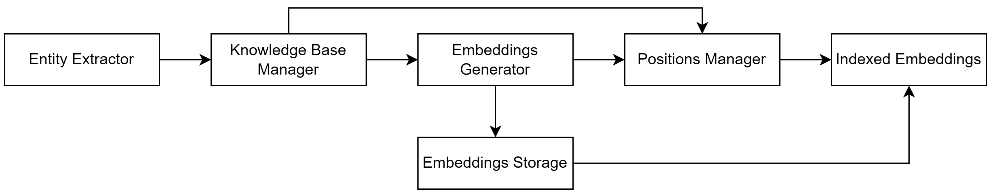


Figure 3.7: High-level service architecture

For the sake of having a modular and reliable implementation, an object oriented infrastructure was built [Figure 3.8]. The broad principle is to have a general interface for each component that can then be implemented with its own concrete logic. The functionalities are the following:

1. **Entities** The entities represent the base blocks of the system. Each concept has a set of associated labels that has an associated string. Each of these entities can be further specified with eventual scores, positions, or embeddings.
2. **Storage Management** It is responsible for storing the various entities and additional information. The type of storage is independent from the storage concept, indeed it can be chosen and implemented in various ways. For the sake of this project, only SQL storage have been built.

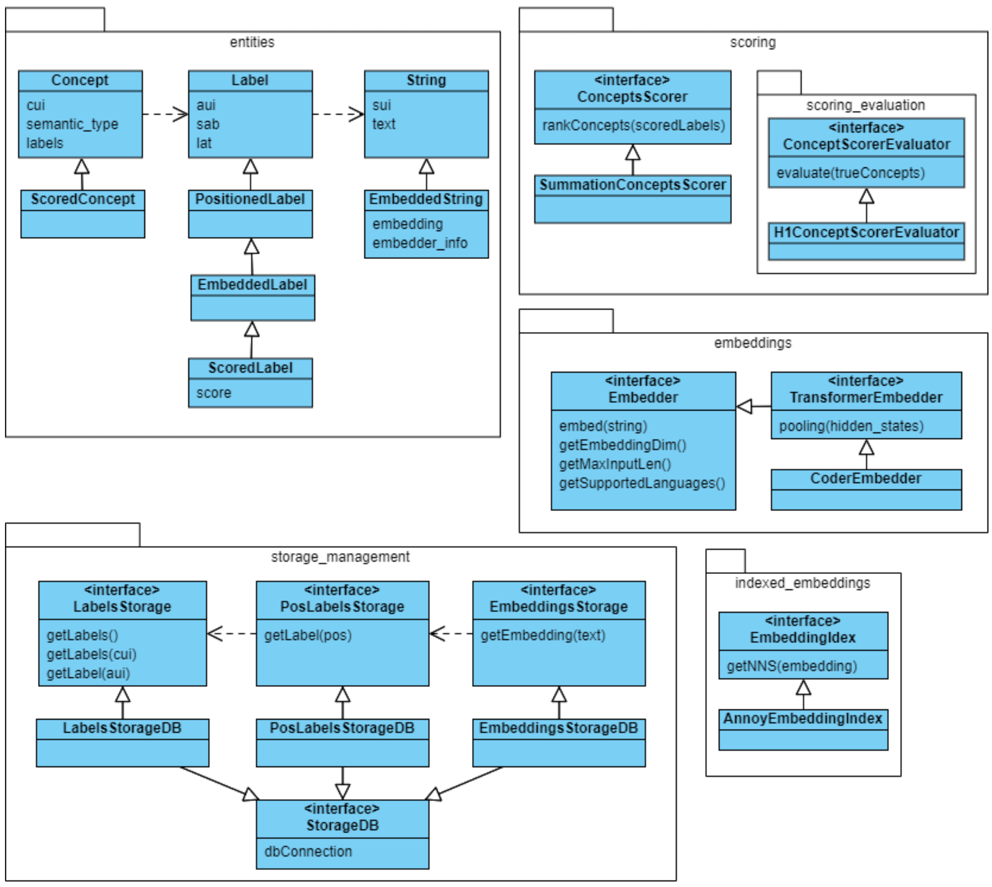


Figure 3.8: Object-oriented class diagram

3. **Indexed Embedding** This module has the responsibility of performing a nearest neighbors search. The only implementation is done through Annoy, but many other are possible like Faiss, Pinecone, etc.
4. **Embeddings** The generation of embeddings is the crucial part of this project. The embedder can be exchanged in a modular way.
5. **Scoring** The scoring system aggregates labels and assigns scores to concepts. Different types of evaluation strategies can be implemented upon this architecture.

3.7 Service Industrialization

The proposed architecture has been industrialized to provide REST APIs that efficiently fulfill the core task of obtaining a ranking of concepts (CUIs) based

on input text. To realize this, the implementation was carried out using the Python library FastAPI[11], which facilitates the development of a robust and user-friendly API system. FastAPI is a high performance micro web framework that simplifies the creation of web applications, suitable for building RESTful APIs.
To further enhance the performance and scalability of the REST APIs, the ASGI (Asynchronous Server Gateway Interface) server Uvicorn[12] was integrated with FastAPI. Uvicorn is a lightning-fast ASGI server that allows for asynchronous request handling. By combining FastAPI with Uvicorn, the architecture ensures efficient and responsive handling of incoming HTTP requests.
These REST APIs, powered by FastAPI and Uvicorn, allow users to seamlessly submit text queries and receive ranked concept responses. This integration not only makes the system accessible and user-friendly but also ensures that it can handle a large number of concurrent requests.

The supporting Neo4J graph and MySQL database are deployed on separate independent services. Thus, the whole architecture could be easily ported on a series of Docker[13] containers.

Services

The developed API endpoints are intended to exploit the whole already built architecture with GET requests and are the following:

1. **Embed** This endpoint is used to retrieve the embedding of a list of strings according to the specified embedder.
2. **Nearest Neighbors** Finds the nearest neighbors of a string, given an embedder and a distance metric [Listing ref TODO]. For each neighbor returns its position in the index, the distance from the target, the derived score, and the label information (CUI, SUI, STR, LAT) [Table 3.3].

[11]`https://fastapi.tiangolo.com/`
[12]`https://www.uvicorn.org/`
[13]`https://www.docker.com/`

3. **Aggregated Nearest Neighbors** Given an aggregation strategy with its eventual parameters, it computes the nearest neighbors and aggregates the relative concepts. It returns the list of concepts with the CUI, the associated score, and the preferred label of that concept [Table 3.4].
4. **Multi** It is possible to use the 'multi' version of the preceding endpoints by providing multiple strings. The result will be the list of the results of each individual string. In this way, the overhead of eventual multiple invocations is reduced and the system can optimize the inference by leveraging batch computation.
5. **Concept Info** It is a utility function that aggregates UMLS data. Given a CUI, it returns all its labels with the associated information. It also aggregates additional label information such as the semantic type, the computed rank, and the definition if present.
6. **Vocabulary Code Concept** It is also a utility function that returns the source code of a concept given its CUI and the filtered source. For instance, this will be useful in the case of ICD9CM usage.
7. **Shortest Path** This endpoint exploits the Neo4J shortest path function to retrieve all the paths between two concepts that have the minimum distance. The distance is not weighted and is computed only with the number of traversed edges. It is possible to specify some filters like the maximum search distance (it impacts the performances), the exclusion of paths that traverse the same nodes, the usage of only 'is a' relations, the usage of relations only in one direction, and the inclusion/exclusion of some specified relationships and/or sources. This function leverages the optimization done on the relations derivation on concepts.
8. **All Paths** This endpoint is similar to the shortest path one but instead returns all the paths between two concepts with a maximum specified distance. The available filters are the same as the previous endpoint.

As shown in tables 3.3 and 3.4, this system is able to provide good quality answers in a fast and reliable way. Indeed, the processing time of the request,

String	SUI	CUI	LAT	Score	Distance
humerus neck	S11899211	C0448034	ENG	0.79	0.64
cuello de húmero	S12447682	C0448034	SPA	0.79	0.64
Collum chirurgicum humeri	S9239249	C0223685	ENG	0.78	0.65
Neck of humerus	S0952987	C0448034	ENG	0.77	0.67
neck of humerus	S11917261	C0448034	ENG	0.77	0.67
cuello de la escápula	S4720608	C0223632	SPA	0.76	0.68

Table 3.3: Nearest neighbors service response on the string ‘collo omerale’ (Italian for ‘humerus neck’), non existing in UMLS. Using $k = 6$, CODER embedder and angular distance/scoring.

CUI	Preferred string	Score
C0448034	Neck of humerus	0.52
C0223685	Structure of surgical neck of humerus	0.13
C0223632	Structure of neck of scapula	0.13

Table 3.4: Aggregated Nearest neighbors service response on the results in table 3.3 with summation aggregation strategy.

without HTTP overheads, is about 250ms. If further insights are needed, it is possible to use the graph endpoints to gain more knowledge about the resulting concepts.

Chapter 4

Datasets

In the context of conducting research for this thesis, a significant challenge was encountered due to the absence of readily available datasets containing Italian mentions explicitly associated with UMLS concepts. This absence posed a substantial hurdle as the primary objective of this study was to explore and analyze the relationships between Italian medical terms and their corresponding UMLS concepts. The unavailability of such datasets meant that alternative data collection and annotation efforts were required to establish a foundation for the research.

4.1 UMLS Subset

UMLS is an extensive resource [Section 2.1.1] but it is too large to be handled during the experiments. Moreover, the contribution of some labels can be intuitively considered low. For this purpose, only a subset of it was used. When referring to the results of the system, this subset has to be considered and not the whole UMLS.

Even if it is recognized that for the sake of the context of this thesis, some sources can be more valuable than others and that not all the semantic types are equally useful, however, the filtering choice is done only with respect to the languages. Indeed, it was chosen to include only the Italian, Spanish, and

English non-suppressed labels of concepts that have at least one label either in Italian, in Spanish, or both. This because they are the top-3 represented languages,

Recalling the UMLS quantitative analysis [Section 2.1.1], there are 9,421,201 English, 1,172,015 Spanish, and 239,011 Italian. The number of concepts is respectively 4,261,033 for English, 463,552 for Spanish, and 164,584 for Italian. There are 164,579 concepts that have at least one English and Italian label, and 462,770 for English and Spanish. Thus, there are 5 missing Italian concepts and 782 Spanish ones, without a corresponding English label. This leads to 481,792 concepts that have an English label and an Italian or Spanish one. Finally, it was decided to drop all the duplicated labels for strings, concepts, and language. All this process led to the dataset illustrated in table 4.1.

Language (LAT)	**Concepts**	**Atoms**	**Atoms per Concept**
ENG	481,792	2,210,114	4.59
SPA	462,770	1,116,424	2.53
ITA	164,579	202,280	1.23
TOTAL	481,792	3,528,818	7.32

Table 4.1: Quantitative analysis for the chosen UMLS subset

The significant increase of the atoms per concept ratio from 3.17 in the full UMLS [Table 2.1] to 7.32 of the selected subset [Table 4.1] indicates that the selected concepts have a higher representation than the average. It could be hypothesized that the most represented concepts are of a higher quality and better connected because of the large number of sources converging to them. However, the drawback might be that some very specific terms are missing.

Considering embeddings in $\mathbb{R}^{768}$ for the pooled output of CODER, and a 4-byte representation for real numbers, the overall memory allocation of the uncompressed embedded labels of this subset is 10.84 GBs.

4.2 MedMentions

MedMentions[1] is a corpus of biomedical texts annotated against UMLS [Mohan and Li, 2019]. It is a valuable resource for NLP tasks in the medical domain, in particular for entity linking. It consists of a corpus of 4,392 papers, including titles and abstracts, which were randomly selected from PubMed[2] in 2016. These papers are exclusively from the biomedical field and are published only in English. Although it does not contain any Italian information, it is still a valuable resource.

To ensure the high quality of annotations, a team of professional annotators with extensive experience in biomedical content curation was engaged. Their task was to comprehensively annotate all mentions in these papers using the UMLS 2017AA terminology. Although no stringent Inter-Annotator Agreement (IAA) data was gathered, a precision assessment was performed, by selecting a small subset of the corpus and two independent biologists as the reviewers. This test led to an agreement with the annotators of a 97.3% precision.

Data extraction The dataset follows the PubTator format [Wei et al., 2013], with each paper document identified by its PubMed identifier (PMID), accompanied by its title, abstract, and mentions. Each annotated mention is a row with the start and end indices (obtained by concatenating the title and the abstract), the actual textual span of the mention, its semantic type (TUI), and the associated UMLS concept (CUI).
A script was implemented to extract the data in a more decontextualized way with respect to the original format. Indeed, it would be easier to use only the mentions without their surrounding context to evaluate individually the CUI guesser service. Thus, the extracted dataset contains the string of the mention, the CUI, the full sentence where the string is contained, the preferred string of

[1]`https://github.com/chanzuckerberg/MedMentions`
[2]`https://pubmed.ncbi.nlm.nih.gov/`

the concept, and the legacy MedMentions information to identify a mention (PMID, start index, end index) [Table 4.2].

However, not all the concepts in MedMentions are included in the UMLS subset [Section 4.1]. Thus, filtering is necessary to draw fair considerations. The filtered MedMentions subset that contains only concepts of the UMLS subset will be referred to as ‘pruned MedMentions’ and it will be the default dataset in the absence of other specifications.

String	**CUI**	**Sentence**	**Preferred String**	**PMID**	**SI**	**EI**
DCTN4	C4308010	DCTN4 as a modifier of chronic Pseudomonas aer...	DCTN4 protein, human	25763772	0	5
chronic Pseudomonas aeruginosa infection	C0854135	DCTN4 as a modifier of chronic Pseudomonas aer...	Pseudomonas aeruginosa infection	25763772	23	63
cystic fibrosis	C0010674	Pseudomonas aeruginosa (Pa) infection in cysti...	Cystic Fibrosis	25763772	124	139

Table 4.2: MedMentions [Mohan and Li, 2019] extracted subset example

4.3 ICD-9-CM Codebook

ICD-9 stands for the International Classification of Diseases[3], 9th Revision, Clinical Modifications. It is a globally recognized system for classifying and coding various diseases, conditions, and medical procedures. ICD-9 was developed and published by the World Health Organization (WHO) and is used for tracking and reporting diseases and health conditions, as well as for billing and statistical purposes in healthcare settings.

ICD-9 codes consist of alphanumeric characters and provide a standardized way to represent medical diagnoses and procedures. These codes are organized into a hierarchical structure, allowing for the classification of diseases into categories and subcategories. The use of ICD-9 codes is essential to ensure consistency and accuracy in medical record-keeping. In Italy, the legislation [Italian Government, 2008] forces the codification of some information in the hospital discharge records against ICD-9-CM.

[3]`https://www.who.int/standards/classifications/classification-of-diseases`

It is worth noting that ICD-9 has been succeeded by ICD-10 (the 10th Revision) in many countries, which offers a more extensive and detailed coding system. ICD-10 provides greater specificity and granularity in describing diseases and medical procedures, reflecting advances in medical knowledge and technology. However, some regions may still use ICD-9 for certain purposes or as a historical reference.

UMLS includes ICD-9-CM among its sources. The associated labels generated by it are only the official English ones. Because the Italian government itself relies on ICD-9-CM an official codebook was scraped by MAPS S.p.A.. The obtained dataset contains Italian diagnosis isolated mentions and the relative ICD-9-CM code. However, they are written in the original codebook in a hierarchical way but the hierarchy is not preserved in the extracted dataset. Moreover, the main issue is that the ICD-9-CM codes proposed are the same across the hierarchy, thus pointing to the most general concept.

4.4 Medical Reports

MAPS S.p.A. possesses a substantial repository of anonymized medical reports within its resources. It is important to note that while these reports remain unannotated, rendering them unsuitable for training or quantitative assessment purposes, they still represent a worth mentioning valuable dataset. Indeed, this dataset, albeit unstructured, can serve a distinct purpose of qualitative analysis and manual inspection. It allows for conducting examinations that may uncover valuable nuances within the medical reports and annotation strategies. While it may not be employed for quantitative evaluations, its qualitative value should not be underestimated, making it a noteworthy component within the available resources.

An unannotated dataset was derived from this corpus by randomly sampling 5,000 medical reports written in Italian. Subsequently, the mentions were extracted using the target guesser, yielding a total of 35,370 targets.

Chapter 5

Experiments and Results

The experiments and results presented in this study are notably impacted by the scarcity of annotated Italian language resources in the field. Addressing this challenge required a substantial endeavor to maximize the utilization of the limited datasets at our disposal. The research was conducted within the constraints of this linguistic resource gap, prompting the adaptation of the existing resources to suit the Italian context. The scarcity of annotated data necessitated a more resourceful and creative approach.

5.1 Dense Representation

The first thing to be dealt with in the proposed system is the dense representation method. Because of the lack of resources to perform intensive trainings, it was decided to rely on a pre-trained language model [Section 2.2]. The available and most prominent models are mBERT[1], MedBIT[2], and CODER[3]. Unfortunately, there are no Italian annotated datasets to test the models with. Moreover, to rely on the quality of the results, the query strings should be lexically different from the ones in the knowledge base but with the same semantic meaning.

[1]mBERT uncased `https://huggingface.co/bert-base-multilingual-uncased`
[2]MedBIT$_{r3}$+ `https://huggingface.co/IVN-RIN/medBIT-r3-plus`
[3]CODER$_{ALL}$ `https://huggingface.co/GanjinZero/coder_all`

A qualitative evaluation was performed by manually checking a little bunch of labels [Table 5.1]. To obtain neutral results, the summation aggregation strategy is chosen with $k = 6$, the distance is the angular one, and only the first concept in the ranking is considered. A score equal to 2 is assigned when the model correctly predicts the concepts or finds an alternative correct representation, a score of 1 if the concept is not the correct one but closely related, 0.5 if the concept is wrong but it has some far relation with the correct one, 0 if completely wrong. The knowledge base is represented by the UMLS subset [Section 4.1]. The evaluations are done by non-physicians on the basis of the descriptions of reliable sources. All the mentions to annotate are not present in Italian or are not present at all in UMLS. A concept is considered good if it is either the right one or can be considered its direct parent.

ID	Mention	CUI	Description
0	Collo omerale	C0448034	Neck of humerus
1	Intossicazione da oppiacei	C0029100	Opioid intoxication
2	Fosfaturia renale	C0282201	Phosphate Diabetes
3	Allergia gastrointestinal	C0221034	Gastrointestinal allergy
4	Angiografia oculare	C1444574	Ophthalmic angiography
5	Sindrome del dolore miofasciale del collo	C0458110	Myofascial pain syndrome of neck
6	Sedimetria	C1176468	Erythrocyte sedimentation rate measurement
7	Flogosi articolare	C0574941	Inflamed joint
8	Fibro fog	-	-
9	Tender point miofasciali in sede atipica	- -	- -

Table 5.1: Tiny dataset of hard Italian mentions

The results presented in Table 5.2 clearly demonstrate the superior performance of the CODER model compared to others. As a result, CODER is selected as the reference model. However, it's important to note that there is still significant room for improvement, as CODER achieved a score of only 10.5 out of 20.

Surprisingly, the MedBIT model performed worse than mBERT. This could

ID	mBERT	MedBIT	CODER
0	C0043425	C0456914	C0448034
	Yolk Sac	Intrastomal	Neck of humerus
	0.33	0.15	0.52
	Wrong: 0	Wrong: 0	Right: 2
1	C0161558	C1290402	C0857503
	Poisoning by barbiturate	Neoplasm of myocardium	Opiate toxicity
	0.17	0.15	0.47
	Wrong: 0	Wrong: 0	Right alternative: 2
2	C0002534	C0032519	C0031678
	Renal aminoacidurias	Polymenorrhea	Phosphoric Monoester Hydroleases
	0.17	0.16	0.34
	Wrong: 0	Wrong: 0	Wrong: 0
3	C0017184	C0162275	C1720579
	Gastrointestinal Motility	Ketonuria	Allergic disorder of digestive system
	0.17	0.16	0.51
	Wrong: 0	Wrong: 0	Right alternative: 2
4	C0007767	C1290401	C1444574
	Cerebral Angiography	Neoplasm of endocardium	Ophthalmic angiography
	0.33	0.15	0.86
	Almost right: 1	Wrong: 0	Right: 2
5	C0027073	C1278535	C0458110
	Myofascial pain syndromes	Post infarct angina	Myofascial pain syndrome of neck
	0.17	0.15	0.61
	Almost right: 1	Wrong: 0	Right: 2
6	C0009661	C0228087	C0036289
	Conductometry	Macroglia	Scattering, radiation
	0.33	0.29	0.19
	Wrong: 0	Wrong: 0	Wrong: 0
7	C0003881	C0230060	C1253936
	Arthrodesis	Structure of inferior wall of orbit	Hydrarthrosis
	0.17	0.16	Hydrarthrosis
	Wrong but related: 0.5	Wrong: 0	Wrong: 0
8	C0015282	C0854374	C0028882
	Exocrine glands	Pigmentation lip	Odontoma
	0.17	0.16	0.50
	Wrong: 0	Wrong: 0	Wrong: 0
9	C1969817	C0600521	C1504509
	Secondary nocturnal enuresis	Radiotherapy, conformal	Localised muscle pain
	0.17	0.15	0.33
	Wrong: 0	Wrong: 0	Wrong but related: 0.5
TOT	2.5	0	10.5

Table 5.2: Results of the different models for data in table 5.1 with their predicted CUI, the preferred string of that concept, the confidence score of the prediction, and the result evaluation.

be attributed to potential issues like catastrophic forgetting, despite the regularization techniques employed by the authors. Nevertheless, it is essential to acknowledge that this empirical evaluation lacks quantitative evidence and relies on a somewhat simplistic metric. Additionally, the absence of an annotated Italian dataset poses challenges, and the complexity of the domain necessitates validation of the considerations by medical professionals.

Furthermore, it's important to mention that the selected mentions [Table 5.1] lack Italian labels that are syntactically similar. In cases where such labels are absent, it is possible that the target concept is not included in the chosen subset, nor are its English labels. Lastly, during the inference process, a contextual analysis should be conducted by considering the entire textual context containing the mention.

5.2 MedMentions

MedMentions [Mohan and Li, 2019] represents a valuable resource though only in English [Section 4.2]. It cannot be considered a reliable candidate for assessing quantitative results against Italian mentions annotations. The assumption that considerations done for English mentions hold also for the Italian ones, is over-simplistic. However, it can be used to gain insights about the system like hyperparameters tuning and hypothesis validation.

Hyperparameter k In order to find a good k hyperparameter for the number of neighbors extracted from the index [Section 3.3], some tests were performed by evaluating the top-n accuracy, namely, hits at n (H@n), and the mean reciprocal rank (MRR) [Equation 2.2], for all $n \leq k$. In particular, the test was performed by taking into consideration the summation aggregation strategy in order to understand also how much noise is obtained as k increases.

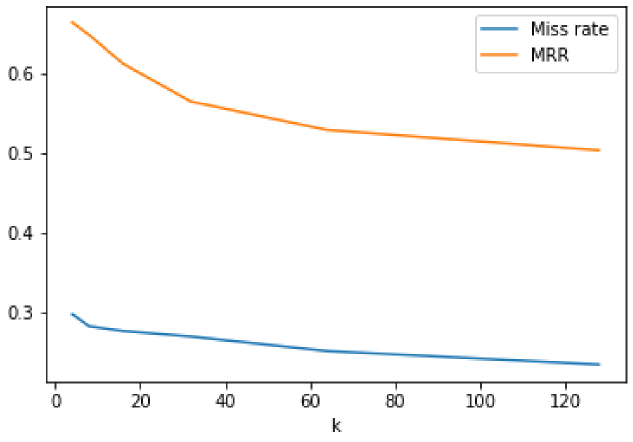


Figure 5.1: Mean Reciprocal Rank and miss rate on MedMentions

As illustrated in figure 5.1, as the value of k increases, both the miss rate (indicating the absence of correct concepts among the top-k predictions) and the MRR exhibit a decrease, which aligns with our expectations. Notably, the miss rate experiences a gradual decrease, implying the effectiveness of the employed strategies. Nevertheless, there is room for improvement in the aggregation strategy to elevate lower-ranked concepts to higher positions.

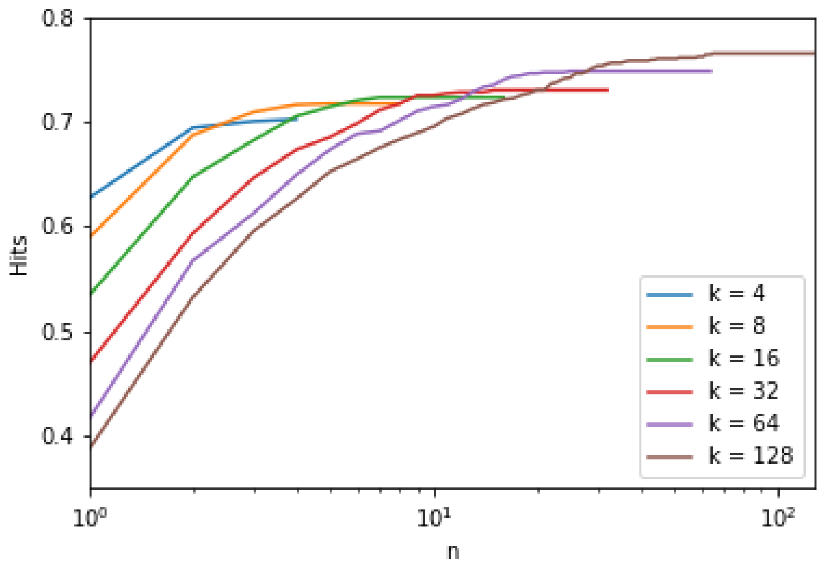


Figure 5.2: Hits at k trend

Figure 5.2 summarizes the results obtained by changing k from 2^2 to 2^7. The summation strategy brings lower-ranked wrong concepts to high positions. That's why a weighted summation depending on the ranking should be employed. It is clear the dominance of the top-ranking concepts. Increasing k too much worsens efficiency and effectiveness. It is believed that an optimal value is around 8. After that, the noise gathered is more heavy than the possibility of coding a right lower-ranked concept.

Worth to mention is the astonishing increase between H@1 and H@2. This suggests that often the system is in doubt between two similar concepts but only one is the right one. Moreover, it could be the case that the top-ranked concept is correct but a duplicate of the one in the reference dataset.

Overall, even if the performances on MedMentions of H@1 reached 62.70%, and H@32 of 72.30%, these experiments still provided good insights into the behavior of the system and how to choose the hyperparameters.

5.3 Medical Reports

The dataset of the targets extracted from the medical reports of MAPS S.p.A. was used to manually check the annotations provided by the system. For each mention $m_i \in M$ its number of occurrences o_i among the extracted targets, was counted. The top-counting 216 mentions were selected. All these mentions constitute 14,670 occurrences in the documents, over the total 35,370: the 41% of all the targets.

Then they are automatically annotated by the built system with the summation aggregation strategy and the top-ranking concept is selected. Then, the annotations are manually checked for correctness, assigning them a boolean score. In this way, the ambiguity of repeated concepts in UMLS is avoided. The results aggregated per semantic group are reported in table 5.3, with an overall 94.96% corpus accuracy. The corpus accuracy is obtained by considering the occurrences of each mention, while the standard accuracy is computed as if the multiplicity is one. Filtering only for the most relevant semantic groups (anatomy, disorders, chemicals & drugs, and procedures) leads to an even higher 95.55% corpus accuracy.

Semantic Group	**Count**	**Occurrences Count**	**Correct Count**	**Wrong Count**	**Accuracy**	**Accuracy Corpus**
Disorders	86	5789	75	11	87.21	92.85
Anatomy	80	4693	79	1	98.75	99.00
Procedures	27	1662	26	1	96.30	96.69
Concepts & Ideas	10	1531	10	0	100.00	100.00
Living Beings	4	688	2	2	50.00	72.53
Chemicals & Drugs	7	233	6	1	85.71	84.98
Occupations	1	46	1	0	100.00	100.00
Phenomena	1	28	1	0	100.00	100.00
TOTAL	216	14,670	200	16	92.59	94.96

Table 5.3: Results of medical report annotation

Hard Mentions

As shown in table 5.4, the system is able to correctly annotate mentions whose Italian labels are not present in the UMLS knowledge base. This is a very important result, allowing the codification of things that are never seen. This demonstrates the ability of the model to learn good embeddings and of the overall surrounding ranking system to extract valuable results.

Mention	**Annotated CUI**	**Annotated Preferred string**	**Annotated Semantic Group**
Lesioni pleuro parenchimali	C1536731	Disorder of pleura AND/OR pleural cavity	Disorders
Impregnazioni contrastografiche	C0009924	Media, Contrast	Chemicals & Drugs
Alterazioni densitometriche	C0544704	Density, abnormal	Disorders
Microcalcificazioni patologiche	C0521174	Microcalcification, calcified structure	Disorders
Alterazioni bronchitiche	C0006261	Bronchial Disease	Disorders

Table 5.4: Correctly annotated hard mentions

Error Analysis

It is important to analyze the errors of the system. In table 5.5 are shown 5 of the 16 wrongly annotated mentions. They belong to different semantic groups and their annotations are clearly wrong.
The mention 'lobo'[4] is associated with a disorder while it should be an anatomy. The context is nonexistent but if the system is prompt with a specification of it like 'lobo auricolare'[5], then it returns the correct concept C0229315 with a 0.55 confidence score. Thus, the context surely helps the system to find more accurate results.
Other mentions are coded wrongly but are not so distant from the real target. That's why a binary metric fails to capture these nuances. For instance, 'parenchima polmonare'[6] even if its codification is 'pulmonary alveoli', this is not so distant from the true concept. Indeed, they could be considered siblings in

[4]Italian for 'lobe'
[5]Italian for 'ear lobe'
[6]Italian for 'lung parenchyma'

a way, they belong to the same organ and are physically very close.

Mention	**Occur-rences**	**Annotated CUI**	**Annotated Preferred string**	**Annotated Semantic Group**
Lobo	68	C0152066	Blastomycosis, keloidal	Disorders
Controllo ortopedico	55	C0024725	Manipulation, Orthopedic	Procedures
Spazi liquorali periencefalici	49	C0023529	Encephalomalacias, Periventricular	Disorders
Parenchima polmonare	47	C0034051	Pulmonary Alveoli	Anatomy
Macrocalcoli	35	C0032483	Glycols, Polyethylene	Chemicals & Drugs

Table 5.5: Errors of medical report annotation

5.4 Graph Distances

One claim of this thesis is that relations in the knowledge graph generated by UMLS can help to disambiguate terms. In particular, distances between the mentions of the same sentence are considered. Moreover, an endeavor was made to prune and group the relationships. Indeed, some relations that can easily be derived, like the ‘sibling’ one, are cut away in order to mitigate the noise and have distances that are more representative of reality. Moreover, some of them were grouped in a way similar to the semantic groups, with the same purposes, for instance, the PAR/CHD and RB/RN are IS-A/HAS-A relationships. The new relationships will be referred to as ‘simplified’. Several layered tests were made to observe the connection patterns by using the filters provided by the service [Section 3.7]. All the shortest path experiments were tested with a maximum search distance equal to 5.

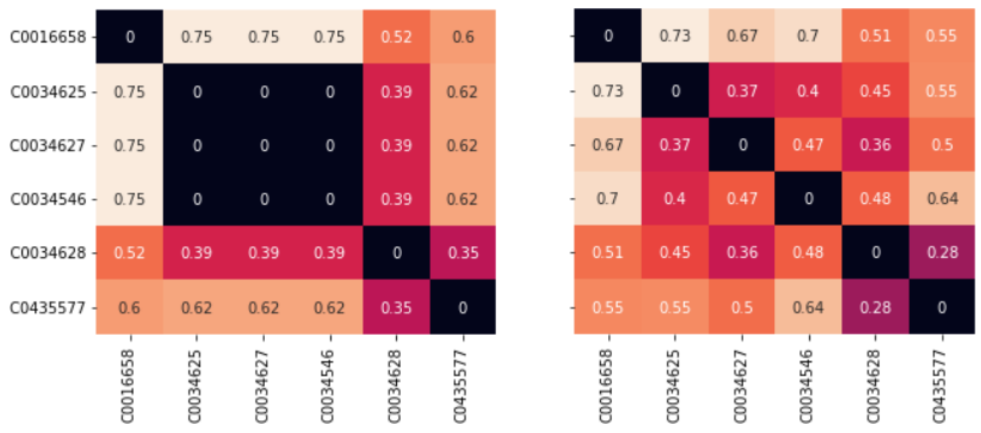
The sentence under examination is ‘frattura del capitello del radio’[7]. The annotated targets by exact label matching are reported in table 5.6. As it is possible to see, the word ‘radio’ is heavily ambiguous because can have three different meanings, incompatible with each other. The objective is to exclude the non-related concepts.

[7]Italian for ‘fracture of the radial head’

Mention	Concepts	Preferred labels
Frattura	C0016658	Fracture
radio	C0034625	Radium
radio	C0034627	Bone structure of radius
radio	C0034546	Radio communications
Frattura del radio	C0034628	Radius fractures
Frattura del capitello del radio	C0435577	Fracture of radial head

Table 5.6: Exact label matching example

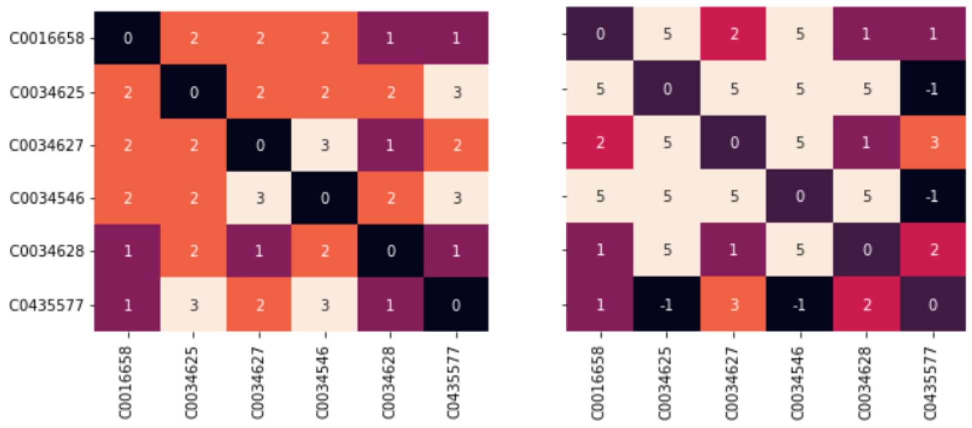
Embedding distances The embedding angular distances [Figure 5.3] computed on the target text spans, show that disambiguation with lexically similar terms is difficult. One strategy is to perform the measurements on the preferred labels of the directly matched mentions, however, it is risky because if it does not introduce additional context, the problem still remains. This case was lucky because of the descriptions of the preferred labels.



(a) Strings (b) Preferred labels

Figure 5.3: Mention embedding distances

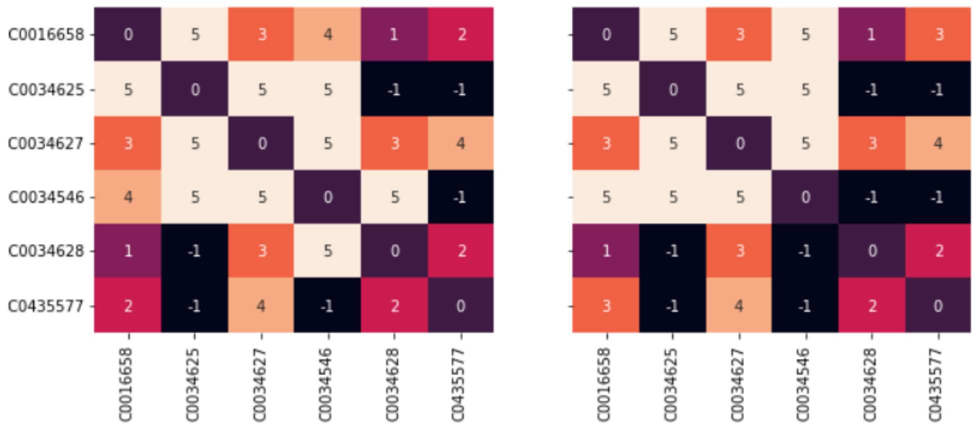
All undirected relationships Another strategy could be to measure the distances between the concepts [Figure 5.4], hoping that non-related concepts are put far away from the others. This technique relies on the number of target mentions within a sentence. However, it results that also not-related concepts are highly connected. That’s because of the collagen effect of some UMLS relationships and too general and too close parent nodes.



(a) Standard relationships (b) Simplified relationships

Figure 5.4: Shortest path distance, all undirected relationships

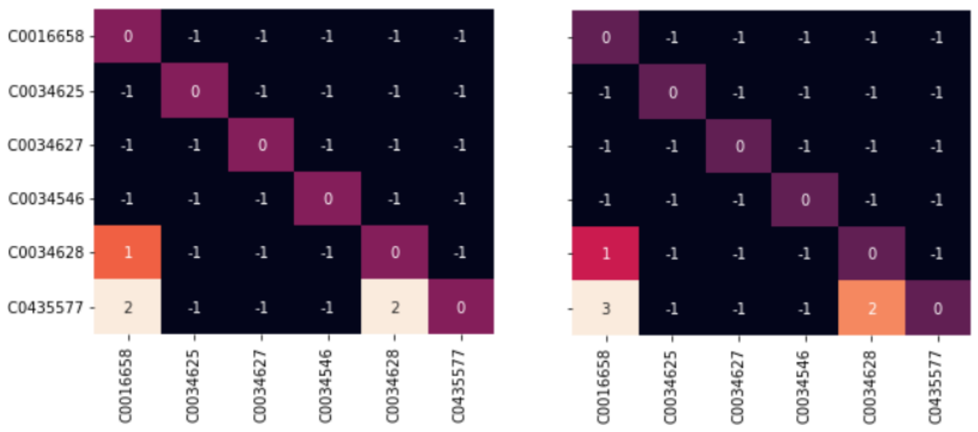
Is-a undirected relationships Filtering the relationships by keeping only the IS-A ones [Figure 5.5], is an effective strategy. This filtering approach yields results that are notably straightforward and relevant in terms of related concepts. However, it's worth noting that this type of query can be computationally intensive and may require substantial time and resources to execute.



(a) Standard relationships (b) Simplified relationships

Figure 5.5: Shortest path distance, is-a undirected relationships

Is-a directed relationships The hardest strategy is to rely only on directed IS-A paths [Figure 5.6]. This results in a too-strong filtering. However, it can still be used for hard confirmations. It is important to notice that this approach fails to capture closely related siblings. Nevertheless, its potentiality stays in the fact that it can be used to determine the generality level of the concept to emit, and eventually prune general concepts in favor of more specific ones, or the opposite in the case of safer requirements.



(a) Standard relationships (b) Simplified relationships

Figure 5.6: Shortest path distance, is-a directed relationships

In summary, the assessment of distances between candidate concepts associated with a given mention proves to be a valuable tool for disambiguating biomedical terms. This approach can seamlessly integrate with the concept guessing pipeline, enhancing the precision of its outcomes. Furthermore, it holds the potential to significantly support the concept ranking strategy, offering a more robust and context-aware mechanism for identifying the most relevant concepts. As demonstrated, by incorporating such distance measurements and considerations, it is expected an improved precision entity linking task, and an enhanced reliability of the scores.

Chapter 6

Conclusions

This thesis has successfully demonstrated an effective ranking methodology that integrates and implements an NLP pipeline for the annotation of biomedical documents. The carefully designed architecture addresses the industrial needs of MAPS S.p.A., making it a valuable asset for the company.
Despite the scarcity of Italian resources, leveraging all available assets, including Italian non-annotated corpora and English datasets from the literature, has yielded valuable results by harnessing the UMLS relational knowledge.
The primary objective of this project was to enhance the capabilities of MAPS S.p.A.'s Clinika software in the entity linking task for biomedical targets against the UMLS. Through the utilization of state-of-the-art language models and ranking techniques, an impressive accuracy rate of 94.96% was achieved on domain-specific data. Notably, the system demonstrated the ability to code labels that are not present in the UMLS knowledge base and disambiguate lexically similar mentions.
The entity extraction pipeline plays a crucial role as an input provider for the concept guesser architecture, supporting the overall effectiveness.
In summary, it can be confidently stated that the designed component for Clinika is reliable and effective, representing a substantial improvement in the multilingual entity linking task, particularly against predominantly English knowledge bases.

6.1 Future Works

Progress in NLP opens up opportunities to enhance different facets of this project's architecture. Furthermore, it's worthwhile to investigate and assess alternative solutions.

Ranking Strategy The ranking strategy could be enhanced by incorporating the graph's relational knowledge and utilizing machine learning techniques to identify the most significant candidates. Additionally, analyzing all the labels linked to a candidate concept could provide further improvements. Furthermore, it would be valuable to automatically reason upon the semantic types and dependencies provided by the entity extractor.

Graph Embeddings The Neo4J Graph Data Science (GDS) library provides a straightforward way to generate graph embeddings given a similarity, an encoder, a decoder, and a loss function. However, for ranking purposes, it's crucial that these embeddings are inductive. TNeo4J offers native algorithms such as Fast Random Projection, node2Vec, and GraphSAGE. The first algorithm is based on linear algebra, while the latter two leverage neural networks. It would be valuable to assess the performance of graph-generated embeddings, including the possibility of properly selecting the relationships.

ICD-9 Linking Providing an ICD-9 code for a given target is a valuable functionality. Nonetheless, the challenge lies in the generality of the codes linked to UMLS concepts. During this project, an analysis was made about the distances between the coded specific correct concept and the concept with the correct ICD-9 code associated. A systematic approach should be designed.

CODER++ Training As demonstrated, CODER constitutes a valuable resource but CODER++ was not tested because of its training that was made only on English corpora. It is plausible that training CODER++ on Italian and other languages could yield noteworthy improvements in embedding performance. Regrettably, this endeavor was not pursued in this thesis due to constraints in computational resources and time.

Large Language Models Recent advancements in the field of Large Language Models have significantly advanced AI capabilities. Major companies, like Google, are actively developing medical-specific language models like Med-PaLM [Singhal et al., 2023]. Investigating and leveraging such solutions represent the most promising frontier in the current landscape.

Indexing Technologies The current indexing architecture relies on Annoy and MySQL. Keeping the positions aligned is a full responsibility of the developers. Also, MySQL is not optimized for this kind of positioned label retrieval task. New solutions appeared in the computer science/engineering landscape, like Pgvector and PostgreSQL. While PostgreSQL is a well-established object-relational DBMS, Pgvector was developed only recently and it is fully integrated with PostgreSQL, dealing with the position alignment task automatically in an optimized way.

BIOS - Biomedical Informatics Ontology System BIOS[1] is a recently published Chinese-based ontology system similar to UMLS. Exploring and integrating new solutions might be interesting for the improvements of the whole Clinika pipeline.

Italian Dataset To evaluate progress more accurately, it is crucial to create a specialized Italian dataset tailored to Clinika's objectives. Within each document, every relevant piece of text should be associated with the corresponding concept. To facilitate the dataset's construction, the existing pipeline can be utilized for initial text annotations. Subsequently, manual review, correction, and supplementation of automatically identified matches can be performed.

[1]`https://bios.idea.edu.cn/lang-en`

Bibliography

[Aronson, 2001] Aronson, A. R. (2001). Effective mapping of biomedical text to the umls metathesaurus: the metamap program. In *Proceedings of the AMIA Symposium*, page 17. American Medical Informatics Association.

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

[Attardi et al., 2015] Attardi, G., Cozza, V., and Sartiano, D. (2015). Annotation and extraction of relations from italian medical records. In *IIR*.

[Attardi et al., 2009] Attardi, G., Dei Rossi, S., Dell'Orletta, F., Vecchi, E. M., et al. (2009). The tanl named entity recognizer at evalita 2009. In *Proceedings of the Workshop Evalita 2009*.

[Barbieri, 2023] Barbieri, M. (2023). Design and development of an nlp pipeline for the annotation of clinical texts in italian. Master's thesis, University of Parma, Computer Engineering.

[Bloom, 1970] Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426.

[Blundell, 2023] Blundell, J. (2023). Health information and the importance of clinical coding. *Anaesthesia & Intensive Care Medicine*, 24(2):96–98.

[Bodenreider, 2004] Bodenreider, O. (2004). The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270.

[Buonocore et al., 2023] Buonocore, T. M., Crema, C., Redolfi, A., Bellazzi, R., and Parimbelli, E. (2023). Localizing in-domain adaptation of transformer-based biomedical language models. *Journal of Biomedical Informatics*, 144:104431.

[Choi et al., 2016] Choi, Y., Chiu, C. Y.-I., and Sontag, D. (2016). Learning low-dimensional representations of medical concepts. *AMIA Summits on Translational Science Proceedings*, 2016:41.

[Damerau, 1964] Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Emiliani, 2017] Emiliani, V. (2017). System for automated verification of medical documents. IT Patent IT201700050680A1.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Italian Government, 2008] Italian Government (2008). Aggiornamento dei sistemi di classificazione adottati per la codifica delle informazioni cliniche contenute nella scheda di dimissione ospedaliera e per la remunerazione delle prestazioni ospedaliere. Decree 18 December 2008.

[Leaman et al., 2013] Leaman, R., Islamaj Doğan, R., and Lu, Z. (2013). Dnorm: disease name normalization with pairwise learning to rank. *Bioinformatics*, 29(22):2909–2917.

[Lee et al., 2020] Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., and Kang, J. (2020). Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

[Liu et al., 2020] Liu, F., Shareghi, E., Meng, Z., Basaldella, M., and Collier, N. (2020). Self-alignment pretraining for biomedical entity representations. *arXiv preprint arXiv:2010.11784*.

[Ma et al., 2019] Ma, X., Wang, Z., Ng, P., Nallapati, R., and Xiang, B. (2019). Universal text representation from bert: An empirical study. *arXiv preprint arXiv:1910.07973*.

[McCallum et al., 2000] McCallum, A., Freitag, D., Pereira, F. C., et al. (2000). Maximum entropy markov models for information extraction and segmentation. In *Icml*, volume 17, pages 591–598.

[Miftahutdinov and Tutubalina, 2019] Miftahutdinov, Z. and Tutubalina, E. (2019). Deep neural models for medical concept normalization in user-generated texts. *arXiv preprint arXiv:1907.07972*.

[Mohan and Li, 2019] Mohan, S. and Li, D. (2019). Medmentions: A large biomedical corpus annotated with umls concepts. *arXiv preprint arXiv:1902.09476*.

[Salazar et al., 2020] Salazar, J., Liang, D., Nguyen, T. Q., and Kirchhoff, K. (2020). Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712, Online. Association for Computational Linguistics.

[Singhal et al., 2023] Singhal, K., Tu, T., Gottweis, J., Sayres, R., Wulczyn, E., Hou, L., Clark, K., Pfohl, S., Cole-Lewis, H., Neal, D., et al. (2023). Towards expert-level medical question answering with large language models. *arXiv preprint arXiv:2305.09617*.

[Sung et al., 2020] Sung, M., Jeon, H., Lee, J., and Kang, J. (2020). Biomedical entity representations with synonym marginalization. *arXiv preprint arXiv:2005.00239*.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[Wang et al., 2019] Wang, X., Han, X., Huang, W., Dong, D., and Scott, M. R. (2019). Multi-similarity loss with general pair weighting for deep metric learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5022–5030.

[Wasylewicz and Scheepers-Hoeks, 2019] Wasylewicz, A. and Scheepers-Hoeks, A. (2019). Clinical decision support systems. *Fundamentals of clinical data science*, pages 153–169.

[Wei et al., 2013] Wei, C.-H., Kao, H.-Y., and Lu, Z. (2013). PubTator: a web-based text mining tool for assisting biocuration. *Nucleic Acids Research*, 41(W1):W518–W522.

[Wu et al., 2016] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

[Yuan et al., 2022] Yuan, Z., Zhao, Z., Sun, H., Li, J., Wang, F., and Yu, S. (2022). Coder: Knowledge-infused cross-lingual medical term embedding for term normalization. *Journal of Biomedical Informatics*, page 103983.

[Zeng et al., 2022] Zeng, S., Yuan, Z., and Yu, S. (2022). Automatic biomedical term clustering by learning fine-grained term representations. In *Proceedings of the 21st Workshop on Biomedical Language Processing*, pages 91–96, Dublin, Ireland. Association for Computational Linguistics.