Alma Mater Studiorum · Università di Bologna

**Computer Science and Engineering - DISI**
**Master Degree in Artificial Intelligence**

# Knowledge graph embedding enhancement using ontological knowledge in the biomedical domain

**Supervisor:**
Prof. Paolo Torroni
**Co-Supervisors:**
Prof. Andrea Galassi
Vieri Emiliani

**Candidate:**
Lorenzo Niccolai

**III Session**
**Academic Year 2022-23**

# Abstract

The biomedical field is a critical area for natural language processing (NLP) applications because it involves a vast amount of unstructured data, including clinical notes, medical publications, and electronic health records. NLP techniques can help extract valuable information from these documents, such as disease symptoms, medication usage, and treatment outcomes, which can improve patient care and clinical decision-making. MAPS S.p.A. currently produces Clinika, a software that extracts knowledge from clinical corpora. Clinika performs the task of Named Entity Recognition (NER) by linking entities to medical concepts from an established knowledge base, in this case, the Unified Medical Language System (UMLS).

This dissertation details how we approached the design and implementation of a component for the new version of Clinika, specifically a model that outputs mentions embeddings to perform entity linking with UMLS concepts. We focused on enhancing existing dense contextual embeddings by injecting ontological knowledge, using two parallel approaches: (1) taking the embeddings as a by-product of an entity alignment model aided by an ontology, and (2) fine-tuning a contextual language model with contrastive learning.

We evaluated both approaches with suitable experiments from the relevant literature. After testing, we discontinued the first approach but found more significant results using the second. The results on the tasks chosen to evaluate the embeddings were not promising, we address the causes in the Error Analysis section, and finally discuss further work on this topic.

# Contents

**Abstract** **iv**

**1 Introduction** **1**
1.1 Objectives . . . 1
1.2 Background . . . 3
1.2.1 Knowledge Graph Embedding . . . 3
1.2.2 Language Models . . . 3
1.2.3 UMLS . . . 4
1.3 Thesis Structure . . . 5

**2 Work Methodology** **7**
2.1 Entity Alignment Approach . . . 8
2.1.1 Entity Alignment Problem Statement . . . 9
2.1.2 Related Work . . . 9
2.1.3 Techniques . . . 12
2.1.4 Our contribution . . . 18
2.2 Transformer based approach . . . 19
2.2.1 Term Representation . . . 19
2.2.2 Related Work . . . 20
2.2.3 Techniques . . . 21
2.2.4 Our contribution . . . 23

**3 Experiments and Results** **33**
3.1 OntoEA . . . 33
3.1.1 Downstream Task . . . 33
3.1.2 Data . . . 33
3.1.3 Metrics . . . 34
3.1.4 Results . . . 34
3.2 CODER . . . 35
3.2.1 Semantic Group Classification . . . 36
3.2.2 MedMentions . . . 38

3.3 Error Analysis . . . . . 39
3.4 Discussion . . . . . 43

**4 Conclusions** **45**

**Bibliography** **46**

# List of Figures


2.1 Example of an entity mapping using both graph and ontology information 13
2.2 Basic example of how TransE embeds entities which have a relation connecting them as a translation in the embedding space. . . . . 14
2.3 The big picture of OntoEA framework . . . . . 16
2.4 CODER contrastive learning process applied to UMLS concepts. . . . . 23
2.5 The Siamese network setting for a Sentence BERT model. . . . . 24
2.6 A fragment of SNOMEDCT_US ontology, one of the source vocabularies present in UMLS Knowledge Graph. The yellow and red elements are part of hierarchies which are disjointed. . . . . 25
2.7 An example of hierarchy route cutting on the same atoms of figure 2.6. The red dashed line between C3 and C4 represents where the route is cut for the Hierarchical samples creation process. . . . . 27
2.8 Scheme of the fine-tuning loop with sampling and online negative mining. The purple box launches the algorithm 2 on each of the atoms present in the random sample to form 16k additional hard negatives. . . . . 31

3.1 An example of how we mapped the vocabularies entities to the Semantic Types to create an artificial ontology. The dashed lines will be the membership links in our setting, the solid lines are extracted from UMLS Metathesaurus. . . . . 34
3.2 A two-dimensional visualization of three dense embeddings, with the CODER embeddings marked in blue and the HierCODER embeddings in red. In this case, the 'Structure of the third trochanter of the femur' is the focal point, while 'Bone and/or joint structure' is part of the positive hierarchical sample and 'Structure of the third cuneiform facet of the cuboid bone' is the paired negative atom. . . . . 40
3.3 A two-dimensional visualization of three dense embeddings, with the CODER embeddings marked in blue and the HierCODER embeddings in red. In this case, the 'Structure of medial epicondyle of femur' is the focal point, while 'Limb structure' is part of the positive hierarchical sample and 'Structure of midshaft of femur' is the paired negative atom. . . . . 41

3.4 A two-dimensional visualization of three dense embeddings, with the CODER embeddings marked in blue and the HierCODER embeddings in red. In this case, the 'Methyldopate hydrochloride' is the focal point, while 'L-Tyrosine, 3-hydroxy-alpha-methyl-, Ethyl Ester, Hydrochloride' is part of the positive hierarchical sample and 'Manidipine hydrochloride' is the paired negative atom. . . . . 42

# List of Tables


2.1 Models results on FR_EN_V2 benchmark. . . . . 19

2.2 A representative fragment of the collection resulting from concept 'Blood Clot' (CUI `C0302148`), using the algorithm 1. The first column is added to highlight the specific type of sample. . . . . 28

2.3 The results, the function $f(e, n)$ for the embedding of the atom *Blood Clot* and $n = 12$. . . . . 30

3.1 This table shows the results of the models that participated in Task 1 of OAEI 2021. Fine-TOM, which ranked 10th in the challenge, is included to provide additional context for the performance of the OntoEA model. 35

3.2 The number of samples seen by HierCODER, FinerCODER and FocusCODER during the fine-tuning. The percentages are in proportion to the total (51,814,562) of positive samples generated by Algorithm 1. . . . . 36

3.3 The average accuracy, precision, recall and F1-score for each of the fine-tuned CODER models on the seen and unseen terms semantic group classification. . . . . 38

3.4 The results of CODER, HierCODER, FinerCODER and FocusCODER on the MedMentions entity linking task, with a zero-shot setting. . . . . 39

# Chapter 1

# Introduction

## 1.1 Objectives

This thesis outlines the work undertaken during a curricular internship at MAPS S.p.A. with the aim of enhancing Clinika, a patented software product for analyzing medical documents (patent for industrial invention no. 102017000050680 – "Systems for automated verification of medical documents"), which 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. Specifically, Clinika performs Named Entity Recognition (NER) on medical documents such as medical reports, medical prescriptions, and discharge letters. Notably, all processed documents are in the Italian language, presenting a significant challenge due to the limited labeled data resources available in this language. Moreover, the objective is not only to link entities to a limited number of classes, but rather to map them to an already existing terminology system that consists of millions of potential categories. One of the most prominent and established systems in the biomedical field is UMLS.

The Unified Medical Language System (UMLS)[35] is a biomedical ontology developed by the National Library of Medicine. It is divided into three parts:

**Metathesaurus** — integrates approximately two hundred data sources, including vocabularies, thesauri, and ontologies, into a unified system to code and standardize the terms and lexical variants used in natural language for referring to concepts. In addition to relationships and hierarchies between concepts, the Metathesaurus serves as a vast knowledge base. It can be represented as a graph with two types of nodes: concepts and atoms. Each concept is identified by a unique ID called CUI and encompasses multiple atoms from different sources. Each atom is identified by a unique ID called AUI, which corresponds to a textual label and is associated with a broader concept. Relations exist between concepts and between atoms, with the only possible relationship between these two categories of nodes being the

membership of an atom in a concept.

**Semantic Network** — consists of a set of 133 semantic types and a set of relations between the various types. Most of the concepts present in the Metathesaurus are associated with one or more semantic types.

**SPECIALIST Lexicon and Lexical Tools** — an English lexicon comprising biomedical and non-biomedical terms, accompanied by syntactic and grammatical information. The lexicon is accompanied by Lexical Tools, a set of tools that implement essential NLP functionalities in Java.

The project of developing the new version of Clinika, namely Clinika 2.0, was carried out by me and two other interns over the course of the past year. The project comprises three key components:

- The data extraction pipeline, which is a combination of SpaCy components that aim to identify important tokens in a given input text. The most crucial component is the Dependency Parser, which determines the best syntactic structure for the input. This is a challenging task since many medical documents do not adhere to logic and grammar conventions. The goal of this component is to identify the part of text which can be linked to the reference knowledge base (i.e., UMLS). This pipeline outputs a list of *mentions* - which can include one or more tokens – and optionally some metadata.

- The embedding model, which generates dense embeddings for a given sequence of tokens. This was the primary focus of my work: aiming to obtain the most effective dense representations for biomedical terms.

- A ranker, that combines embeddings of *mentions* from the model and metadata from the pipeline to rank possible UMLS concepts and decide which one to associate with a given *mention*.

So the main questions for my research were:

1. What are the best methods to embed a *mention* and ensure its representation is consistent with the reference knowledge graph, specifically UMLS in our case?

2. Which method is most suitable for our case?

3. Can we improve the chosen method using ontological knowledge?

In this context, ontological knowledge refers to the relationships within the UMLS that depict the hierarchical organization of concepts and their constituent atoms. The challenge is quite complex as it involves embedding a text span and obtaining a graph representation. The first task has become commonplace in recent years with the emergence

of large language models and the idea of pre-training and fine-tuning them for specific needs. The second task arises from the need to map a *mention* to a set of concepts that are not independent, but rather have multiple layers of relations between them. In the next section, we will introduce the ideas behind the main tools we used to address this problem.

## 1.2 Background

The following paragraphs will introduce all the necessary background elements required for a complete understanding of the thesis work.

### 1.2.1 Knowledge Graph Embedding

Knowledge graph embedding is a technique used to represent entities and their relationships in a knowledge graph as dense vectors in a high-dimensional space. This allows for more efficient computation and enables the use of machine learning algorithms to make predictions and discover patterns in the data. In the biomedical context, knowledge graph embedding can be used to represent medical concepts and their relationships in a compact and meaningful way. This can aid in the process of mapping mentions of medical concepts in text to their corresponding concepts in a reference terminology system, such as the UMLS. By utilizing knowledge graph embeddings, it becomes possible to better capture the semantic relationships between medical concepts, which is not possible using purely contextual embeddings that do not take into account the knowledge coming from an ontology.

### 1.2.2 Language Models

Language models are computer programs for natural language processing that use transformers and neural networks, that are designed to process natural language text, learn the patterns and relationships between words, and use that knowledge to perform various tasks such as language generation, classification, and translation. The BERT (Bidirectional Encoder Representations from Transformers)[9] family of models are a type of language model developed by Google that have revolutionized the field of natural language processing. Unlike previous language models that processed text in a unidirectional manner (either left-to-right or right-to-left), BERT is a bidirectional model that can take into account the entire context of a sentence when making predictions. This allows it to perform tasks such as question answering and sentiment analysis with unprecedented accuracy. BERT models are pre-trained on massive amounts of text data, such as Wikipedia articles and online books, and then fine-tuned on specific tasks to

make predictions on new data. BERT models have been used in a wide variety of applications, from chatbots and language translation to medical text mining and clinical decision support systems.

### 1.2.3 UMLS

A medical terminology system is a standardized vocabulary that allows for the uniform and precise representation of medical concepts and their relationships. These systems typically include a set of medical terms and codes that are used to identify and describe diseases, procedures, medications, and other medical concepts. They provide a common language for healthcare professionals, researchers, and other stakeholders to communicate and share information about patient care and outcomes. Medical terminology systems can also include hierarchical structures that reflect the relationships between different concepts, allowing for more efficient information retrieval and analysis. Some commonly used medical terminology systems include the International Classification of Diseases (ICD), the Current Procedural Terminology (CPT), and the Unified Medical Language System (UMLS).

The Unified Medical Language System (UMLS) is a comprehensive and integrated medical terminology system developed by the National Library of Medicine (NLM). It is designed to facilitate the management and sharing of biomedical information and knowledge among different applications and systems. The UMLS comprises a vast collection of concepts and their relationships from more than 100 different controlled vocabularies, classification systems, and other biomedical resources. These resources include MeSH[1], SNOMED CT[2], RxNorm[3], and LOINC[4], among others. The UMLS is built on a metathesaurus that integrates these different sources of biomedical knowledge and provides a unified and standardized representation of medical concepts and their relationships. The UMLS serves as a critical resource for various applications in the biomedical domain, including information retrieval, natural language processing, clinical decision support, and data analysis.

The UMLS Metathesaurus is composed of over 3 million biomedical concepts, which are interconnected by more than 11 million relationships that represent semantic connections between them. These relationships are of various types, such as synonymy, hierarchical, associative, and temporal. In addition to concepts and relationships, the Metathesaurus also includes various other types of information, such as supplementary concept information, attribute value pairs, and source vocabulary information. The supplementary concept information includes additional semantic information about the

[1] https://www.nlm.nih.gov/mesh/meshhome.html
[2] https://www.snomed.org/
[3] https://www.nlm.nih.gov/research/umls/rxnorm/index.html
[4] https://loinc.org/

concept, such as its definition, usage notes, and examples of how it is used in clinical practice. Attribute value pairs represent additional attributes of a concept, such as its spelling variants, abbreviations, and alternate representations. The source vocabulary information provides metadata about the source vocabularies, such as their name, version, and license agreement.

To facilitate data analysis and computational processing of UMLS data, various statistical summaries and descriptive analyses have also been provided. These include frequency distributions of concepts and relationships, cross-referencing information, and other metadata about the Metathesaurus content. The size and complexity of the UMLS Metathesaurus, along with its extensive coverage of biomedical knowledge, make it a valuable resource for many medical informatics applications, including medical term normalization, natural language processing, and data integration.

## 1.3 Thesis Structure

The thesis will follow this structure:

- In the next chapter we will define the work methodology. How we decided to tackle our problem, what methods we utilized and what we added as our original contribution.
- Then we will present the experiments defined to assess the outcome of our work. How we will measure it, on which data and the results, along with the error an analysis.
- Finally, we will draw the final conclusions.

# Chapter 2

# Work Methodology

We can define the UMLS knowledge graph as follows.

Let $\mathcal{D} = \{n_i\}_{i=0}^{\|\mathcal{D}\|}$ be the UMLS concept set, where $n_i$ is a concept identified by CUI $cui_i$. Each concept $n_i$ has several labels $\{s_i^j\}_{j=1}^{c_i}$ as equivalent terms which can express the concept, each one identified by AUI $aui_i^j$. We define the embedding associated to label $s_i^j$ as $\mathbf{e}_i^j \in \mathbb{R}^l$. Finally, the term normalization goal that we defined in 1.1 can be expressed as:

$$\hat{n} = n_{argmax_i(cos(\mathbf{e}_i^j, \mathbf{e}))} \tag{2.1}$$

where $s$ is an input string with embedding $\mathbf{e}$, $n$ is the related UMLS concept to predict and $cos$ is the cosine similarity measure in the Euclidean space.

In order to obtain a useful embedding $\mathbf{e}_i^j \in \mathbb{R}^l$ from textual relational data (i.e., a string of text which is part of a graph, $s_i^j$) employing the Ontological knowledge we explored two different approaches in parallel:

1. Performing Entity Alignment between two parts of the UMLS Knowledge Graph using an architecture which takes advantage by the presence of an ontology which lies above the two. This way of tackling the problem assumes that data maintains its relational structure when it is fed to the model, specifically the representation will be based mainly on its relations extracted from the graph. This approach gives the embeddings as a by-product of the alignment.

2. Employing a Transformer based model which does not consider directly the data relational structure. In this setting the graph relations are used indirectly, specifically the model is trained with a contrastive learning method, applied to atom pairs mined from UMLS. After that, we can do inference on any arbitrary string and obtain an embedding.

For each of the aforementioned approaches we will now present the techniques adopted, the architecture employed and what we added or produced as our original contribution.

PREFIX ex: <http://dbpedia.org/resource/>
REL_PREFIX rex: <http://dbpedia.org/ontology/>

RDFS:
<ex:Hip_house> <rex:derivative> <ex:Eurodance> .

Listing 2.1: A RDF triple representing the statement: "Hip house is a derivative of Eurodance"

## 2.1 Entity Alignment Approach

The first approach is based on Entity Alignment, which is the task of finding entities in two knowledge graphs that refer to the same real-world object.
A knowledge graph is a structured representation of objective world knowledge, including entities, relationships, attributes, and semantic descriptions. The knowledge is typically stored using RDF[18], with each piece of knowledge represented as a relational triplet or an attribute type triplet. A triple is a set of three entities that codifies a statement about semantic data in the form of subject–predicate–object expressions. To illustrate, consider the following statement: 'Hip house - is a derivative of - Eurodance,' where the subject, predicate, and object are represented by the three segments separated by dashes. It can be represented in RDF as depicted in listing 2.1. In this case, 'http://dbpedia.org/resource/' is set as a prefix for the subject and the object, while 'http://dbpedia.org/ontology/' is used as a prefix for the predicate. From this basic structure, triples can be composed into more complex models, by using triples as objects or subjects of other triples.

Knowledge graphs are an important source of information for many AI applications, including information extraction and fact checking. However, as the requirements of applications become increasingly diversified, a single knowledge graph is often insufficient.

Entity alignment is the process of merging heterogeneous knowledge from different data sources and languages into a unified and consistent knowledge graph. This is particularly relevant for integrating data from multiple sources that can be partially overlapping but complimentary (as in the case of two or more medical terminology dictionaries), or that are in different languages (cross-lingual alignment). Also, knowledge graphs are often incomplete, meaning that not all entities are represented, or that their attributes and relationships are only partially represented. Entity alignment can help to fill in these gaps and create more complete, and usable knowledge graphs.

In addition, entity alignment can also help to facilitate interoperability between different systems and applications, improving the completeness and accuracy of knowledge

graphs. By providing a common representation of entities across multiple sources, it becomes easier to exchange and share data between different systems.

In the next sections, we will provide a formal description of the Entity Alignment problem, report the relevant literature and, how we decided to approach it in the context of our goal, which is to leverage ontological knowledge to improve the expressiveness of multilingual, biomedical embeddings.

### 2.1.1 Entity Alignment Problem Statement

The entity alignment problem between two Knowledge Graphs using a shared ontology can be defined as follows.

A KG is denoted as $G = (E, R, T)$, where $E, R, T$ are the set of entities, relations and triples, respectively. A triple $(h, r, t) \in T$ is made of a head entity $h \in E$, a relation $r \in R$ and a tail entity $t \in E$. Their embeddings are denoted as $\boldsymbol{h,r,t}$ respectively. Assuming we want to create an alignment between two Knowledge Graphs, we will denote them as $G_i$ and $G_j$.

An ontology is denoted as $O = (C, H)$, where $C$ is the classes set and $H$ is a set of triples which contains only subsumption relations. Furthermore, the membership relation sets, which link the entities and the corresponding classes, are denoted as $B_i$ and $B_j$. Taking $B_i$ for example, it links $O$ and $G_i$, so it is composed of pairs $b_i = (e_i, c)$ where $e_i \in E_i$ and $c \in C$. Their embeddings are denoted as $\boldsymbol{c,\ b\_i}$ respectively.

An entity mapping, denoted as $m = (e_i, e_j)$ where $e_i \in E_i, e_j \in E_j$, indicates that $e_i$ and $e_j$ refer to the same concept. The entity alignment (EA) problem aims to find all the mappings $M$ between $E_i$ and $E_j$, where we assume a small set of known entity mappings (or seed mappings) $M_s$ is given.

### 2.1.2 Related Work

The literature on Entity Alignment consists of both traditional and neural methods. The traditional methods, according to [55], include various approaches, such as comparing symbolic features like entity names, attributes, and attribute values using similarity calculations. Some researchers have proposed expanding the alignment entity iteratively through similarity propagation. One notable example of a state of the art, non-neural model in this category is Agreement Maker Light (AML)[11], which primarily uses lexical matching algorithms but also includes structural algorithms for matching and filtering, as well as a logical repair algorithm. AML also employs external sources of background knowledge, and it has been recently applied for the task of building a network of 28 integrated ontologies in the biomedical domain[38].

Recently, there has been a rapid development in knowledge graph representation learning methods, and as a result, new entity alignment methods have emerged. These

new models utilize knowledge graph representation learning or graph-based methods to represent entities as low-dimensional vectors based on their semantic or structural information in the knowledge graph. Finally, they compare the similarity between these vectors to identify equivalent entities, which reduces the impact of differences in knowledge graphs on entity alignment. By encoding the diverse knowledge of various knowledge graphs as a vector representation of entities, these methods simplify the inference process and can automatically identify equivalent entity pairs from the knowledge graphs on a larger scale.

Since the seminal work of TransE[6], many methods of knowledge graph representation learning have emerged. Following the taxonomy proposed by [55], we can group them in four main classes: Translational distance models (such as TransE and RotatE[43]) that are based on vector translation; Semantic matching models that use complex transformations to calculate the plausibility of fact triples (e.g., RESCAL[34], DistMult[53], and TuckER[2], which we also took into consideration as one of the potential candidates for our project); Neural network-based models, where Convolutional Neural Networks (see ConvE[8], ConvKB[33]) and Graph Neural Network (GNN) or a combination of these techniques (as per R-GCN[37] and KBGAT[32]) are used to model the structural information of knowledge graphs. Finally, the last class extends the aforementioned models making use of additional knowledge on top of the fact triples, such as the description of the entities (DKRL[52]), or even pictures.

In the last years, there has been significant interest in entity alignment methods that leverage knowledge graph representation learning methods. These alignment models represent entities as low-dimensional vectors using graph-based or knowledge graph representation learning methods, in combination with knowledge graph structure information or external resources. By calculating the similarity between these vectors, equivalent entity pairs can be automatically extracted from heterogeneous knowledge graphs at a large scale, without requiring many artificial features. Due to these benefits, this approach has gained popularity in both academia and industry. These methods can be broadly categorized into two types: semantic matching-based models and graph neural network-based models.

#### Semantic matching-based models

Semantic matching entity alignment is about learning low-dimensional vector representations for each entity in the knowledge graph. One approach involves TransE-based models, such as MTransE[7], that use linear transformation to align entities from different knowledge graphs. Another approach is to enrich entity semantics by integrating various types of knowledge, as exemplified by JAPE[40]. Other models focus on the structural information of the knowledge graph, such as TransEdge[45], which use novel

models to align entities by better spreading relationships between them or representing complex relationships in multiple dimensions. The development trend is towards integrating more dimensional knowledge graph information and multi-modal external resource information. The latest state-of-the-art model is BERT-INT[47], but it requires powerful description information that is difficult to obtain in many real entity alignment scenarios.

### GNN-based models

Graph Neural Networks (GNNs) have emerged as a prominent area of study for deep learning researchers in recent years, demonstrating remarkable efficacy in addressing a broad range of graph-based problems. The rapid advancement of GNN-related technologies has fostered the growth of entity alignment models based on GNNs. In such models, a graph neural network processes graph-structured data, with each node's representation influenced by those of its neighbors, facilitating the capture of both local and global structure information. By leveraging the natural graph structure of the knowledge graph, the entity alignment model based on GNNs has yielded favorable outcomes, learning low-dimensional vector representations for distinct entities.

For example, AliNet[44] is a graph neural network (GNN) based model for entity alignment. It leverages the structural information of knowledge graphs to learn entity representations that capture both local and global graph structure. AliNet uses a novel attention mechanism to weigh the importance of different parts of the graph for entity alignment. It also employs a cross-graph attention mechanism to align entities from different graphs by considering their common and distinct neighbors. GCNAlign[50] is another GNN-based model for entity alignment. Like AliNet, GCNAlign also operates on a bipartite graph representation of two heterogeneous networks, and it aims to learn a low-dimensional embedding for each entity in each network such that entities that are similar across the two networks are close to each other in the embedding space. However, GCNAlign uses graph convolutional networks (GCNs) to learn these embeddings, while AliNet uses an attention mechanism.

### OntoEA

Across our literature review, an interesting ontology-guided entity alignment method emerged, named OntoEA, that outperforms many of the models mentioned above in several benchmarks. OntoEA is a model for entity alignment that incorporates an external ontology to guide the alignment process. The model uses joint knowledge graph embedding to learn low-dimensional representations of entities from two different knowledge graphs, and then incorporates the ontology to guide the alignment process by enforcing constraints on the entity embeddings. The model achieves state-of-the-art results on

several entity alignment benchmarks and can effectively handle the heterogeneity and sparsity of knowledge graphs. Also, the authors released the code of their implementation of the framework, although implemented in Tensorflow 1.x.

[24] has compared traditional and neural entity alignment methods on a multitude of benchmarks and concluded that PARIS [13], a state-of-the-art non-neural method, statistically outperforms all representative state-of-the-art neural methods in terms of both efficacy and efficiency across various datasets. Since our approach requires producing entity embeddings, we had to use a neural method. Moreover, OntoEA yielded promising results on MED-BBK-9K[56], a biomedical entity alignment benchmark where this model outperforms all the models mentioned before. These elements, among all the others, led us to select OntoEA for our experiments.

### 2.1.3 Techniques

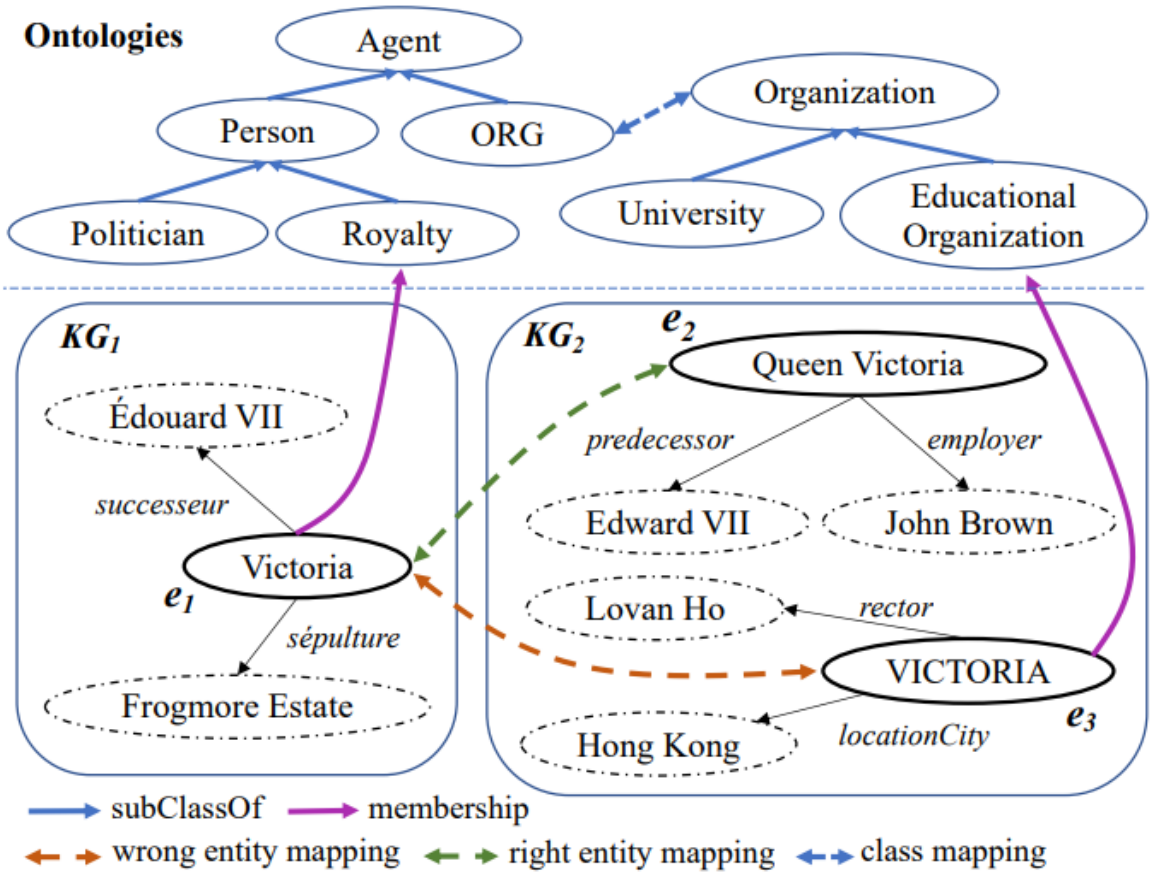
OntoEA[51], introduces a model that performs ontology-guided entity alignment between two knowledge graphs. We utilized this model for our application for three reasons. Firstly, the model produces embeddings that take into account the ontology, which can be further examined to determine the effectiveness of incorporating ontological knowledge. Secondly, aligning entities between different UMLS sources is a challenging task due to the intrinsic inconsistency of the UMLS knowledge graph[16], and achieving good alignment results can act as an indicator of the effectiveness of using the ontology. Lastly, some source vocabularies, particularly those in low resource languages, are not fully represented in the UMLS Metathesaurus. OntoEA's ability to produce a good alignment can be leveraged to automatically extend the UMLS Metathesaurus for new sources.

It is important to note that OntoEA focuses on the Entity Alignment problem (defined in 2.1.1), modifying existing embeddings according to the knowledge extracted from the relations of the Knowledge Graph and the ontology. The initial embedding of entities and relations - i.e. obtaining $\boldsymbol{c}$ from $c$ - can be chosen arbitrarily. For example, the default OntoEA setting initializes the embeddings randomly, but it is also possible to load pre-trained embeddings coming from any kind of source: FastTText[5], Word2Vec[30], BERT[9], etc. Of course this holds while assuming that the entities to align has at least one textual label as attribute - which is our case - and if they have more it is possible to apply any sort of pooling.

The main idea behind OntoEA is illustrated in figure 2.1, which displays a case of class conflict in a mapping, where the use of the ontological knowledge prevents a wrong match due to polysemy. Since we want also to use the knowledge coming from an ontology lying above the two Knowledge Graphs, but the Knowledge Graphs do not always share one ontology, here we state how to handle the two cases:

- If the Knowledge Graphs share an ontology already, that ontology will be used.

Figure 2.1: Example of an entity mapping using both graph and ontology information




- If the Knowledge Graphs have two different ontologies, they will be merged into a single ontology which will act as the one shared by the Knowledge Graphs. This can be done using existing ontology alignment systems such as PARIS[39] and LogMap[15] and/or cost sensitive human intervention.

The main design choice concern how to inject the knowledge coming from relations into the embeddings. OntoEA utilizes different methods when it comes to embed the Knowledge Graph components and the ontology components.

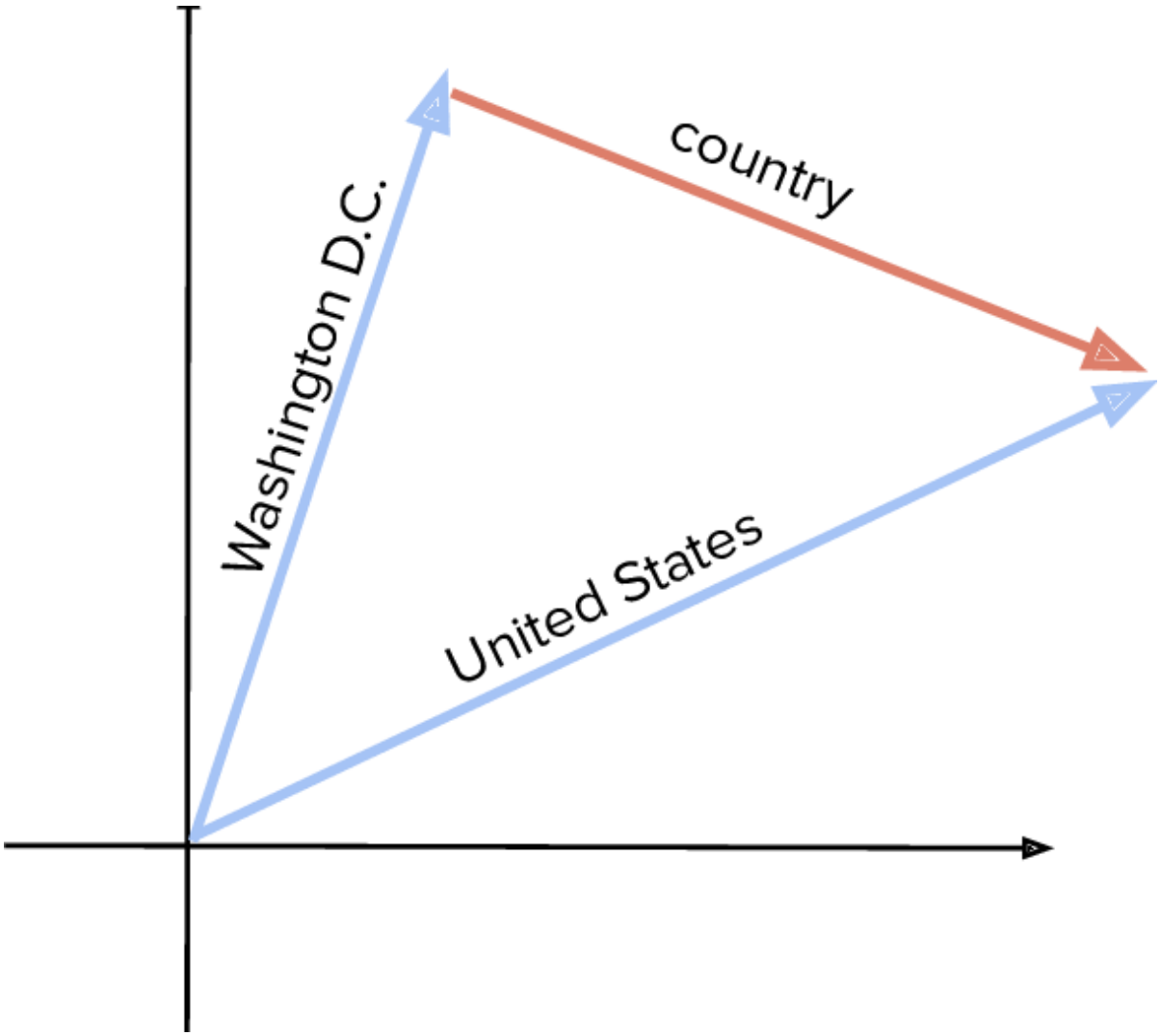
### Knowledge Graph Embedding

The technique employed to embed the Knowledge Graphs $G_j, G_i$ is based on TransE[6], which is a translation-based method proposed by Bordes et al. in 2013. With this method, relationships are represented as translations in the embedding space. For example, if $(h, r, t)$ holds, then the embedding of the tail entity $t$ should be close to the embedding of the head entity $h$ plus some vector that depends on the relationship $r$. Figure 2.2 provides an illustration of this process using the example of the triple $(h, r, t)$, where $h$ is 'Washington D.C.', $r$ is 'country', and $t$ is 'United States'. So ideally the

following should hold:

$$\boldsymbol{t} = \boldsymbol{h} + \boldsymbol{r} \tag{2.2}$$

Figure 2.2: Basic example of how TransE embeds entities which have a relation connecting them as a translation in the embedding space.




To ensure that the embeddings satisfy this condition, OntoEA incorporates a dedicated module that transforms it into a mixed margin-based loss and limit-based loss, based on the ideas proposed in [57]. We will discuss this further in section 2.1.3.

### Ontology Embedding

Although TransE can be used to embed the ontology $O$, the authors of OntoEA recognize that the ontology subsumption relation is transitive. This means that we can infer (A, subClassOf, C) via (A, subClassOf, B) and (B, subClassOf, C), leading to one-to-many and many-to-one mappings (or triples) in the ontology. TransE cannot adequately

address this transitive property[27]. Therefore, the chosen method for embedding the ontology comes from [12]. Instead of using a translation in the embedding space, it employs a non-linear transformation. If the triple $(c_h, r, c_t)$ exists in $H$, ideally, the following should hold:

$$c_t = \tanh\left(W_o c_h + b_o\right) \tag{2.3}$$

Where $W_o \in \mathcal{R}^{d_o \times d_o}$ and $b_o \in \mathcal{R}^{d_o}$ are learnable parameters and $d_o$ denotes the ontology embedding dimension. This tends to encode each class as a sphere and each subclass as a vector in the same semantic space after the non-linear transformation. The relative positions are then used to model the relations between a class and its subclass.

### Sampling Strategy

This model employs two different sampling strategies, which are as follows:

- Uniform Negative sampling is used in [6] and it is a basic method to sample negative triples. For a given a triple $(h, r, t) \in T$, a negative triple $(u, r, t)$ (or $(h, r, u)$) is created by replacing either $h \in E$ or $t \in E$ - not both at the same time - with a random entity $u \in E$.

- $\epsilon$-truncated sampling, which is proposed by [42], is designed to mine negative triples which are hard to distinguish from positive ones. Instead of sampling the replacement entity from all entities as in Uniform Negative sampling, the sampling scope is limited to a group of candidates. Specifically, the replacement entity for a given entity $x \in E$ is chosen from its $s$-nearest neighbors in the embedding space, where $s = \lceil (1 - \epsilon)|E| \rceil$ and $\epsilon \in [0, 1)$. The search for $x$'s neighbors uses the cosine similarity between embeddings.

### Conflict Matrix

OntoEA creates a Class Conflict Matrix (CCM) to represent inter-class conflicts and extract knowledge from the ontology to provide a better Entity Alignment. Within the CCM, the entry on the $i^{th}$ row and $j^{th}$ column - denoted as $m_{i,j}$ - represents the class conflict degree between class $c_i$ and class $c_j$. Given an ontology $O$ and two classes $c_i, c_j$, $m_{i,j} \in [0, 1]$ can be divided into the following cases:

- $m_{i,j} = 0$, this means that there is no conflict between the classes and at least one of the following conditions hold:
  - $c_i \equiv c_j$
  - $\exists\ b_i, b_j \in B_k$ such that $b_i = (e_k, c_i), b_j = (e_k, c_j)$, where $e_k \in E_k$

- $m_{i,j} = 1$, this means that the class conflict degree is maximum, which means that $c_i$ and $c_j$ are disjointed. This can be explicitly stated by the ontology or implicitly discovered from the entities.

- $m_{i,j} \in \ ]0, 1[$, this means that none of the previous conditions has been met, so the score is calculated according to:

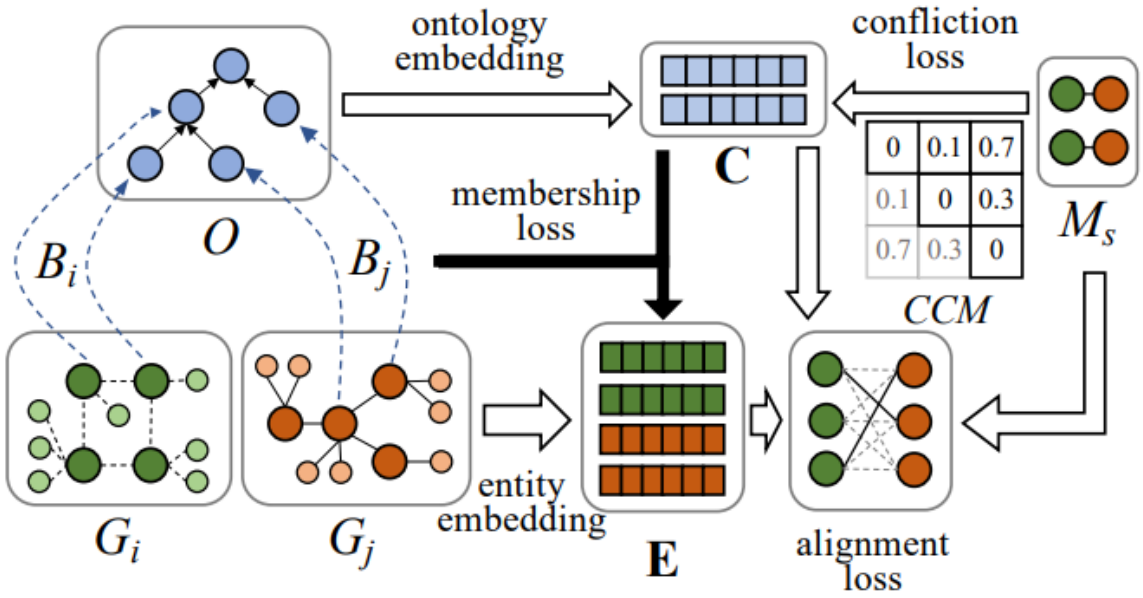
$$m_{i,j} = 1 - \frac{|S(c_i) \cup S(c_j)|}{|S(c_i) \cap S(c_j)|} \tag{2.4}$$

  Where, $S(c_i), S(c_j)$ are respectively the set of classes passed by routing from $c_i, c_j$ to the ontology root node.

### Architecture

OntoEA is a multi-objective architecture, it is composed of five independent modules, each of which has its own loss function and optimizer. Figure 2.3 can give a general idea of how the OntoEA framework puts together the modules.

Figure 2.3: The big picture of OntoEA framework




The five modules consist of:

**Entity Embedding** It aims to embed the Knowledge Graphs with the methods defined in 2.1.3. The loss function adopts the $\epsilon$-truncated sampling strategy mentioned in 2.1.3.

**Ontology Embedding** This module aims to embed the Ontology with the methods defined in section 2.1.3. The loss function adopts the uniform negative sampling strategy mentioned in 2.1.3.

**Confliction Loss** This module minimizes the negative log-likelihood loss defined in Equation (2.5) to incorporate the class conflicts represented by CCM - defined in section 2.1.3 - into the class embeddings.

$$\mathcal{L}_{\mathcal{C}} = -\sum_{c_i \in C} \sum_{c_j \in C} m_{i,j} \log d_{cos}(c_i, c_j) \tag{2.5}$$

where $m_{i,j}$ is the CCM entry defined in 2.1.3 and $d_{cos}(c_i, c_j) = 1 - \cos(\mathbf{c}_i, \mathbf{c}_j)$.

**Membership Loss** This module associates the Knowledge Graph embedding spaces with the Ontology embedding space, it uses the same non-linear transformation as the Ontology Embedding module. However, it applies it to the $b_i, b_j \in B_i, B_j$ pairs using a uniform negative sampling strategy.

**Alignment Loss** This final module performs the actual mapping. Starting from the seed mappings $m = (e_i, e_j) \in M_s$, and its loss function is the following:

$$\mathcal{L} = \sum_{e_i, e_j \in M_s} f_a(e_i, e_j) \tag{2.6}$$

with $f_a(e_i, e_j)$ defined as:

$$\|W_a e_i - e_j\|_2 \tag{2.7}$$

where $W_a \in \mathcal{R}^{d_e \times d_e}$ is a learnable square matrix and $d_e$ denotes the Knowledge Graph embedding dimension.

An iterative co-training strategy is proposed to incorporate these five modules, and each module loss is computed independently and sequentially. This training process affects the embeddings of entities, class, and relations. After a validation condition is met, the training process stops, and the computed entity embeddings and class embeddings can be used to create an alignment. Given two entities $e_i \in E_i$, $e_j \in E_j$ and their respective classes $c_i, c_j \in C$, the weighted similarity score is calculated as follows:

$$sim(e_i, e_j) = \beta \cos(e_i, e_j) + (1 - \beta) \cos(c_i, c_j) \tag{2.8}$$

Where the hyperparameter $\beta \in [0, 1]$ balances the similarities of entity embeddings and class embeddings.

During prediction, we rank all candidate entities in $G_j$ by their weighted similarity scores for each entity in $G_i$ to be aligned.

### 2.1.4 Our contribution

To enhance the comprehensibility of the OntoEA architecture and improve code readability, we decided to migrate the code from Tensorflow 1.x (TF1.x) to PyTorch. To verify the effectiveness of our migration, we conducted experiments on the EN-FR-15K-V2 dataset[41], which was used as a Entity Alignment benchmark in the original paper. The dataset is a cross-lingual version of DBpedia, aligned through *owl:sameAs* links among different language sources. To build the knowledge graphs (KGs) of DBpedia, the benchmark employed the DBpedia ontology and membership relationships from the DBpedia SPARQL endpoint by querying the classes of each entity with *rdfs:type*. Additionally, the CCM was initialized with the class disjointness constraints specified by *owl:disjointWith*.

Let $M$ be the set of the $t$ ground truth alignments, denoted by $m_k = (e_i, e_j)_{k=0}^{t}$, where $e_i \in E_i, e_j \in E_j$. For an entity $e_i \in E_i$ to be aligned, the alignment module generates an ordered ranking $R_i = \{e_u\}_{u=0}^{m}$, where $m = |E_j|$ and $e_u \in E_j$. The metrics used for the Entity Alignment task and reported in the paper are:

**Hits@n** where $n$ is an arbitrary parameter, the default Hits metrics for this model are $n \in \{1, 5\}$. Given a model that produce an alignment, we can say the that the Hits@n response to input entity $e_i$ is positive if the following conditions hold:

$$e_j \in \{e_f\}_{f=0}^{n} \in R_i \tag{2.9}$$

$$(e_i, e_j) \in M \tag{2.10}$$

In simpler terms, if a model scores 50% in the Hits@5 metric, it means that half of the aligned entities ground truth was included among the top 5 candidate entities ranked by the model.

**Mean Rank** which is defined as:

$$MR = \frac{\sum_{i=0}^{m} x_i}{m} \tag{2.11}$$

where $x_i$ is the position in $R_i$ of $e_j \in E_j | (e_i, e_j) \in M$.

**Mean Reciprocal Rank** which is defined as:

$$MRR = \frac{1}{m} \frac{1}{\sum_{i=0}^{m} x_i} \tag{2.12}$$

It is the reciprocal of the Mean Rank defined in 2.11.

In table 2.1 we report the results of the two implementations on FR_EN_V2.

Table 2.1: Models results on FR_EN_V2 benchmark.

| | **Hits@1** | **Hits@5** | **Mean Rank** | **Mean Reciprocal Rank** |
|---|---|---|---|---|
| **Paper** | .654 | .891 | 3.022 | 0.757 |
| **TF1** | .631 | .874 | 3.124 | 0.738 |
| **PyTorch** | .607 | .861 | 3.458 | 0.718 |

While the PyTorch implementation performed slightly worse compared to both the results presented in the original paper and the TF1 implementation that we ran on our system, it is important to note that minor implementation and environmental differences may affect the metrics. Therefore, we can conclude that the PyTorch implementation is comparable to the TF1 implementation. Furthermore, we have successfully produced a PyTorch version of the OntoEA architecture, which is more readable and can be made available to the research community.

## 2.2 Transformer based approach

The second approach utilizes Transformers, specifically Language Models, to tackle the task of medical term normalization. Language Models based on Transformers are capable of generating a contextual representation of any given text span, which can then be utilized to perform the normalization of medical terms. This task involves linking non-standard names, abbreviations, and misspellings to their corresponding standard terms or concept IDs in an existing terminology system, such as the Unified Medical Language System (UMLS). It is a critical challenge, particularly when dealing with unstructured biomedical data, as is the case in our study. In the following sections, we will elaborate on the problem and explain our approach to addressing it.

### 2.2.1 Term Representation

We can define the term embedding as follows.

A transformer language model takes an input term $s$ and outputs an embedding vector $\mathbf{e}$. The input term $s$ is tokenized to sub-words [CLS],$s_0, s_1, ..., s_k$,[SEP], with the first and the last being special tokens. The language model (in this specific case BERT) then produces a series of hidden states:

$$\mathbf{h}_{[CLS]}, \mathbf{h}_0, ..., \mathbf{h}_k, \mathbf{h}_{[SEP]} = LM(\mathbf{s}_{[CLS]}, \mathbf{s}_0, ..., \mathbf{s}_k, \mathbf{s}_{[SEP]}) \tag{2.13}$$

The embedding of $s$ can be:

- The representation of the [CLS] token:

$$\mathbf{e} = \mathbf{h}_{[CLS]} \tag{2.14}$$

- Any kind of pooling (mean, sum, etc.) applied to all the hidden states:

$$\mathbf{e} = pool\left(\mathbf{h}_{[CLS]}, \mathbf{h}_0, ..., \mathbf{h}_k, \mathbf{h}_{[SEP]}\right) \tag{2.15}$$

### 2.2.2 Related Work

The literature on transformer-based approaches for medical term representation counts a number of methods to encode the terms: Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), or Pre-trained Language Models (PLM). [48] proposes a hybrid approach that combines an RNN for encoding the semantic and contextual information of a medical term with a conditional random field for classifying the term into its corresponding medical concept code for medical term normalization. The same authors then publish [29], which utilizes various neural approaches (LSTM, BERT) that encode the semantic and contextual information of a medical term into a multi-class classification problem and use the predicted class to normalize the term.

Another possibility for this task is ranking methods, to determine the similarity between the input term and potential target terms by training them as pairs of positive and negative examples. [22] learns to rank target terms by calculating similarities between TF-IDF vectors. [25] encodes input and target terms by CNN and uses a pairwise approach for ranking. [10] trains a Siamese LSTM network and uses hard negative samplings to find informative pairs. BIOSYN [46] uses both TF-IDF and BioBERT [23] to represent terms. Synonym marginalization is used to maximize similarities between synonyms.

When it comes to the PLM term embedding, many others employ BERT-derived models specifically trained on UMLS as an embedding method. BioBERT [23] is a pre-trained biomedical language representation model based on the transformer architecture. BioBERT has shown success in medical term normalization by utilizing a combination of domain-specific pre-training and fine-tuning on a small labeled dataset of medical terms. SciBERT [4] is a similar model that has been specifically trained on scientific text. SciBERT's pre-training objectives are designed to capture domain-specific knowledge, such as sentence-level coherence and scientific term usage, which has shown to improve its performance on medical term normalization tasks. SapBERT [26] uses a self-alignment pre-training process on term synonyms, with BERT as a starting point.

Overall, these models demonstrate the effectiveness of various approaches to medical term normalization, including the use of transformer-based models, hybrid models, and models based on CNNs and LSTMs. For our goal, we found the best fit in contrastive learning on knowledge graphs for cross-lingual medical term representation, or CODER [54]. CODER is a transformer model proposed in 2021, with all the code publicly available on GitHub[1]. The motive behind the creation of CODER is to improve

[1] `https://github.com/GanjinZero/CODER`

Term Normalization on biomedical data by the exploitation of contrastive learning. However, in practice, the CODER model generates a meaningful dense embedding for any given string $s$.

### 2.2.3 Techniques

CODER (**co**ntrastive learning on knowle**d**ge graphs for cross-lingual m**e**dical term **r**epresentation) is a Transformer Language Model, which starts from a BERT pre-trained on biomedical data and exploits contrastive learning techniques, based on UMLS to perform medical term normalization. We chose CODER because:

- The contrastive training strategy is similar to the fine-tuning that we wanted to experiment, which will be further discussed in section 2.2.4
- CODER provides cross-lingual dense medical term representation for many languages, for example Spanish, Italian, Russian, German, etc. This was needed in our case since the objective is mainly related to Italian health records.

When it comes to contrastive learning, the main design choices entail how to mine the positive samples, but even more so the negative samples. Now we will introduce the strategy adopted by CODER.

#### Contrastive Learning for UMLS

CODER training strategy is based on the concept of contrastive learning, so it aims to separate positive and negatives examples. Let $b$ be a batch of training data of size $n$, which consists of $n$ triples $(h, r, t)$, where $h \in \mathcal{D}$ is the head concept, $t \in \mathcal{D}$ is the tail concept and $r \in \mathcal{R}$ is a relation linking them in the UMLS Knowledge Graph. Then for each concept $c_i \in \{h_j\}_{j=0}^{n} \cup \{t_j\}_{j=0}^{n}$, $\{s_i\}_{i=0}^{2n}$ terms are sampled from $\{s_i^j\}_{j=1}^{c_i}$ and then embedded into $\{\mathbf{e}_i\}_{i=0}^{2n}$.

The training works at two levels:

- term-term pairs
- (term-relation)-term pairs

For the term-term sampling we define the label $\tau_{ij}$ in $b$ as:

$$\tau_{ij}^{term} = \begin{cases} 1 & c_i = c_j \\ 0 & c_i \neq c_j \end{cases}$$

And the similarity between terms in $b$ as:

$$S_{ij}^{term} = cos(\mathbf{e}_i, \mathbf{e}_j) \tag{2.16}$$

Then the positive and negative pairs are mined according to positive relative similarity. For an anchor $i$, it defines:

$$\mathscr{N}_i^{term} = \{j|\tau_{ij}^{term} = 0, S_{ij}^{term} > \min_{\tau_{ik}^{term}=1} S_{ik}^{term} - \epsilon\} \tag{2.17}$$

$$\mathscr{P}_i^{term} = \{j|\tau_{ij}^{term} = 1, S_{ij}^{term} < \max_{\tau_{ik}^{term}=0} S_{ik}^{term} + \epsilon\} \tag{2.18}$$

where $\epsilon$ is a margin.

CODER also learns Knowledge Graph embedding inspired by semantic matching methods, which aim to approximate $M_r^\top h \approx t$. Consider two triples $(h_0, r, t_0)$ and $(h_1, r, t_1)$, with the same relation $r$ and $h_0, h_1$ being semantically similar, may suggest that $t_0$ and $t_1$ are also semantically similar. We can define the (term-relation)-term similarity between relation $(s_i, r_i), \forall 1 \leq i \leq n$ and term $s_j, \forall n+1 \leq j \leq 2n$ in batch $b$ by:

$$S_{ij}^{rel} = \frac{e_i^\top M_{r_i} e_j}{\|M_{r_i}^\top e_i\| \|e_j\|} \tag{2.19}$$

where $M_{r_i} \in \mathbb{R}^{l \times l}$ and $\|\cdot\|$ is the L2-norm. For the term-term sampling, we define the label $\tau_{ij}^{rel}$ in $b$ as:

$$\tau_{ij}^{rel} = \begin{cases} 1 & t_i = t_{j-k} \\ 0 & t_i \neq t_{j-k} \end{cases}$$

Then the positive and negative (term-relation)-term pairs. For an anchor $(s_i, r_i)$, the sets are defined as:

$$\mathscr{N}_i^{rel} = \{j|\tau_{ij}^{rel} = 0, S_{ij}^{rel} > \min_{\tau_{ik}^{rel}=1} S_{ik}^{rel} - \epsilon^{rel}\} \tag{2.20}$$

$$\mathscr{P}_i^{rel} = \{j|\tau_{ij}^{rel} = 1, S_{ij}^{rel} < \max \tau_{ik}^{rel} = 0 S_{ik}^{rel} + \epsilon^{rel}\} \tag{2.21}$$

where $\epsilon^{rel}$ is a margin.

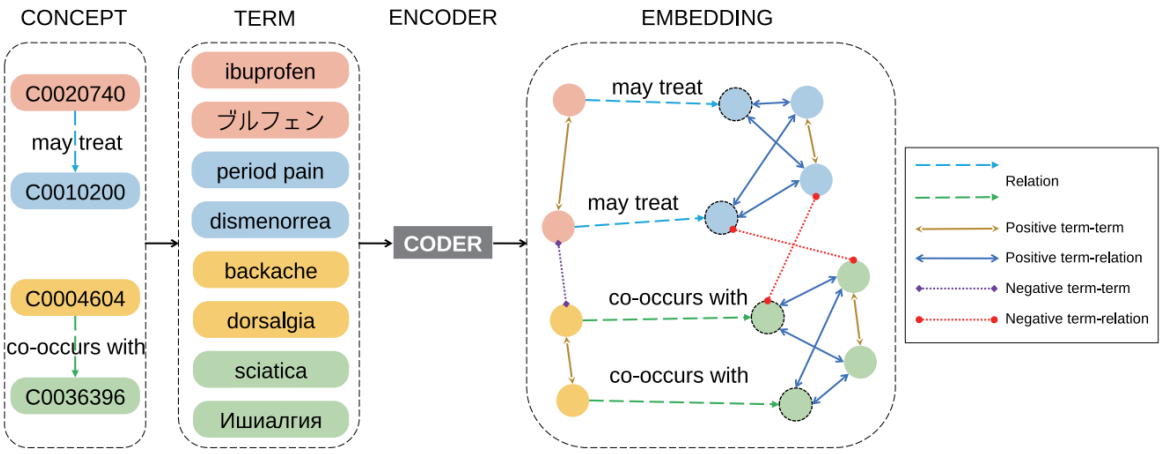
### Architecture

CODER architecture does not particularly differ from the pre-trained BERT model which it builds upon, the most important novelty is the contrastive training process, which can be appreciated in figure 2.4.

The loss adopted to exploit the positive and negative samples is Multi-Similarity Loss (MS-Loss)[49]. The margin loss function for *term* and *relation* similarity - assuming a batch $b$ with $n$ triples - is defined as follows:

$$\mathcal{L}_{MS}^z = \frac{1}{2n} \sum_{i=1}^{2n} \left( \frac{\log(1 + \sum_{j \in \mathscr{P}_i^z} \exp(-\alpha^z (S_{ij}^z - \lambda^z)))}{\alpha^z} + \frac{\log(1 + \sum_{j \in \mathscr{N}_i^z} \exp(\beta (S_{ij}^z - \lambda^z)))}{\beta^z} \right) \tag{2.22}$$

Figure 2.4: CODER contrastive learning process applied to UMLS concepts.




With $z \in \{term, rel\}$, where $\alpha^z, \beta^z, \lambda^z$ are hyperparameters.
The total loss function - applied on a batch $b$ - is defined as follows:

$$\mathcal{L} = \mathcal{L}_{MS}^{term} + \mu \mathcal{L}_{MS}^{rel} \tag{2.23}$$

After being trained with the strategy defined in the last sections, CODER can produce meaningful medical term contextual representations.

### 2.2.4 Our contribution

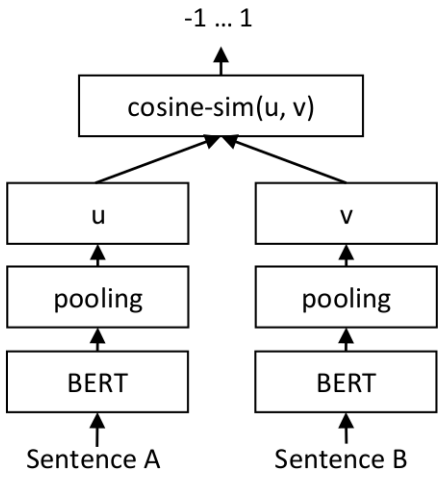
Although using CODER in its original form for our purposes could be a viable approach - and has been done, even as a baseline - we have taken an additional step of fine-tuning to attempt to enhance the embeddings. The fine-tuning process we have used is taken from [36], which has also made the `Sentence-Transformers` Python framework available[2]. A Sentence Transformer is essentially any architecture in which a pooling layer is placed on top of a Transformer model to generate a fixed-size embedding from any input text. The pooling is applied to the hidden states produced by BERT, as described in 2.14 and 2.15.

The fine-tuning process defined in [36] is based on Siamese networks. This type of neural network consists of two or more identical subnetworks. In this context, 'identical' means that they have the same configuration with the same parameters and weights, and parameter updating is mirrored across both subnetworks. This method is used to determine the similarity of inputs by comparing their feature vectors with cosine-similarity. A schematic of this architecture is illustrated in Figure 2.5.

In our case, we used this setting with CODER as the BERT model and applied the CLS pooling defined in 2.14 to remain consistent with CODER training. CODER model

[2] `https://github.com/UKPLab/sentence-transformers`

Figure 2.5: The Siamese network setting for a Sentence BERT model.




is publicly available on the HuggingFace hub[3].

### Fine-tuning Strategy

The choice we made for the fine-tuning strategy was based on the following idea:

- CODER uses all the relations present in the UMLS Knowledge Graph and weights them regardless of the semantic meaning
- For our scope we want to privilege the hierarchical relations, i.e. the `is-a` relation

The second need comes from the fact that we would like our term normalization service to be robust when it will have to deal with text labels that may be not present in the UMLS data CODER has been trained on. For example, the UMLS concept *Femoral Neck Fractures*(CUI C0015806) does not have an associated term in Italian language - that would be *Frattura del collo del femore* - and CODER may not be able to normalize correctly an input string like "*frattura del collo del femore*", even if the labels *frattura*, *femore* and *collo* are present in the Knowledge Graph. Giving importance to the hierarchical relations, we want to bring closer the embeddings of terms which are related on an ontological level. Doing this, we hope to have a broader embedding space area to handle cases in which the term normalization could not retrieve a specific concept and fall back to a broader concept. In this case we hypothesize that having the representations of *frattura* and *femore* near in the embedding space, while having *collo* at a certain distance, would be effective for the cases mentioned before.

Based on this principle we created a set of positive sentence pairs as follows:

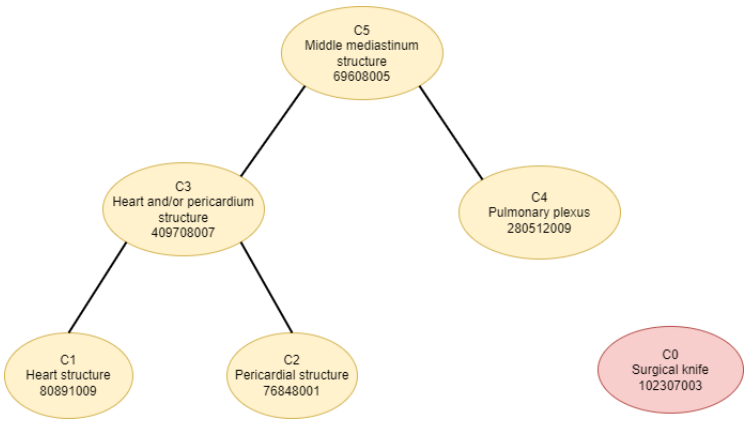
[3] `https://huggingface.co/GanjinZero/coder_all`

**Hard Positives** these are pairs of labels $s_1, s_2 \in \{s_i^j\}_{j=1}^{c_i}$, which refer to the same UMLS concept $c_i \in \mathcal{D}$, the score is $s = 1.0$

**Hierarchy Positives** these are pairs of labels $s_1, s_2$ with $s_1 \in \{s_i^j\}_{j=1}^{c_i}, s_2 \in \{s_h^j\}_{j=1}^{c_h}$, which lies on the same hierarchy and their score is some value $s \in ]0.0, 0.90]$

Since UMLS lacks a single hierarchy between its concepts and instead includes several relations that represent the hierarchy (such as `isa`, `related_broader`, `related_narrower`, etc.), we opted to use the hierarchies provided by the source dictionaries as a reference. Specifically, the hierarchy is established between AUIs and their corresponding terms. We selected the similarity measure proposed by [3], which has been shown to be appropriate for the biomedical domain, like our case.

Figure 2.6: A fragment of SNOMEDCT_US ontology, one of the source vocabularies present in UMLS Knowledge Graph. The yellow and red elements are part of hierarchies which are disjointed.




To consider the amount of common information between a pair of concepts, the authors proposed a similarity measure that calculates the ratio of non-shared knowledge to the sum of shared and non-shared knowledge. In this case, knowledge refers to the common ancestors of a concept. For example, let's refer to the SNOMED fragment shown in figure 2.6. For $(c_1, c_2)$, the number of common super concepts is 2, and the number of non-common super concepts plus the non-equal concept pair is 2. Hence, the distance between $(c_1, c_2)$ will be smaller than between $(c_3, c_4)$, as $2/(2+2) = 0.5$ and $2/(1+2) = 0.66$, respectively. This definition penalizes those cases in which the number of shared super concepts is small. It's worth noting that for $(c_0, c_i)$ with $i \in [1, 5]$, the distance between $(c_0, c_i)$ would be, $2/(2+0) = 1$ since the concept $c_0$ does not have any common super concepts with the other concepts.

To smooth the assessments and transform the function into a similarity, the authors also introduced an inverted logarithm function. Additionally, we normalized the co-domain between 0 and 1 by dividing the original measure by the logarithm of the union. Finally, to reserve the 1.0 score for the Hard Positives, which are not considered for this

measure, the upper bound of the co-domain is clipped to 0.90. We will refer to this measure as *Ontological similarity* from now on. The score $s$ of a Hierarchy Positive pair $(c_1, c_2)$ is their *Ontological similarity*:

$$s(c_1, c_2) = f_c\left(\frac{-\log_2 \frac{|T(c_1) \cup T(c_2)| - |T(c_1) \cap T(c_2)|}{|T(c_1) \cup T(c_2)|}}{\log_2 |T(c_1) \cup T(c_2)|}\right) \tag{2.24}$$

with

$$f_c(x) = \begin{cases} x & x < 0.9 \\ 0.9 & else \end{cases}$$

Let us define the full concept hierarchy ($H^C$) of concepts ($C$) of an ontology as a transitive `is-a` relation $H^C \in C \times C$, and we define $T(c_i) = \{c_j \in C \mid c_j \text{ is super concept of } c_i\} \cup \{c_i\}$ as the union of the ancestors of the concept $c_i$ and $c_i$ itself.

After defining our strategy for creating positive samples, we applied the process to the data selected for fine-tuning. We began with the UMLS Metathesaurus Knowledge Graph and filtered the subset of atoms (AUIs) based on the following conditions:

1. The atom has at least one hierarchy coming from the source vocabulary present in UMLS (this is stored in the `MRHIER` table).

2. The atom language is one of the following: English, Spanish or Italian.

3. The concept associated with the atom, also known as the CUI, belongs to a subset $C_{si}$ of $\mathcal{D}$. This subset is defined as $C_{si} = C_i \cup C_s$, where $C_i$ is a subset of $\mathcal{D}$ comprising concepts that have at least one term in the Italian language, and $C_s$ is defined similarly for Spanish terms.

The sql query to obtain all the atoms that meet the conditions and their related hierarchy is:

```
SELECT MRCONSO.CUI, MRCONSO.STR, MRCONSO.AUI, MRCONSO.SUI,
MRHIER.PTR
FROM MRCONSO
INNER JOIN MRHIER ON MRCONSO.AUI = MRHIER.AUI
WHERE (MRCONSO.LAT = 'ENG' OR MRCONSO.LAT = 'ITA' OR
MRCONSO.LAT = 'SPA')
```

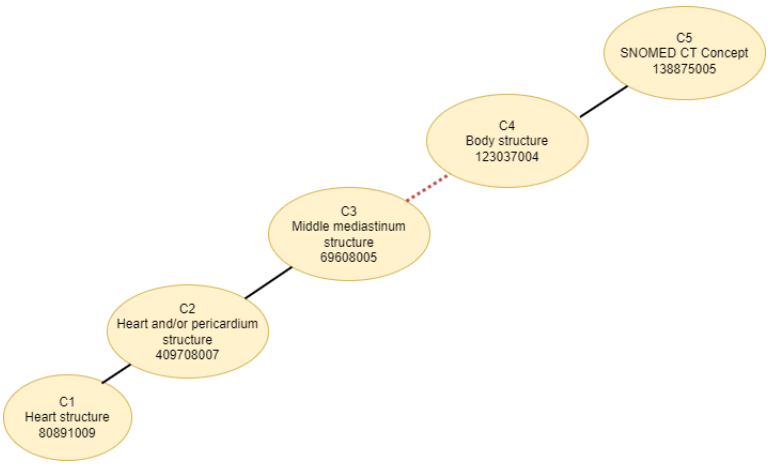
The algorithm 1 describes the process of producing a collection of paired atoms (the positive samples) with their corresponding similarity score. The algorithm takes as input a concept set $C$, and uses some auxiliary dictionaries: a dictionary $M_{ca}$ which maps a concept to its related atoms, and a dictionary $M_{ah}$ which maps an atom to its super concepts.

**Algorithm 1** The positive samples creation loop

```
d ← []                                          ▷ The collection which we will fill
for each c ∈ C do
    atoms ← M_ca[c]
    for each (a_i, a_j) ∈ atoms × atoms do
        p ← (a_i, a_j, 1.0)
        d.insert(p)                             ▷ Insert Hard positive with score 1.0
    end for
    for each a ∈ atoms do
        for each a_h ∈ M_ah[a] do
            s ← ontoScore(a, a_h)               ▷ The Ontological similarity function
            t ← (a, a_h, s)
            d.insert(t)                         ▷ Insert Hierarchical positive with score s
        end for
    end for
end for
```

To clarify, in the process of obtaining the hierarchy of an atom from $M_{ah}$, we remove the two highest-level ancestors in the hierarchy, which correspond to generic root atoms of the vocabulary. This can be seen in figure 2.7. This step is taken to filter out pairs that would be too general for the fine-tuning process.

Figure 2.7: An example of hierarchy route cutting on the same atoms of figure 2.6. The red dashed line between C3 and C4 represents where the route is cut for the Hierarchical samples creation process.




For example, if we assume that $C$ contains only one concept $c_o$ - which is 'Blood Clot' (CUI `C0302148`) - a fragment of the collection $d$ produced by the algorithm can be seen in table 2.2.

Table 2.2: A representative fragment of the collection resulting from concept 'Blood Clot' (CUI `C0302148`), using the algorithm 1. The first column is added to highlight the specific type of sample.

| | Positive Sample Type | Atom 1 | Atom 2 | Score |
|---|---|---|---|---|
| **0** | Hard | Blood Clot | clot | 1.0 |
| **1** | Hard | Blood Clot | Trombo | 1.0 |
| **2** | Hard | Blood Clot | Thrombi | 1.0 |
| **3** | Hierarchical | Blood Clot | Finding | 0.38 |
| **4** | Hierarchical | Blood Clot | Finding by Site or System | 0.61 |
| **5** | Hierarchical | Blood Clot | Cardiovascular System Finding | 0.90 |

After applying the positive samples creation strategy to the UMLS Metathesaurus Knowledge Graph, we obtained a collection of 80,531,855 pairs of atoms belonging to the concept subset $C_{si}$. We then removed the duplicate pairs based on the atoms' labels, resulting in a total of 51,814,562 pairs and 5,143,110 unique atoms. The fraction of different types of positive samples in the collection was as follows:

- 70% of Hard positive samples, atom pairs which refer to the same concept.
- 30% of Hierarchical positive samples, atom pairs which lies on the same hierarchy.

### Online Negative Mining and Training

In the previous section, we discussed fine-tuning CODER, and we faced the challenge of dealing with many positive samples (51,814,562). This made it impractical to run a classic training loop on the entire dataset. To overcome this challenge, we chose to extract a sample from the data collected by algorithm 1 for each epoch. This allowed us to run the fine-tuning loop for various epochs and examine how it affected the model embeddings.

Instead of using a negative sampling strategy similar to the one proposed in [6], which involves creating pairs of atoms with no common ancestor in their hierarchies, we chose to adopt a different approach. Our strategy focused on identifying atoms whose CODER embeddings were similar to the atom in the positive sample, but were not similar in the source ontology. We considered the concept of "closeness" to be based on cosine similarity in one case, and on the *ontological score* between the atoms in the other.

We developed an algorithm called the `negative mining function` (refer to algorithm 2) to implement the negative sample mining method. This function takes as input an atom $a$ and uses some auxiliary dictionaries: a dictionary $M_{ac}$ that maps each atom to its corresponding UMLS concept, a dictionary $M_{ap}$ that maps each atom to its positive sample. We also assume to have a function $f(e, n)$ that returns the $n$ atoms with the

nearest CODER embeddings for a given embedding $e$, based on cosine similarity. The function is used to mine a negative sample to pair with the input atom and create a Hard negative sample. Notably, the $f(e, n)$ function returns the atoms ordered by ascending distance, where the first atom returned is the nearest to $e$. Thus, returning the first element of the collection that satisfies our requirements also returns the nearest atom of that subset.

**Algorithm 2** The negative mining function

**Require:** $a$ is a given atom ▷ The atom for which we will mine a negative sample
  $s \leftarrow a.s$ ▷ A textual label of the atom
  $e \leftarrow embedding(s)$ ▷ The CODER embedding of s
  $d \leftarrow []$ ▷ An empty collection
  $pos \leftarrow M_{ap}[a]$
  **for each** $p \in pos$ **do**
    $c \leftarrow M_{ac}[p]$
    $d.insert(c)$ ▷ Insert concepts which appear as positive
  **end for**
  $nn \leftarrow f(e, n)$ ▷ Get the n atoms with nearest CODER embeddings to given atom
  **for each** $a_n \in nn$ **do**
    **if** $M_{ac}[a_n]$ is in $d$ **then**
      **pass**
    **else**
      **return** $a_n$
    **end if**
  **end for**

Let us consider the atom $a_0$ displayed in Table 2.2, which corresponds to the text label 'Blood Clot' and its related UMLS concept CUI `C0302148`, to illustrate the algorithm. Suppose we set $n = 12$. The function $f(e, n)$ is applied to the embedding of the atom *Blood Clot*, and the results are shown in Table 2.3. The table displays the 12 atoms with the nearest CODER embeddings to the input atom. To meet the negative sampling criteria stated before, we filter out from the resulting atoms those whose related CUI is not contained in the input atom CUI hierarchy. This leaves us with 'Clot' as a negative atom to pair with 'Blood Clot'. This pair forms a hard negative sample with a score of $s = 0.0$.

The online negative mining strategy takes advantage of the function $f(e, n)$ mentioned before. The nearest neighbor search is sped up by creating an index with all the atoms CODER embeddings, the framework used in the implementation is `Annoy`[4]. The online mining guarantees us that in each batch we will change the representation of an atom,

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

Table 2.3: The results, the function $f(e, n)$ for the embedding of the atom *Blood Clot* and $n = 12$.

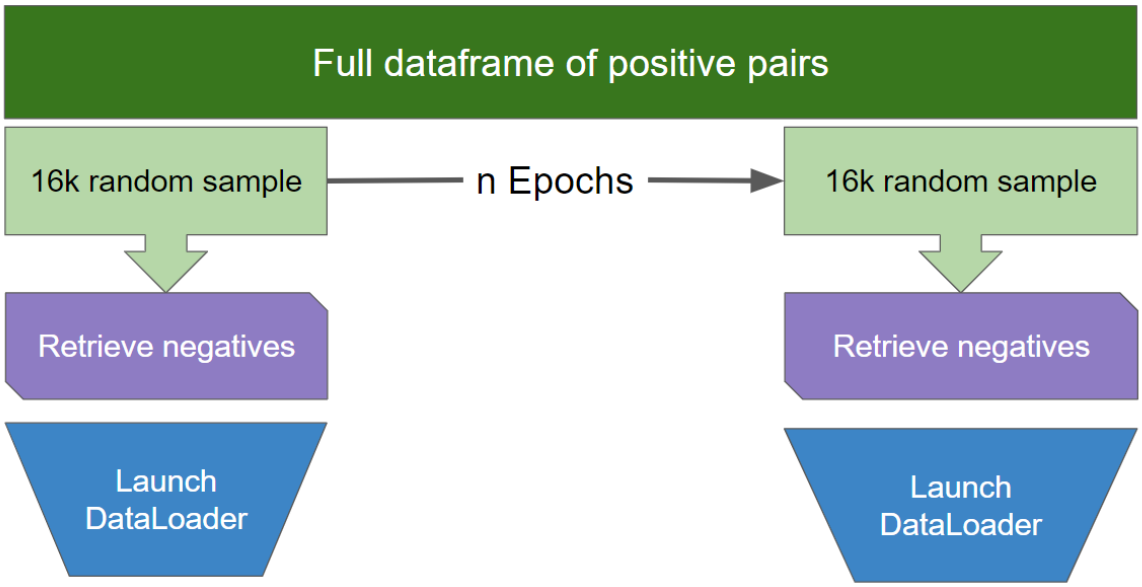
| | CUI | SUI | Atom |
|---|---|---|---|
| 0 | C0087086 | S6658250 | blood clot |
| 1 | C0087086 | S2070142 | Blood Clot |
| 2 | C0302148 | S0601741 | Blood clot |
| 3 | C0302148 | S1651270 | BLOOD CLOT |
| 4 | C0302148 | S6658250 | blood clot |
| 5 | C0302148 | S2070142 | Blood Clot |
| 6 | C0087086 | S0601727 | Blood Clots |
| 7 | C0302148 | S11857799 | blood clots |
| 8 | C0302148 | S0601727 | Blood Clots |
| 9 | C0009074 | S4035862 | CLOT |
| 10 | C0302148 | S0840496 | Clot |
| 11 | C0302148 | S11868755 | clot |

looking at both a positive and a negative sample. Furthermore, this approach gives us the future possibility to change the index used in the function $f(e, n)$ progressively during the fine-tuning, but this will be deepened in section 3.4. An overview of the fine-tuning process is presented in Figure 2.8.

The process creates a batch $b$ which has half positive samples and half negative samples, these pairs have a common atom. The loss used for the fine-tuning process is the `CosineSimilarityLoss`[5], the process follows the structure showed before in figure 2.5.

[5] `https://www.sbert.net/docs/package_reference/losses.html#cosinesimilarityloss`

Figure 2.8: Scheme of the fine-tuning loop with sampling and online negative mining. The purple box launches the algorithm 2 on each of the atoms present in the random sample to form 16k additional hard negatives.



# Chapter 3

# Experiments and Results

In this chapter, we will describe the experiments conducted for both of our approaches and present the corresponding results.

## 3.1 OntoEA

### 3.1.1 Downstream Task

For our experiments, we aimed to establish a benchmark for the Entity Alignment task in the biomedical domain, with a specific focus on the UMLS Knowledge Graph. To achieve this, we utilized the 2021 Large BioMed Track[1] proposed by the Ontology Alignment Evaluation Initiative (OAEI)[2]. This track features six tasks of increasing complexity and aims to find alignments between the Foundational Model of Anatomy (FMA), SNOMED CT, and the National Cancer Institute Thesaurus (NCI), all of which are part of the UMLS Knowledge Graph[35]. The reference alignments for this track are based on the UMLS Metathesaurus. We also leveraged MELT[13], a system that evaluates the performance of a given matcher, as part of the OAEI challenge.

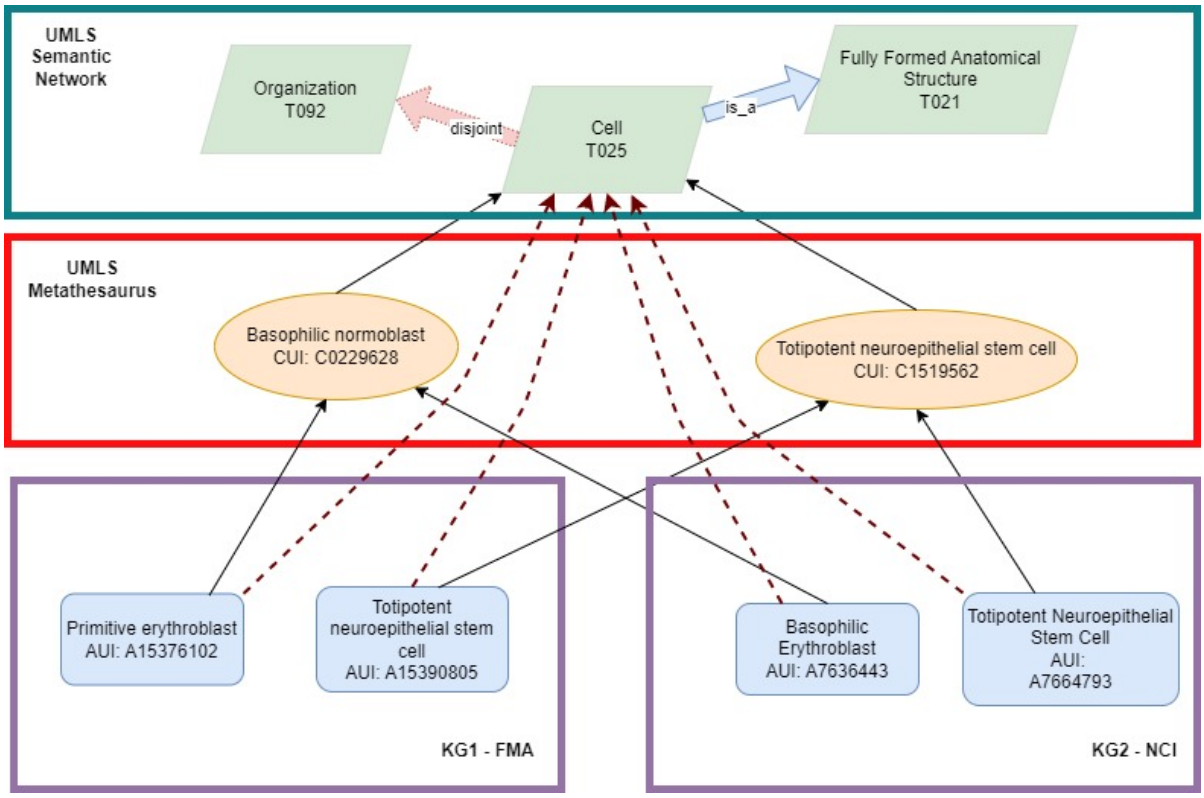
### 3.1.2 Data

Our first experiment aimed to match two relatively small fragments of the FMA and NCI, consisting of 3,696 entities (5% of FMA) and 6,488 entities (10% of NCI), respectively. As the task only provided the two knowledge graphs to align and not an ontology, we traced all the entities back to their UMLS atoms and used their Semantic Type as the entity class. The process is depicted in Figure 3.1. Consequently, we used a fragment of the UMLS Semantic Network as the ontology for the Entity Alignment using OntoEA.

[1] http://www.cs.ox.ac.uk/isg/projects/SEALS/oaei/2021/

[2] http://oaei.ontologymatching.org/

Figure 3.1: An example of how we mapped the vocabularies entities to the Semantic Types to create an artificial ontology. The dashed lines will be the membership links in our setting, the solid lines are extracted from UMLS Metathesaurus.




### 3.1.3 Metrics

For evaluating the alignment between the given fragments, we used the following metrics provided by MELT:

- **Precision**: measures the proportion of correctly aligned entity pairs in the output alignment
- **Recall**: measures the proportion of correctly aligned entity pairs in the reference alignment that are also correctly aligned in the output alignment
- **F1-measure**: measures the harmonic mean of precision and recall

For example, if the matcher aligns entities $(e_i, e_j)$ such that $e_i \in E_i$ and $e_j \in E_j$, but in the reference alignment, $e_i$ is aligned with $e_k \in E_j$, then this alignment is deemed a false positive.

### 3.1.4 Results

Table 3.1 presents the results of the original implementation of OntoEA, our PyTorch implementation, and Fine-TOM[19], which ranked 10th out of the 12 participants in the

challenge task. Fine-TOM is included as a reference baseline for comparison.

Table 3.1: This table shows the results of the models that participated in Task 1 of OAEI 2021. Fine-TOM, which ranked 10th in the challenge, is included to provide additional context for the performance of the OntoEA model.

| | **Precision** | **Recall** | **F1-measure** |
|---|---|---|---|
| **Original Model** | .264 | .212 | .235 |
| **Torch porting** | .263 | .21 | .234 |
| **Fine-TOM** | .949 | .277 | .429 |

The performance of OntoEA on the task fell slightly short of our expectations. However, the use of neural models in this field is a relatively new approach. In fact, none of the 12 algorithms that participated in the OAEI task utilized a purely neural approach like OntoEA. The baseline model that we used was the most similar to our approach, utilizing a matching pipeline that employed a Transformer filter and various fine-tuned BERT models, all different variants of *albert-base-v2*[21], to generate a confidence score for an existing alignment. However, given these factors, we decided to explore alternative approaches and discontinue the use of the aforementioned method.

## 3.2 CODER

We decided to fine-tune CODER with three different settings:

**HierCODER** : this version follows the full fine-tuning strategy defined in 2.2.4, with a *learning rate* of $2e-5$

**FinerCODER** : this version was fine-tuned with a smaller learning rate of $1e-8$ to maintain consistency with the initial embeddings.

**FocusCODER** : this version has been fine-tuned with different *learning rates*: $2e-5$ for the positive samples and $1e-8$ for the negative samples.

It is important to note that, as stated in section 2.2.4, it was impractical to train a model on all the positive samples resulting from our positive mining strategy due to limited computational and time resources. Therefore, we fine-tuned the models on a variable number of samples as shown in Table 3.2 and evaluated them using two different settings: (1) considering only the terms seen by the model during fine-tuning and (2) considering terms that the model did not encounter.

The idea of using different learning rates stemmed from the fact that the positive and hierarchical pairs are implicitly derived from the UMLS Knowledge Graph structure, whereas the negative sampling strategy defined in section 2.2.4 draws from the

Table 3.2: The number of samples seen by HierCODER, FinerCODER and FocusCODER during the fine-tuning. The percentages are in proportion to the total (51,814,562) of positive samples generated by Algorithm 1.

| | **HierCODER** | **FinerCODER** | **FocusCODER** |
|---|---|---|---|
| **n of terms** | 1612206 | 399948 | 399948 |
| **Percentage of samples** | 3.16% | 0.78% | 0.78% |

nearest CODER embeddings and filters using the concepts related to a given atom. This approach may result in noisy negative mining because, as pointed out in [16], the graph structure has several inconsistencies. Giving more weight to the positive samples will help us evaluate the effectiveness of our negative mining strategy.

As there is no annotated dataset specific to our case for the task of medical term normalization performed by the embedding model, we turned to several benchmarks from the literature, such as [17] and [58], which are two English medical term normalization datasets used in some experiments mentioned in [54]. Although the authors referred to the multi-language dataset [20], it excludes Italian and maps terms only to a subset of UMLS sources. Therefore, we opted to use different evaluation methods in the absence of a gold standard for our task, which we will outline in the following sections.

### 3.2.1 Semantic Group Classification

A possible downstream task that can assess the quality of medical term embeddings for UMLS is Semantic Group Classification. As we mentioned in section 1.1, UMLS comprehends a Semantic Network which attributes one or more semantic types to each of the concepts. Using the term embeddings as features to feed a classifier is a possible way of evaluating their quality.

#### Data

Instead of the Semantic Types, we used the Semantic Groups. A coarser categorization, which creates clusters of semantically similar types. And from all the 15 Semantic Groups, we selected a subset of the four most common groups in our case: Disorders, Substances, Procedures and Anatomy. As mentioned before, we employed two different datasets for the evaluation: one with terms seen by the fine-tuned model and one with terms not seen by the model. For each term, we associated the related semantic group and filtered the semantic groups we were not interested in. We used a fixed sample of 10,000 data points.

### Metrics

The evaluation metrics were the standard ones for a multi-class classification task:

- **Precision:** Precision measures how many of the predicted positive instances are actually positive. It is calculated as the ratio of true positive predictions to the sum of true positive and false positive predictions.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall:** Recall measures how many of the actual positive instances are correctly predicted as positive. It is calculated as the ratio of true positive predictions to the sum of true positive and false negative predictions.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1 Score:** The F1 score is the harmonic mean of precision and recall. It is a single score that combines the two measures to give an overall performance score. It is calculated as 2 times the product of precision and recall, divided by their sum.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **Accuracy:** Accuracy measures the overall correctness of the model's predictions. It is calculated as the ratio of the number of correct predictions to the total number of predictions made.

$$\text{Accuracy} = \frac{C}{N}$$

### Results

We ran the experiment described before for each fine-tuned model on a 5-fold cross-validation setting, for both the datasets. The results are reported in table 3.3, the metrics are averaged over the cross-validation folds. We used a linear classifier, trained for 10 epochs with a *learning rate* of $5e-2$, for each of the 5 folds.

These results provide insight into the effectiveness of the two sides of our fine-tuning strategy. HierCODER produced significantly worse results compared to CODER, the model performed even better on unseen terms. However, the increasingly better metrics of FinerCODER and FocusCODER respectively, suggest that focusing on the positive samples is the right approach and that our negative samples were not useful for the process, and may have even worsened the embeddings. It's worth noting that CODER was exposed to different orders of magnitude of training data, as the multilingual model has been trained for a million steps, with 128 triplets per training step, as stated in [54]. So, CODER has seen 128M samples.

| | | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|---|
| CODER | Seen | .930 | .904 | .930 | .914 |
| | Unseen | - | - | - | - |
| HierCODER | Seen | .422 | .534 | .422 | .384 |
| | Unseen | .446 | .530 | .446 | .416 |
| FinerCODER | Seen | .882 | .830 | .882 | .846 |
| | Unseen | .924 | .898 | .924 | .900 |
| FocusCODER | Seen | .924 | .900 | .924 | .908 |
| | Unseen | .912 | .888 | .912 | .896 |

Table 3.3: The average accuracy, precision, recall and F1-score for each of the fine-tuned CODER models on the seen and unseen terms semantic group classification.

### 3.2.2 MedMentions

Another task that we considered, to assess the quality of our embeddings, is MedMentions entity linking. The MedMentions dataset provides a valuable resource for linking biomedical concepts, as described in [31]. Specifically, the task of concept linking in MedMentions involves linking a given entity mention (in English language) to its corresponding UMLS concept. This task is similar to CODER's Medical Term Normalization task and presents significant challenges in the biomedical domain due to the similarity between different concept names and abbreviations. For example, the abbreviation "BMI" can refer to either "Body Mass Index" or "Bone Marrow Infiltration". MedMentions is a large and diverse dataset with various entity types and contexts, making it an excellent resource for developing and evaluating biomedical entity and concept linking systems.

#### Data

The data for this study was sourced from PubMed Central (PMC)[3], a large collection of journal literature in the life sciences and biomedical fields. The MedMentions corpus was created from 4,392 abstracts randomly selected from those released on PubMed between January 2016 and January 2017, and includes 352,496 total mentions, of which 34,724 are unique UMLS concepts. To ensure accurate annotation, a team of professional annotators with extensive experience in biomedical content curation was recruited to exhaustively annotate UMLS entity mentions from the abstracts. Due to the size of the UMLS, only about 1% of its concepts are covered in MedMentions, making a zero-shot learning approach the most appropriate. The complete corpus is publicly available on GitHub[4].

[3] `https://www.ncbi.nlm.nih.gov/pmc/`

[4] `https://github.com/chanzuckerberg/MedMentions`

### Metrics

The evaluation metric for the MedMentions concept linking task is Hits@n, as defined in section 2.1.4, where the considered values for the parameter are $n \in [1, 4]$. Hits@n measures the position of the ground truth concept link in the ranking generated by the nearest neighbor search performed on the CODER embeddings of the entity mentions in the dataset. The search is carried out on an Annoy index, as described in section 2.2.4, which is constructed on the embedding space of the CODER model.

### Results

We ran the MedMentions entity linking task on our three different fine-tuning versions of CODER. The entity linking experiment has been conducted with a zero-shot setting, the results with the metrics defined before are reported in table 3.4.

Table 3.4: The results of CODER, HierCODER, FinerCODER and FocusCODER on the MedMentions entity linking task, with a zero-shot setting.

| | **H@1** | **H@2** | **H@3** | **H@4** |
|---|---|---|---|---|
| **CODER** | .59 | .69 | .73 | .75 |
| **HierCODER** | .28 | .35 | .38 | .40 |
| **FinerCODER** | .58 | .67 | .71 | .73 |
| **FocusCODER** | .55 | .65 | .70 | .73 |

These metrics confirm the results we reported in Section 3.2.1. The models that prioritized the positive samples yielded results comparable to CODER, while HierCODER performed worse than the other three. This clearly indicates that the negative samples mined have a negative effect on the embeddings and raises new doubts about the consistency of the UMLS structure.

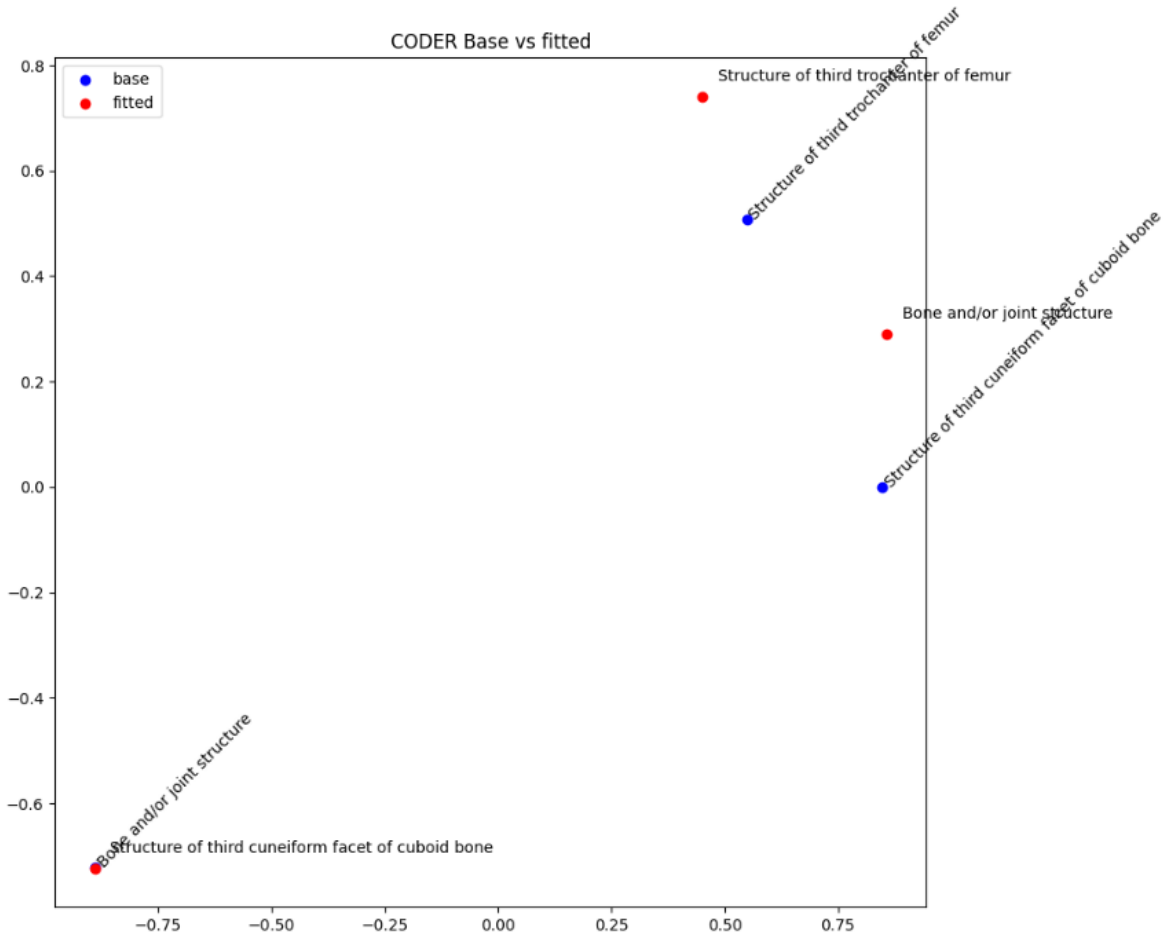
## 3.3 Error Analysis

To better understand why our full fine-tuning strategy (HierCODER) did not have a positive impact on the embeddings, we realized that we may not have obtained enough information from the evaluation metrics on the task that we reported in the previous section. Therefore, we attempted to visualize the evolution of the dense embeddings by applying Principal Component Analysis (PCA) to reduce the dimensionality, and then used t-Distributed Stochastic Neighbor Embedding (t-SNE)[28] to plot the embeddings on a two-dimensional map. Specifically, we used the openTSNE library[5], which is open-source and publicly available on GitHub. We chose openTSNE because it is currently the

[5] `https://opentsne.readthedocs.io/en/stable/`

only library that enables embedding new points into an existing embedding. This allowed us to plot both the CODER and the HierCODER embeddings to better appreciate the effects of the fine-tuning.

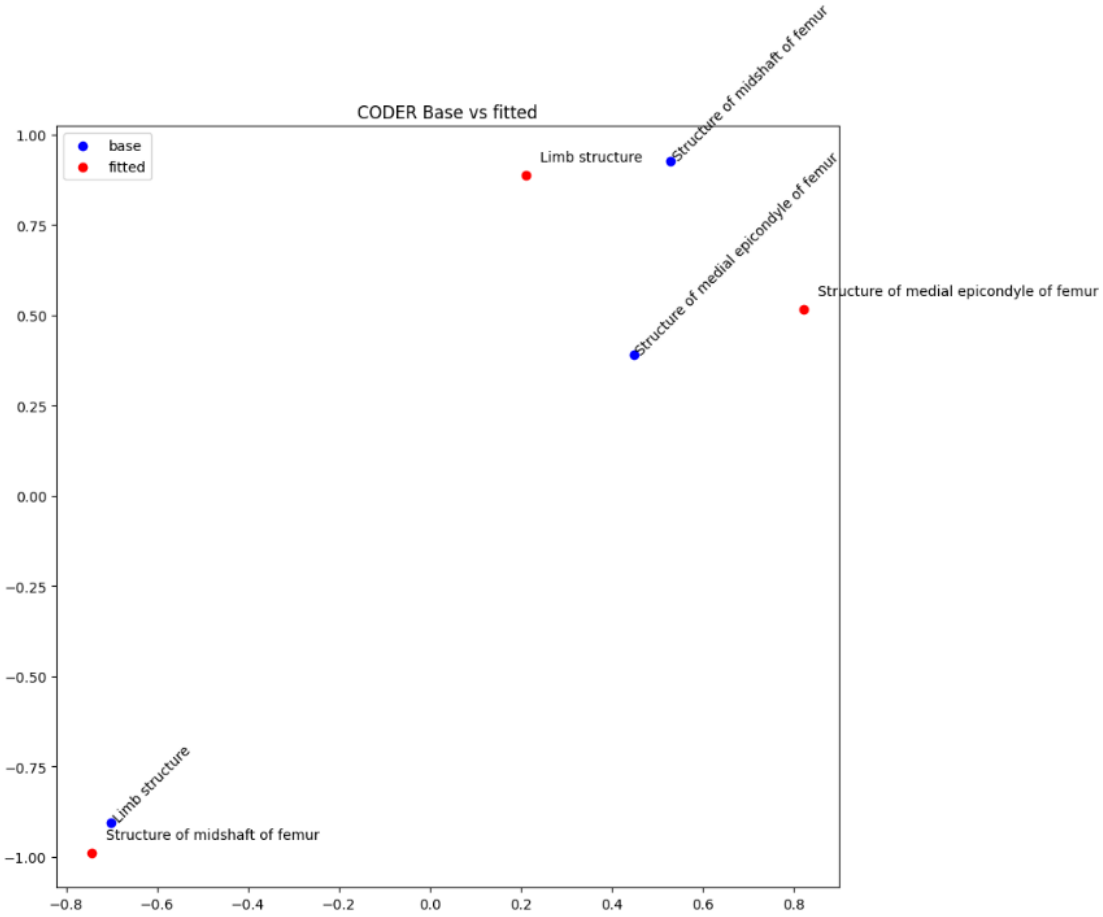
Figure 3.2: A two-dimensional visualization of three dense embeddings, with the CODER embeddings marked in blue and the HierCODER embeddings in red. In this case, the 'Structure of the third trochanter of the femur' is the focal point, while 'Bone and/or joint structure' is part of the positive hierarchical sample and 'Structure of the third cuneiform facet of the cuboid bone' is the paired negative atom.




In figure 3.2, it is displayed the scatter plot of the dense embeddings of three terms: 'Structure of the third trochanter of the femur', 'Bone and/or joint structure' and 'Structure of the third cuneiform facet of the cuboid bone'. The first one forms a hierarchical sample with the second and a negative sample with the third, respectively. The scatter points are blue if the term representation comes from CODER and red if the term representation comes from HierCODER. From this approximation, we can assume that the fine-tuning process does what is expected to do. In fact, the super concept (i.e., 'Bone and/or joint structure') embedding is nearer to the 'Structure of the third trochanter of

the femur' embedding when using HierCODER. Also, 'Structure of the third cuneiform facet of the cuboid bone' embedding - which is not a more general concept, but rather a different one that is very lexically similar to 'Structure of the third trochanter of the femur' - is more distant in the HierCODER embedding space.

Figure 3.3: A two-dimensional visualization of three dense embeddings, with the CODER embeddings marked in blue and the HierCODER embeddings in red. In this case, the 'Structure of medial epicondyle of femur' is the focal point, while 'Limb structure' is part of the positive hierarchical sample and 'Structure of midshaft of femur' is the paired negative atom.




The scatter plot in Figure 3.3 illustrates the dense embeddings of three terms: 'Structure of medial epicondyle of femur', 'Limb structure', and 'Structure of midshaft of femur'. The first term forms a hierarchical sample with the second and a negative sample with the third. The setting is similar to the one presented earlier. However, the samples used in this case were not utilized in the fine-tuning process. This implies that HierCODER can generalize the embedding differences for two comparable cases, where both contain the term 'femur' in this instance.

Figure 3.4: A two-dimensional visualization of three dense embeddings, with the CODER embeddings marked in blue and the HierCODER embeddings in red. In this case, the 'Methyldopate hydrochloride' is the focal point, while 'L-Tyrosine, 3-hydroxy-alpha-methyl-, Ethyl Ester, Hydrochloride' is part of the positive hierarchical sample and 'Manidipine hydrochloride' is the paired negative atom.

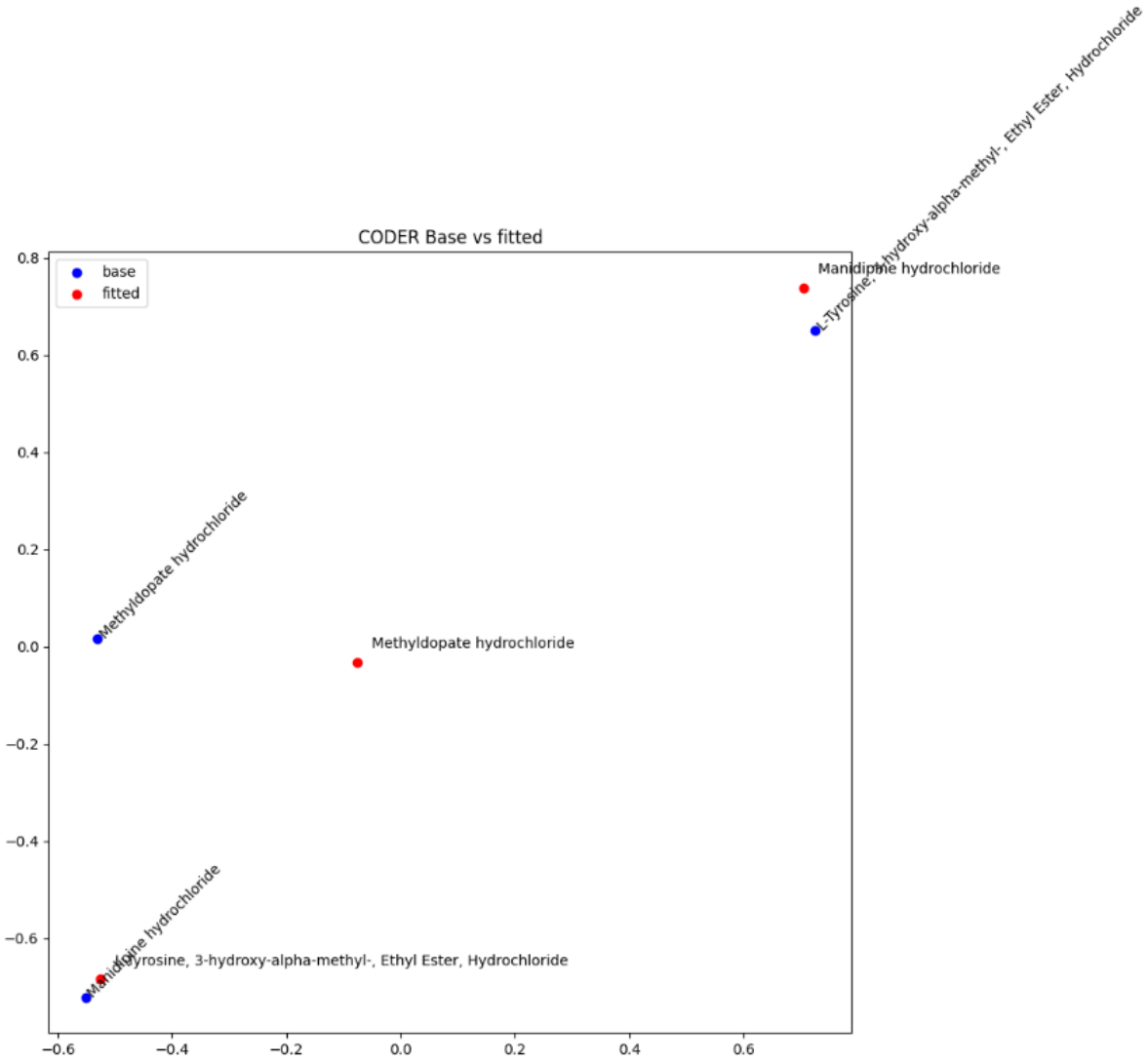



Figure 3.3 presents a scatter plot of the dense embeddings of three terms: 'Methyldopate hydrochloride', 'L-Tyrosine, 3-hydroxy-alpha-methyl-, Ethyl Ester, Hydrochloride', and 'Manidipine hydrochloride'. The first term forms a hierarchical sample with the second and a negative sample with the third. The setting is identical to the previous cases. However, in this instance, none of the samples related to other concepts were used in the fine-tuning process. Notably, there is no clear difference in the distance between the embeddings for the positive and negative samples. Although the embeddings have been influenced by the fine-tuning process, which may result in inconsistencies in the embedding space.

Based on this error analysis, we can extrapolate the following points:

1. The fine-tuning process had the intended effect of reducing the distance between terms that belong to the same hierarchy or refer to the same concept, while increasing the distance between them and the negatives mined using the strategy defined in section 2.2.4.

2. However, when tested on relevant tasks, the resulting contextual embedding space performed worse than our starting point (i.e., CODER).

3. The analysis conducted on both seen and unseen samples suggests that a longer and more accurate fine-tuning process could positively impact the embeddings. This will be further explored in the next section.

While considering these points, it is important to remember that we cannot evaluate the performance of the embeddings for our specific objective due to the absence of an annotated dataset. The tasks we used are only proxies for assessing the quality of the embeddings.

## 3.4 Discussion

In this dissertation, we employed the OntoEA architecture for Entity Alignment between two biomedical ontologies, although the results were not promising. One possibility for future work would be to extend the knowledge graph input fed to the model to create the entity embeddings. In fact, we only used part of the OntoEA modules. The authors state that it is possible to create an embedding based on entity relations, membership, and attributes. Moreover, we only used the hierarchical relationships based on our hypothesis that such relations are more important than others to obtain a good dense embedding. Another approach could be to arbitrarily weight the different types of relations.

In addition, we introduced a novel fine-tuning strategy for CODER (and any BERT-based language model), which could be tailored to other applications. For instance, one could modify the scoring criterion for positive samples or use a different loss function to consider more samples for a single concept. The online negative mining approach could be adjusted with various settings. With sufficient computational resources, an Annoy index generated from the model itself could be used in the negative mining function. This may be a viable future approach to address the problematic aspects of the embeddings.

Although there are various possible adjustments to try to overcome the limitations we encountered, it is worth considering that we may have reached a plateau in the expressiveness of Transformer-based models when it comes to tackling Knowledge Graph Embedding. Language models, particularly Large Language Models like LaMDA, T5, and the GPT family have been successful in various Natural Language Processing tasks,

including entity extraction and linking, due to their learning objective of reproducing highly plausible linguistic patterns. However, there are still several limitations of Language Models in regard to factual consistency and knowledge inference. This suggests that the language representation provided by this approach alone may not be sufficient, particularly for tasks that involve inconsistent or incomplete knowledge bases like UMLS. For example, let's assume we have a medical report that says:

> caduta accidentale con trauma chiuso emicostato dx e **frattura del rene** DX a tutto spessore, emodinamicamente stabile

Here, the text in bold needs to be normalized into a UMLS concept. Using CODER embeddings, the linked concept is "Fractures, Tooth" (CUI `C0040441`), while the correct one would be "Rupture of kidney" (CUI `C0347648`). Even though the knowledge graph structure clearly separates concepts related to the kidney from those related to the tooth, the linked concept is still correct if we replace "frattura del rene" with "rottura del rene".

# Chapter 4

# Conclusions

In this thesis, we explored the task of enhancing graph embedding in the biomedical domain with ontological knowledge using two different approaches. Firstly, we used OntoEA, a neural model designed for ontology alignment tasks, and then fine-tuned CODER, a state-of-the-art language model, with a novel positive and negative sampling strategy. Unfortunately, the overall results fell short of expectations, and the models' performance did not exceed the state of the art for the evaluated tasks.

However, this study can be seen as a starting point for future research in the field. The use of OntoEA is a relatively new approach, and fine-tuning a language model like CODER has not been extensively explored. The results obtained can provide valuable insights for future work, and the approaches proposed in this thesis can serve as a baseline for future experiments. Furthermore, we have made a PyTorch version of the OntoEA architecture publicly available, which could be beneficial for future integrations.

In summary, while the overall performance of the models was not satisfactory, this thesis highlights the potential of using knowledge graph embedding in the biomedical domain and provides a foundation for further research in the field. As a byproduct, we also created a PyTorch porting of OntoEA, which could be useful for the community.

To improve the accuracy of biomedical entity extraction and entity linking, Language Models can benefit from leveraging ontological reasoning on knowledge graphs. Knowledge graphs like the Unified Medical Language System (UMLS) contain a vast set of structured and interrelated concepts and relationships that can enhance the quality of concept coding in biomedical text. However, the incompleteness and potential inconsistencies of such knowledge graphs pose a significant threat to the application of traditional Knowledge Representation and reasoning methods. Inconsistencies in the knowledge graph can lead to contradictions, hampering the ability to reason about the relationships between concepts and leading to incorrect or impossible conclusions. Incompleteness can severely limit the ability of traditional reasoning methods to infer new knowledge, again leading to inconsistent conclusions, particularly under the closed-world

assumption (where a statement that is not known to be true is false by definition). Moreover, knowledge graphs are often vast and complex, as is the case with UMLS, making it challenging to identify and correct inconsistencies and missing knowledge. Additionally, relying solely on the term as input and disregarding the context can only achieve a certain level of expressiveness, even if the dense embedding is a good representation. For example, all Transformer-based models (e.g., SciBERT, BioBERT, CODER, etc.) that have been evaluated on MedMentions have reached approximately 72% accuracy, and [1], the state-of-the-art model that scored 76% accuracy, leverages the mention context document to enhance the linking quality.

In recent years, neuro-symbolic artificial intelligence, an area of AI research that aims to integrate the advantages of neural networks and symbolic reasoning while compensating for their respective limitations, has garnered significant interest among the AI research community and has demonstrated competitiveness or out performance of the state-of-the-art in several tasks. In [14], the authors introduce a novel model architecture, named recursive reasoning network, that can accurately perform ontology reasoning on multiple benchmarks, including inferring class membership and relations for UMLS concepts.

In the upcoming sprint of our research project, we aim to explore the integration of neuro-symbolic reasoning and learning on knowledge graphs, such as UMLS, with state-of-the-art biomedical language models like CODER. Our objective is to enhance the accuracy and efficiency of our biomedical annotation system.

One of the main challenges in entity disambiguation is the limited context available to infer the most plausible candidates. To tackle this issue, we plan to incorporate additional context, such as the full sentence or the set of entity candidates extracted from the sentence. This context would aid in inferring the most plausible candidates based on factual evidence and the learned representation of the knowledge graph, thereby improving the accuracy of the entity disambiguation task.

Furthermore, we intend to utilize available unambiguous entities as reasoning clues to disambiguate concepts or identify the most suitable combination of concepts to represent the relevant entities mentioned in the sentence. By leveraging the information and relationships between concepts in the knowledge graph, we aim to improve the accuracy of the entity linking process.

# Bibliography


[1] Dhruv Agarwal et al. "Entity Linking via Explicit Mention-Mention Coreference Modeling". In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Seattle, United States: Association for Computational Linguistics, July 2022, pp. 4644–4658. DOI: 10.18653/v1/2022.naacl-main.343. URL: https://aclanthology.org/2022.naacl-main.343.

[2] Ivana Balažević, Carl Allen, and Timothy M Hospedales. "Tucker: Tensor factorization for knowledge graph completion". In: *arXiv preprint arXiv:1901.09590* (2019).

[3] Montserrat Batet, David Sánchez, and Aida Valls. "An ontology-based measure to compute semantic similarity in biomedicine". In: *Journal of Biomedical Informatics* 44.1 (2011). Ontologies for Clinical and Translational Research, pp. 118–125. ISSN: 1532-0464. DOI: https://doi.org/10.1016/j.jbi.2010.09.002. URL: https://www.sciencedirect.com/science/article/pii/S1532046410001346.

[4] Iz Beltagy, Kyle Lo, and Arman Cohan. "SciBERT: A Pretrained Language Model for Scientific Text". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3615–3620. DOI: 10.18653/v1/D19-1371. URL: https://aclanthology.org/D19-1371.

[5] Piotr Bojanowski et al. "Enriching Word Vectors with Subword Information". In: *arXiv preprint arXiv:1607.04606* (2016).

[6] Antoine Bordes et al. "Translating Embeddings for Modeling Multi-relational Data". In: *Advances in Neural Information Processing Systems*. Ed. by C.J. Burges et al. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf.

[7] Muhao Chen et al. *Multilingual Knowledge Graph Embeddings for Cross-lingual Knowledge Alignment*. 2016. DOI: 10.48550/ARXIV.1611.03954. URL: https://arxiv.org/abs/1611.03954.

[8] Tim Dettmers et al. “Convolutional 2d knowledge graph embeddings”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[9] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: (2018). DOI: 10.48550/ARXIV.1810.04805. URL: https://arxiv.org/abs/1810.04805.

[10] Shobeir Fakhraei, Joel Mathew, and José Luis Ambite. “NSEEN: Neural Semantic Embedding for Entity Normalization”. In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part II*. Würzburg, Germany: Springer-Verlag, 2019, pp. 665–680. ISBN: 978-3-030-46146-1. DOI: 10.1007/978-3-030-46147-8_40. URL: https://doi.org/10.1007/978-3-030-46147-8_40.

[11] Daniel Faria et al. “The AgreementMakerLight Ontology Matching System”. In: *On the Move to Meaningful Internet Systems: OTM 2013 Conferences*. Ed. by Robert Meersman et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 527–541. ISBN: 978-3-642-41030-7.

[12] Junheng Hao et al. “Universal Representation Learning of Knowledge Bases by Jointly Embedding Instances and Ontological Concepts”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’19. Anchorage, AK, USA: Association for Computing Machinery, 2019, pp. 1709–1719. ISBN: 9781450362016. DOI: 10.1145/3292500.3330838. URL: https://doi.org/10.1145/3292500.3330838.

[13] Sven Hertling, Jan Portisch, and Heiko Paulheim. “MELT - Matching EvaLuation Toolkit”. In: *Semantic Systems. The Power of AI and Knowledge Graphs - 15th International Conference, SEMANTiCS 2019, Karlsruhe, Germany, September 9-12, 2019, Proceedings*. 2019, pp. 231–245. DOI: 10.1007/978-3-030-33220-4_17. URL: https://doi.org/10.1007/978-3-030-33220-4%5C_17.

[14] Patrick Hohenecker and Thomas Lukasiewicz. “Ontology Reasoning with Deep Neural Networks”. In: *Journal of Artificial Intelligence Research* 68 (2020). DOI: 10.1613/jair.1.11661. URL: https://doi.org/10.1613%5C%2Fjair.1.11661.

[15] Ernesto Jiménez-Ruiz et al. “Large-scale interactive ontology matching: Algorithms and implementation”. In: *Frontiers in Artificial Intelligence and Applications* 242 (Jan. 2012), pp. 444–449. DOI: 10.3233/978-1-61499-098-7-444.

[16] Ernesto Jiménez-Ruiz et al. “Logic-based assessment of the compatibility of UMLS Ontology Sources”. In: *Journal of Biomedical Semantics* 2.Suppl 1 (2011). DOI: 10.1186/2041-1480-2-s1-s2.

[17] Sarvnaz Karimi et al. "CADEC: A corpus of adverse drug event annotations". In: *Journal of Biomedical Informatics* 55 (2015), pp. 73–81. DOI: 10.1016/j.jbi.2015.03.010.

[18] Klyne et al. "Resource Description Framework (RDF): Concepts and Abstract Syntax". In: (Jan. 2004).

[19] Leon Knorr and Jan Portisch. "Fine-TOM matcher results for OAEI 2021". In: *OM@ISWC*. 2021.

[20] Jan Kors et al. "A multilingual gold-standard corpus for biomedical concept recognition: The Mantra GSC". In: *Journal of the American Medical Informatics Association : JAMIA* 22 (May 2015). DOI: 10.1093/jamia/ocv037.

[21] Zhenzhong Lan et al. *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*. 2019. DOI: 10.48550/ARXIV.1909.11942. URL: https://arxiv.org/abs/1909.11942.

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

[23] Jinhyuk Lee et al. "BioBERT: a pre-trained biomedical language representation model for biomedical text mining". In: *Bioinformatics* 36.4 (Sept. 2019). Ed. by Jonathan Wren, pp. 1234–1240. DOI: 10.1093/bioinformatics/btz682.

[24] Manuel Leone et al. "A Critical Re-Evaluation of Neural Methods for Entity Alignment". In: *Proc. VLDB Endow.* 15.8 (June 2022), pp. 1712–1725. ISSN: 2150-8097. DOI: 10.14778/3529337.3529355. URL: https://doi.org/10.14778/3529337.3529355.

[25] Haodi Li et al. "CNN-based ranking for biomedical entity normalization". In: *BMC Bioinformatics* 18.11 (Oct. 2017), p. 385. ISSN: 1471-2105. DOI: 10.1186/s12859-017-1805-7. URL: https://doi.org/10.1186/s12859-017-1805-7.

[26] Fangyu Liu et al. "Self-Alignment Pretraining for Biomedical Entity Representations". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, June 2021, pp. 4228–4238. DOI: 10.18653/v1/2021.naacl-main.334. URL: https://aclanthology.org/2021.naacl-main.334.

[27] Xin Lv et al. "Differentiating Concepts and Instances for Knowledge Graph Embedding". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 1971–1979. DOI: 10.18653/v1/D18-1222. URL: https://aclanthology.org/D18-1222.

[28] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.

[29] Zulfat Miftahutdinov and Elena Tutubalina. "Deep Neural Models for Medical Concept Normalization in User-Generated Texts". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 393–399. DOI: 10.18653/v1/P19-2055. URL: https://aclanthology.org/P19-2055.

[30] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. DOI: 10.48550/ARXIV.1301.3781. URL: https://arxiv.org/abs/1301.3781.

[31] Sunil Mohan and Donghui Li. *MedMentions: A Large Biomedical Corpus Annotated with UMLS Concepts*. 2019. DOI: 10.48550/ARXIV.1902.09476. URL: https://arxiv.org/abs/1902.09476.

[32] Deepak Nathani et al. "Learning attention-based embeddings for relation prediction in knowledge graphs". In: *arXiv preprint arXiv:1906.01195* (2019).

[33] Dai Quoc Nguyen et al. "A novel embedding model for knowledge base completion based on convolutional neural network". In: *arXiv preprint arXiv:1712.02121* (2017).

[34] Maximilian Nickel, Volker Tresp, Hans-Peter Kriegel, et al. "A three-way model for collective learning on multi-relational data." In: *Icml*. Vol. 11. 10.5555. 2011, pp. 3104482–3104584.

[35] Bodenreider O. *The Unified Medical Language System (UMLS): integrating biomedical terminology*. 2004. DOI: 10.1093/nar/gkh061.

[36] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Nov. 2019. URL: https://arxiv.org/abs/1908.10084.

[37] Michael Schlichtkrull et al. "Modeling relational data with graph convolutional networks". In: *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*. Springer. 2018, pp. 593–607.

[38] Marta Contreiras Silva, Daniel Faria, and Catia Pesquita. "Matching Multiple Ontologies To Build A Knowledge Graph For Personalized Medicine". In: *The Semantic Web: 19th International Conference, ESWC 2022, Hersonissos, Crete, Greece, May 29 – June 2, 2022, Proceedings*. Hersonissos, Greece: Springer-Verlag, 2022, pp. 461–477. ISBN: 978-3-031-06980-2. DOI: 10.1007/978-3-031-06981-9_27. URL: https://doi.org/10.1007/978-3-031-06981-9_27.

[39] Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart. "PARIS: Probabilistic Alignment of Relations, Instances, and Schema". In: *Proc. VLDB Endow.* 5.3 (2011), pp. 157–168. DOI: 10.14778/2078331.2078332. URL: http://www.vldb.org/pvldb/vol5/p157%5C_fabianmsuchanek%5C_vldb2012.pdf.

[40] Zequn Sun, Wei Hu, and Chengkai Li. *Cross-lingual Entity Alignment via Joint Attribute-Preserving Embedding*. 2017. DOI: 10.48550/ARXIV.1708.05045. URL: https://arxiv.org/abs/1708.05045.

[41] Zequn Sun et al. "A Benchmarking Study of Embedding-based Entity Alignment for Knowledge Graphs". In: *Proc. VLDB Endow.* 13.11 (2020), pp. 2326–2340. URL: http://www.vldb.org/pvldb/vol13/p2326-sun.pdf.

[42] Zequn Sun et al. "Bootstrapping Entity Alignment with Knowledge Graph Embedding". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, July 2018, pp. 4396–4402. DOI: 10.24963/ijcai.2018/611. URL: https://doi.org/10.24963/ijcai.2018/611.

[43] Zequn Sun et al. "Knowledge Graph Alignment Network with Gated Multi-Hop Neighborhood Aggregation". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.01 (Apr. 2020), pp. 222–229. DOI: 10.1609/aaai.v34i01.5354. URL: https://ojs.aaai.org/index.php/AAAI/article/view/5354.

[44] Zequn Sun et al. *Knowledge Graph Alignment Network with Gated Multi-hop Neighborhood Aggregation*. 2019. DOI: 10.48550/ARXIV.1911.08936. URL: https://arxiv.org/abs/1911.08936.

[45] Zequn Sun et al. *TransEdge: Translating Relation-contextualized Embeddings for Knowledge Graphs*. 2020. DOI: 10.48550/ARXIV.2004.13579. URL: https://arxiv.org/abs/2004.13579.

[46] Mujeen Sung et al. "Biomedical Entity Representations with Synonym Marginalization". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 3641–3650. DOI: 10.18653/v1/2020.acl-main.335. URL: https://aclanthology.org/2020.acl-main.335.

[47] Xiaobin Tang et al. "BERT-INT:A BERT-based Interaction Model For Knowledge Graph Alignment". In: July 2020, pp. 3146–3152. DOI: 10.24963/ijcai.2020/435.

[48] Elena Tutubalina et al. "Medical concept normalization in social media posts with recurrent neural networks". In: *Journal of Biomedical Informatics* 84 (2018), pp. 93–102. ISSN: 1532-0464. DOI: https://doi.org/10.1016/j.jbi.2018.06.006. URL: https://www.sciencedirect.com/science/article/pii/S1532046418301126.

[49] Xun Wang et al. "Multi-Similarity Loss With General Pair Weighting for Deep Metric Learning". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 5017–5025. DOI: 10.1109/CVPR.2019.00516.

[50] Zhichun Wang et al. "Cross-lingual Knowledge Graph Alignment via Graph Convolutional Networks". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 349–357. DOI: 10.18653/v1/D18-1032. URL: https://aclanthology.org/D18-1032.

[51] Yuejia Xiang et al. *OntoEA: Ontology-guided Entity Alignment via Joint Knowledge Graph Embedding*. 2021. DOI: 10.48550/ARXIV.2105.07688. URL: https://arxiv.org/abs/2105.07688.

[52] Ruobing Xie et al. "Representation learning of knowledge graphs with entity descriptions". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. 1. 2016.

[53] Bishan Yang et al. "Embedding entities and relations for learning and inference in knowledge bases". In: *arXiv preprint arXiv:1412.6575* (2014).

[54] Zheng Yuan et al. *CODER: Knowledge infused cross-lingual medical term embedding for term normalization*. 2020. DOI: 10.48550/ARXIV.2011.02947. URL: https://arxiv.org/abs/2011.02947.

[55] Kaisheng Zeng et al. "A comprehensive survey of entity alignment for knowledge graphs". In: *AI Open* 2 (2021), pp. 1–13. ISSN: 2666-6510. DOI: https://doi.org/10.1016/j.aiopen.2021.02.002. URL: https://www.sciencedirect.com/science/article/pii/S2666651021000036.

[56] Ziheng Zhang et al. "An Industry Evaluation of Embedding-based Entity Alignment". In: *Proceedings of the 28th International Conference on Computational Linguistics: Industry Track*. Online: International Committee on Computational Linguistics, Dec. 2020, pp. 179–189. DOI: 10.18653/v1/2020.coling-industry.17. URL: https://aclanthology.org/2020.coling-industry.17.

[57] Xiaofei Zhou et al. "Learning Knowledge Embeddings by Combining Limit-Based Scoring Loss". In: CIKM '17. Singapore, Singapore: Association for Computing Machinery, 2017, pp. 1009–1018. ISBN: 9781450349185. DOI: 10.1145/3132847.3132939. URL: https://doi.org/10.1145/3132847.3132939.

[58] Maryam Zolnoori et al. "The PsyTAR dataset: From patients generated narratives to a corpus of adverse drug events and effectiveness of psychiatric medications". In: *Data in Brief* 24 (2019), p. 103838. ISSN: 2352-3409. DOI: https://doi.org/10.1016/j.dib.2019.103838. URL: https://www.sciencedirect.com/science/article/pii/S2352340919301891.

# Credits

I would like to express my gratitude to my family for their unwavering support throughout my journey, as well as to my girlfriend, who has been a constant source of inspiration and motivation for my work. I am also grateful to my friends, who have become like a second family to me, and to Professor Paolo Torroni, whose course sparked my interest in this field. I am deeply thankful to Graduate Doctor Vieri Emiliani, who has been my mentor during my internship and a true role model in this field. Lastly, I would like to thank all my colleagues from my degree program, as we have all contributed in our own way to help each other. The international NLP community and their online forums have also been instrumental in fostering research and innovation, and I am grateful to all of them. From the bottom of my heart, thank you.

Lorenzo Niccolai