UNIVERSITÄT
LEIPZIG

Institute of Computer Science
Faculty of Mathematics and Computer Science
Data Science, M.Sc.

Ontology Matching using Graph Neural Networks

Master's Thesis

submitted by:
Jerome Markus Louis Würf

Matriculation Number :
3710433

Referee:
Prof. Dr. Erhard Rahm
Dr. Victor Christen

Abstract

The biomedical domain requires experts and information systems to have a common understanding of terms and concepts used during the exchange of information. Ontologies are a common way to describe these terms and concepts. Research within the biomedical domain constantly pushes the boundaries of knowledge and thus changes the semantics of concepts or enriches ontologies with new concepts. This dynamic environment requires a constant curation of existing ontologies. Ontology matching enables an automation of this curation, allowing different ontologies to integrate. In recent years, the field of ontology matching has seen a rise in machine learning-based matching systems. Systems exploiting the textual representations of concepts within the ontologies dominate this new trend, motivating us to propose **O**ntology **M**atching **u**sing Graph Neural **Net**works (OMUNET) a system that enriches textual representations with graph structural information. Our system models the ontology matching problem as a relation prediction between two graphs. Additionally, we investigate techniques for negative node sampling during the network training. We evaluate our system on the 2023 version of the Bio-ML track of the Ontology Alignment Evaluation Initiative. Our results show that our proposed system cannot outperform existing systems and that one of our negative node sampling strategies exploiting the graph structure of an ontology is only performing slightly better than a baseline.

Jerome Markus Louis Würf
3710433

Contents

List of Abbreviations

AML Agreement Maker Lite

BERT Bidirectional Encoder Representations from Transformers

CNN Convolutional Neural Network

DL Description Logic

DOID Human Disease Ontology

FMA Foundational Model of Anatomy

FNN Forward Nerual Network

GAT Graph Attention Network

GCN Graph Convolutional Network

GNN Graph Nerual Network

Graph SAGE Graph Sample and Aggregate

HAN Hyperbolic Attention

LSTM Long Short-Term Memory Network

MEDTO Medical Data to Ontology Matching

ML Machine Learning

Mondo Mondo Disease Ontology

MRR Mean Reciprocal Rank

NCIT National Cancer Institute Thesaurus

OAEI Ontology Alignment Evaluation Initiative

OM Ontology Matching

OMIM Online Mendelian Inheritance in Man

OMUNET Ontology Matching using Graph Neural Networks

ORDO Orphanet Rare Disease Ontology

OWL Web Ontology Language

UMLS Unified Medical Language System

SNOMED Systematized Nomenclature of Medicine and Clinical Terms

RNN Recurrent Neural Network

List of Figures

List of Tables

1. Introduction

Ontologies serve as formal descriptions of knowledge. In the context of the biomedical domain, these descriptions provide a common understanding of terms within the domain and enable the further integration of data and knowledge. In recent years multiple ontologies have been developed for the biomedical domain. The Foundational Model of Anatomy (FMA) [1] or the Systematized Nomenclature of Medicine and Clinical Terms (SNOMED-CT) [2] are just two examples of the many ontologies that have been developed and are widely used in applications like electronic health records to improve the sharing and retrieval of a patient's diagnosis or the interoperability of health care systems [3].

The biomedical domain is constantly evolving. Beyond others, the advancements are driven by scientific and medical research accumulating new knowledge, standardization efforts to allow further integration of existing information systems, and changes determined by policymakers. This dynamic character of the domain needs to be reflected with a constant adoption and enrichment of existing ontologies, which would otherwise become outdated and less useful as terms might become ambiguous.

Due to their size and complexity, the manual maintenance of existing ontologies would be an error-prone and labor-intensiv. The technique of Ontology Matching (OM) alleviates this burden and allows for the automatic alignment of ontologies, ensuring semantic interoperability between their respective concepts. The ultimate goal of OM is to find correspondences that serve as a base for the integration of either the ontologies themselves or other data sources that can be described by the ontologies [4].

1.1. Motivation

OM is a well-researched field, and multiple approaches have been developed to solve the problem. The approaches can be categorized into different classes, like structural-based and element-based matching [4]. Ultimately, the choice of the matching approach depends on the characteristics of the ontologies to be matched and the requirements of the matching task. Using Machine Learning (ML) based techniques gained traction in recent years. Flued by rapid advances in the computational power provided by graphical processing units, deep learning techniques in computer vision and natural language processing became increasingly sophisticated. This development brings the potential to improve the performance and flexibility of OM systems.

Recently, OM systems using pre-trained language models like Bidirectional Encoder Representations from Transformers (BERT) [5] emerged as prominent candidates challenging the traditional approaches. A majority of matching systems participating at the Bio-ML track of the Ontology Alignment Evaluation Initiative (OAEI) [6] only uses the textual description of concepts. They use the descriptions to generate a representation of the respective concepts. Systems using only the textual descriptions for a matching process are the primary motivation for this work. Inspired by existing research on the topic [7, 8], we further want to investigate the potential of enriching

text-based representations with graph structural data using Graph Neural Networks (GNNs) and thus propose Ontology Matching using Graph Neural Networks (OMUNET). For an improved reproduction of our results, the source code of OMUNET is published on Zenodo[1]. Using OMUNET, we aim to answer the following research questions.

1.2. Research Questions

RQ1 Can the application of a Graph Neural Network to the equivalence matching Bio-ML Tasks of the Ontology Alignment Evaluation Initiative perform higher global and local matching scores than the existing methods?

RQ2 Which negative sampling method of uniform-, structural- or similarity-based sampling has the highest positive influence on the matching score of the implemented Graph Neural Network based matching prediction?

1.3. Structure of this Work

In order to provide a better understanding of the concepts employed in our presented approach, chapter 2 introduces ontologies, OM and GNNs. In chapter 3 we take a look at the OM systems that are most relevant for this work. With the necessary background knowledge, chapter 4 presents the structure of our proposed method: OMUNET. The chapter structures around the phases of our OMUNET framework. Following our proposed method, we take a descriptive look at the experimental results in chapter 5, differentiating between model-specific and full evaluation results. Finally, we discuss the results in the context of our above-mentioned research questions in chapter 6 and conclude our work in chapter 7.

[1] https://zenodo.org/doi/10.5281/zenodo.10659449

2. Background

This second chapter provides the conceptual foundations for understanding our methodology. The first section introduces ontologies together with their theoretic background and applications. Further, this section discusses the problem of OM and well-known matching systems. Next, we present the basics of transformer models (section 2.2), followed by an extensive discussion of the fundamentals of GNNs (section 2.3). The chapter concludes with a differentiation of this work in section 2.4.

2.1. Ontologies

The word ontology comes with multiple different interpretations. In the philosophical case, for example, an ontology is seen as observing how things are connected in our reality. In this sense, the Oxford Dictionary defines an ontology as: "The science or study of being; that branch of metaphysics concerned with the nature or essence of being or existence" [9]. This is not only limited to the observable reality around us but also captures the structure of imaginary things. Another interpretation, the one relevant to this thesis, sees an ontology as some structured information that resides in the storage of an information system. More specifically, an ontology in computer science is an approach to express the topology of a system within a particular domain, containing entities and relations between them [10]. The following chapter will define ontologies, provide preliminary insights on how ontologies can be represented, deal with issues around the heterogeneity between ontologies, and how OM resolves this. Finally, we present existing approaches to OM.

2.1.1. Introduction to Ontologies

This subsection introduces ontologies from a model-theoretic perspective. To get an initial intuition about ontologies, we consider the domain of a hospital. In a hospital, several persons are involved. In a hospital management tool, every person has a unique ID. A person can be a doctor, a nurse, or a patient. Those are concepts. Within the hospital, there are multiple individuals. Each individual is mapped one of the concepts via an unary mapping.

Additionally, concepts can interact, forming binary relations—a nurse, for example *reports to* a doctor. Doctors *cooperate with* each other. Doctors *treat* their patients. The concrete mapping of individuals and relations between them depends on interpreting the meaning of a concept like a doctor or a nurse. A concrete instance of an ontology always captures a specific state of this mapping, which can evolve. To give a rather extreme example: The nurses start a revolution. Now doctors report to nurses, which would change the semantics of this binary relation. The perception of reality changes, which must also lead to a change in ontology. Following [10], the model-theoretic definition of an ontology requires three central properties.

Conceptualization

A conceptualization is an intensional relational structure defined as a triplet $\mathbf{C} = (D, W, \mathfrak{R})$. Here D is a set of existing individuals called *discourse*. In the example above, this would be an ordinary person with only an ID but no associated label. W is a set of different *world* states. A *world* state is the set of mappings from a semantic meaning to a concept or to a relation between concepts. In a world where the above-mentioned binary relation *reports to* is directed from doctor to nurse differs from a world where the direction of the relation is inverted. $\mathfrak{R}$ is a set of conceptual relations on D and W. A single conceptual realization is the mapping from a specific world to the unary, binary, and n-ary relations of and between individuals within D. The intensional relational structure presents the knowledge of the domain.

Specifications of a conceptualization

A key aspect of ontologies is that users agree on a specific conceptualization. To facilitate this property, the relations specified by a conceptualization need to be described formally, machine-readably, using a language $\mathbf{L}$. A language has an associated vocabulary V. Together with a conceptualization $\mathbf{C}$ an ontological commitment $\mathbf{K} = (\mathbf{C}, I)$ is created. I is an interpretation function that maps an element of the vocabulary V either to an element of the discourse D or to an element from the set of conceptual relations $\mathfrak{R}$. Such an ontological commitment also creates another interpretation from a word in V towards the extensional space, e.g., the individual of a concrete world, creating an ontological model. The extensional space provides knowledge about a domain for a particular world.

In order to capture the correct ontology model and remove ambiguity, the interpretation of a language is supported by additional axioms described via the language $\mathbf{L}$ within the commitment $\mathbf{K}$. For example, in the hospital mentioned above domain, an axiom augmenting the binary relation of *reports to* can be specified as an asymmetric directed relation and *cooperates with* as a symmetric relation. An additional axiom within this example could be taxonomic information. A doctor, nurse, and patient are sub-concepts of a person, and vice versa; a person is a super-concept of the other three.

Languages used for the specification of a conceptualization can vary on their level of formality [11]. On the one end, there are informal languages, like a set of terms or thesauri. This kind of language does not provide enough meaning to remove ambiguity. More formal languages allow for the desirable trait of automatic reasoning within an ontology. Automatic reasoning is desirable for querying knowledge and integrating data. The most formal languages, like first-order or higher-order logic, are very expressive, but their rigidity hinders the efficiency of a reasoner. A reasoner is an algorithm that performs logical inferences or deductions to conclude. Thus, the most widely used languages are based on description logic and its most prominent representative, the Web Ontology Language (OWL) [12].

Shared ontological commitments

A reuse of an ontology between communities requires a shared ontological commitment. In practice, a perfect capture of a conceptualization within a domain of the natural world is not feasible. This is due to knowledge gaps or an erroneous perception. Additionally, an ontological commitment can

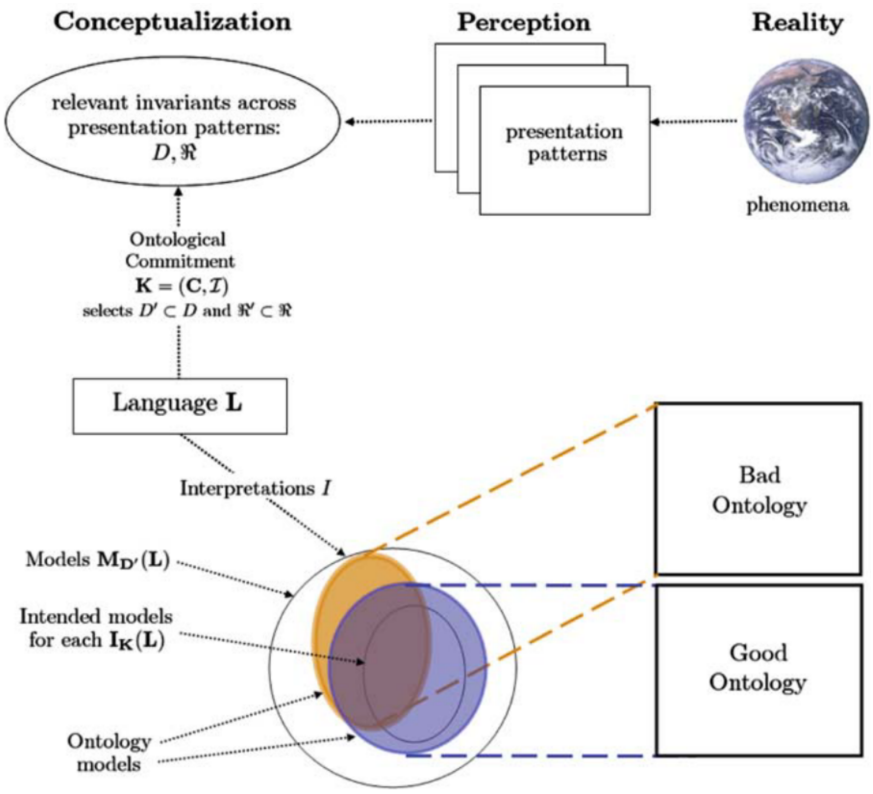

Figure 2.1.: A visualization of the model-theoretic properties of an ontology. The visualization originates from [10]

be misunderstood between parties because the primitives of a language, used in the axioms of concepts and relations, require a mutual agreement. To give an example, the relationship "cooperates with" is symmetric. Both parties, using a commitment with such an axiom, must have the same understanding of the term symmetric.

To summarize, we humans perceive reality and create a conceptualization from this understanding via our perception. To share this conceptualization with other humans or to bring this conceptualization into the realm of information systems, we are using a language, creating an ontological commitment. The language itself is subjected to multiple interpretation perspectives. First, there is an interpretation of the terms of the language towards the conceptualization, and second, there is an interpretation of the terms to the world's individuals. The expressiveness of a language is crucial for the interpretation to capture the intended model. Thus, the following section looks at the representation of ontologies.

2.1.2. Representation of Ontologies using the Web Ontology Language

The previous subsection dealt with conceptualization, ontologies, and languages on a high abstraction layer. This subsection provides an intuition to describe languages and its most prominent application, OWL.

The following contains the essential properties of description logic; for a more rigorous introduction, see [13]. Description Logic (DL) is a fragment of decidable first-order predicate logic. DL represents knowledge by modeling and reasoning about concepts, relations, and constraints. The syntax of DL

is split into two categories, TBox and ABox. TBox contains the termini of the language holding the intensional knowledge. An ABox holds the extensional, or assertional, knowledge associated with specific individuals of a concept of the TBox. For this thesis, the properties of the TBox are more critical. Concepts of the TBox relate to each other in a taxonomic hierarchical structure.

To illustrate, a TBox holds facts like: "A doctor is a subconcept of a person." This binary relationship between concepts is called subsumption. The concept of a doctor is more specific than a person and vice versa; the person is more general than the doctor. Using the outlined subsumption, a doctor inherits all the properties of a person and adds their own. An important property of the declaration of the concept definitions within a TBox is that those must be acyclic; otherwise, statements become undecidable. The decidability of a description language is an important feature, allowing a reasoner to falsify or verify a specific statement. The DL that is utilized by current version of OWL, OWL2, is $\mathcal{SROIQ}$ [14]. $\mathcal{SROIQ}$ is based on $\mathcal{SHOIN}$ [15] which itself is an extension of the most basic DL, $\mathcal{ALC}$ [16]. $\mathcal{ALC}$ means *attributive language with complements*. An *attributive language* allows operations like negations of atomic concepts, the intersection of concepts, restrictions, and existential quantification. Atomic concepts are concepts of a TBox that can not be specific by other concepts. *With complements* means that the DL allows the negation of concepts. The DLs based on $\mathcal{ALC}$ add features like inverses for binary relations or cardinality constraints between individuals to allow for more complicated knowledge modeling.

An important component in OWL is the declaration of entities. There are *Individual*, *Class* and *ObjectProperty*. To come back to the example of the previous section. The set of *Individual* corresponds to a set that would hold the IDs of the persons in a hospital. The set of *Class* corresponds to a set holding the concept of doctor, nurse, or patient, and *ObjectProperty* corresponds to binary relations like "cooperate with" and "reports to". This builds the vocabulary of an ontology. Another important feature of OWL is expressions on the declared entities. *inverse "reports to"* would reverse the semantics of the reporting direction from doctors to nurses. The expression of object property restrictions involves *some* and *only* and are applied between an *ObjectProperty* and a *Class*. *"cooperates with" only doctor* would restrict the cooperation in a hospital to only occur between doctors. Many more class expressions exist which are not directly relevant to this thesis. Class axioms used to model the taxonomic structure of the *Class* entities within the TBox are more important to this thesis. The central axiom is *SubClassOf*, which defines the above-described subsumption of concepts within a DL. This is only a brief introduction to the OWL syntax; refer to [12] for a full specification.

2.1.3. Sources of Heterogeneity between Ontologies

Ontologies are developed by knowledge engineers with different goals, perceptions of reality, and knowledge about available tools to facilitate the engineering process. This creates heterogeneity between ontologies that should model the same or overlapping domains. Heterogeneity comes in four different flavors [4].

- **Syntactic heterogeneity**: Two ontologies are described in different languages. E.e: One is in OWL and the other one in the Resource Description Framework Schema.

- **Terminological heterogeneity**: This happens when concepts of the same semantic have differing terms in the vocabulary. For example, one ontology refers to doctors, and the other to physicians.
- **Conceptual heterogeneity**: The same domain uses different axioms. For instance, identity might be based on legal identity, verified by documents, in one ontology, and on social identity, confirmed by surrounding individuals, in another.
- **Semiotic heterogeneity**: This is the kind of heterogeneity that happens due to different interpretations of the conceptualization or the terms of a language used to describe a conceptualization. Refer to the first subsection of this section for further details. Semiotic heterogeneity can not be resolved algorithmically, as these errors already happen when humans associate meaning within an information system's representation.

To resolve these heterogeneities, ontologies must be integrated. An integration of ontologies is enabled by alignments, which are generated by the process of OM.

2.1.4. Ontology Matching

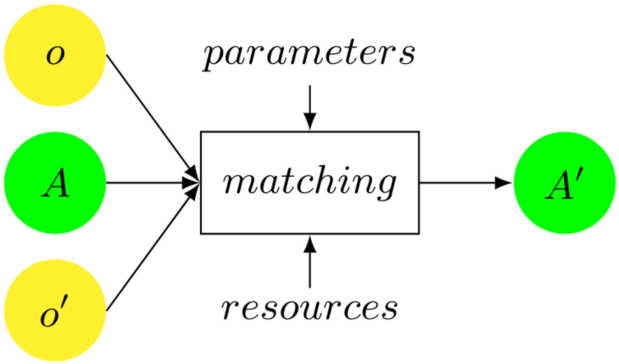

Figure 2.2.: A matching process between two ontologies using a reference alignment and creating an output alignment. The visualization originates from [4]

The goal of a matching process between an ontology o and a second ontology o' is the creation of an alignment A'. An alignment is a set of correspondences between the concepts or relations of two ontologies; see Figure 2.2. A correspondence is a connection between an entity of o and an entity of o' for entities with the same semantic meaning. Matchable entities between two ontologies can be a *Class*, an *Individual*, or an *ObjectProperty*. After the matching process, each correspondence has a score expressing the semantic similarity's confidence degree. A Matching process uses reference alignments A, parameters p like thresholds for the scores, and external resources r, for example, external thesauri, to support the matching process. These variables create the input for a matching function f [4]; see equation 2.1.

$$A' = f(o, o', A, p, r) \tag{2.1}$$

The meaning of a correspondence in an alignment depends on the task of the actual matching process. A task of a matching process can be to detect a relationship r of either equivalence (=),

disjointedness ($\perp$) or a subsumption ($\leq$) between an entity e of ontology o and an entity e' of an ontology o'. A correspondence is thus a triple [4], see Figure 2.2.

$$\langle e, e', r \rangle \tag{2.2}$$

Additionally, each correspondence within an alignment has optional metadata like the already mentioned score or a unique identifier of the correspondence. Most of the time, the score expresses confidence in the interval of $[0, 1]$. A one-to-one matching between entities based on equivalence is conducted in the primary case of matching two ontologies. However, the task can also be more complex; for example, it could aim for the creation of an alignment between more than two ontologies, or one entity of o could build a complex correspondence towards multiple entities of o'. This work only deals with the primary case of one-to-one equivalence correspondences.

On a higher level, conducting a matching process is embedded in a matching workflow [4]. The workflow contains further elements like the selection of ontologies, existing alignments and matchers, the matching itself, and the evaluation of the alignments. The process of selecting matchers, the matching itself, and the evaluation is conducted iteratively to refine the performance of the alignment. An enhancement can be achieved by methods like trimming, consistency checking or the modification of matcher parameters. Trimming in this context means the modification of the thresholds of the confidence scores that ultimately determine the existence or absence of a match. The realization of the workflow depends on the final application where the matching should be conducted. Necessary design requirements for the matching are considerations like: Should the matching be executed on a level of elements or only on the structural level of classes? Can an alignment be produced offline and later applied at the application's run time? Can the application tolerate false positives or false negatives? This work predominantly focuses on the proposal of a new matching process and the evaluation of it.

Following [4], a matching process can be categorized, among others, according to the technique that processes the information provided by the ontologies that should be matched. Conceptually, there are element-level and structural-level techniques (another commonly encountered differentiation is the one between schema and instance-based techniques [17]). Element-level techniques consider the entities to be matched, detached from the connections to other entities. Structural-level techniques center around the opposite, incorporating the relationships between entities.

Structural-level techniques include:

- String-based techniques
- Language-based techniques
- Constraint-based techniques
- Informal Resource-based techniques
- Formal Resource-based techniques

Element-level techniques include:

- Graph-based techniques
- Taxonomy-based techniques
- Model-based techniques
- Instance-based techniques

For a complete specification of all those techniques, see [4]. We focus on the two relevant for this work: language-based techniques and taxonomy-based techniques. Language-based techniques consider approaches that leverage string-based techniques but go beyond the raw comparison of strings by analyzing the construction of a sentence. On the one hand, it includes normalizing text by tokenization, lemmatization, or stopword elimination to analyze the grammatical structure to extract a semantic meaning. On the other hand, it includes using external resources, for example, to look up synonyms, terms with the same meaning or hyponyms, and terms with more general meaning via a Thesaurus to improve the matching capabilities. Taxonomy-based methods are a specialization of graph-based techniques. Graph-based techniques see the input ontologies as labeled graphs. When matching entities with each other, the entities are treated as nodes, and their similarity is evaluated by considering their neighborhood of connecting entities. The taxonomy-based methods restrict this approach by only considering the relations between *Class* entities that define the taxonomic structure of an ontology, e.g., the *subClassOf* relationship in OWL.

The evaluation of a matching process uses its final alignment. The evaluation depends on the goal of the overarching matching process. A competence benchmark allows the improvement of a single system, a comparative evaluation aims at the discrimination between multiple systems on a shared task, and an application-specific evaluation targets a workflow tailored toward an application's requirements. The quality of a matching process is determined by comparing the final alignment to a set of reference alignments. In comparative evaluation settings, the input ontologies are often provided by organizers of matching tasks. For example, the OAEI provides multiple tracks for matching tasks with different settings. Traditionally, evaluation relies on reporting precision, recall, and F1-score measurements between the final alignment of a matching process and a reference alignment. We present evaluation techniques specific to our work in chapter 5.

2.1.5. Traditional Systems for Ontology Matching

The long organization of the tracks of the OAEI (since 2004) has spawned an ever-growing population of OM systems. The following subsection introduces two traditional matching systems, a structural-level one and one that mixes structural- and element-level. Non-traditional, ML-based systems like BERTMap [18] or BioHyperbolic Attention (HAN) [8] are reviewed in greater detail in the related work (chapter 3).

Agreement Maker Lite (AML) [19] is a structural-level matching system developed explicitly for matching large biomedical ontologies. It is based on AgreeMaker [20]. On a high-level view, the system has three key steps: loading, matching, and filtering. The system extracts structural and lexical data at the loading step, which the matching process uses both. A follow-up repair also uses the structural data after the initial matching. In addition to the provided input ontologies, the

matching utilizes a background ontology. Six different matchers use this data to create scores for an initial alignment. Following the matching phase, the initial alignments are passed to a selector filtering alignments with conflicting cardinalities, and then alignments causing logical conflicts are removed in a final repair step.

LogMap [21] mixes structural-level and element-level matching. It combines matching approaches revolving around lexical and structural matching and adds a logic-based repair step. LogMap computes indices for the input ontologies based on lexical and structural features. Then, it builds the intersection between these indices to create initial anchor mappings. With the initial mappings, an iterative refinement alternating a mapping extension and a mapping repair step is performed. The mapping extension is based on an ontology-specific similarity measurement [22] and searches for additional potential matches by searching the class hierarchy. The mapping repair step is based on reasoning, removing mappings that would introduce conflicts. This output builds the final alignment, which is additionally improved by a user's assessment, potentially identifying missing mappings.

This section first introduced the concept of ontologies from a model-theoretic background. Then, we investigated the representation of ontologies using the description language OWL. Even if two ontologies of the same domain deal with semantically similar objects, there can be heterogeneities between ontologies. We discover these sources of heterogeneities and provide an introduction to OM. Finally, we presented traditional OM systems that cleared the ground for the uprising usage of ML-based matching approaches.

2.2. Transformer Models and Word Embeddings

The previous section presented two state-of-the-art OM systems. Both share the characteristic that they rely on transformer models to either compute a synonym score (BERTMap) or to provide an initial embedding (BioHAN). Like BioHAN, the system presented in this work also relies on a language model to obtain an initial embedding. The following briefly introduces transformer models and generation of word embeddings based on the textbook of Jurafsky et al. [23].

Language models can be created by different architectures like classical Forward Nerual Network (FNN) [24] or Recurrent Neural Network (RNN) [25]. The underlying assumption that language models exploit is the distributional hypothesis. The hypothesis intuitively states that similar contexts surround words with semantically equivalent meanings. Language models are pre-trained on vast amounts of text or corpus in a semi-supervised fashion, allowing them to encode the semantic meaning of a word into a vector representation of it. Language models using classical architectures like FNN or RNN generate static embeddings, resulting in a hard-coded mapping from a word type to a vector representation. Transformer models alleviate this problem by encoding a word's context into its corresponding embedding.

Transformer models were introduced with the groundbreaking work of Vaswani et al. [26], which leverages the principle of self-attention. For a full introduction to self-attention, consult Jurafsky et al. [23]. Intuitively, a self-attention layer learns the importance of an element within a sequence regarding the output computation of the previous elements. A transformer model contains multiple transformer blocks. One transformer block itself contains a SelfAttention layer, residual connection

[27] (denoted as +), a FNN layer and LayerNorms, see equation 2.3. The LayerNorm standardizes its input vectors.

$$\begin{aligned} \mathbf{z} &= \text{ LayerNorm } (\mathbf{x} + \text{SelfAttention}(\mathbf{x})) \\ \mathbf{y} &= \text{LayerNorm}(\mathbf{z} + \text{FFN}(\mathbf{z})) \end{aligned} \tag{2.3}$$

The train process of a language model using transformer blocks conducts an autoregressive generation. Autoregressive generations predict the next word based on the conditional prediction of the previous words. The next word of the sentence is known from the corpus, allowing a cross-entropy loss to be computed for each input word. Finally, the cross-entropies are averaged and can be used for the optimization step of the network.

Recently, a new architecture paradigm has evolved, allowing the generating of robust contextualized embeddings. In contrast to static embeddings, contextualized embeddings represent their associated words within their surrounding context. This is achieved by internally representing a word using multiple vectors, one for every context it appears in. The significance of contextualized embeddings seems to be particularly evident in the domain of ontology matching. According to Jurafsky, they enable the evaluation of the semantic similarity of words based on their contextual usage [23].

Classical transformer models only utilize the sequence preceding a word. However, learning contextualization by respecting both sides of a sentence for a word to predict generates embeddings with a superior representational capability.

The first model introducing this approach is BERT. BERT uses masked language modeling, allowing the model to use a word's left and right context to predict it within a sentence. This changes the training objective from "Given the previous sequence of words, predict the next one!" to "Here is a sequence of words with one missing. Predict the missing word!". Besides just masking words, the input sequences of BERT also contain words that are substituted with other words from the training corpus. Specifically, 15% of a single training sequence (sentence) to BERT is used as a gold standard for self-supervision. Of those words, 80% are masked, 10% are permuted, and 10% is not changed at all concerning the input sequence in which the word occurs. Compared to the original transformer models, the averaged cross-entropy loss is not composed of loss values of every word within the input sequence but solely by the sampled gold-standard words for self-supervision. The last layer of BERT, the layer before the loss computation, generates one contextualized word embedding for each word of the input sequence. Often, these embeddings are averaged to obtain an embedding representing the semantics of the input sequence [23].

This section briefly introduced transformer models, their evolution to bidirectional transformers, and the intuition of the difference between static and contextualized word embeddings. For OM systems utilizing GNNs, these word embeddings serve as an initial representation of a node within the input graph. The following section provides an overview of GNNs.

2.3. Graph Neural Networks

Neural network models like Convolutional Neural Networks (CNNs) focus on processing grid-structured inputs, for example, images, whereas RNNs and transformer models focus on sequences like text as inputs. Gird-like data structures can be seen as a specific case of a graph where the elements have a particular order; see Figure 2.3. The aggregation and information propagation process in GNNs resembles the receptive field concept in CNNs. Both mechanisms try to capture and integrate relevant information from the surroundings to enrich the understanding and representation of the target node or region.

GNNs, as the name suggests, takes a graph as an input and potentially generates a representation for each vertex. A graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, holding a set of vertices $v_i \in \mathcal{V}$, from now on referred to as nodes, and a set of edges $\mathcal{E}$ containing tuples of vertices $e_{i,j} = (v_i, v_j) \in \mathcal{E}$. According to the taxonomy of [28], there are four types of GNNs. Recurrent GNNs, Graph Convolutional Networks (GCNs), graph autoencoders, and spatial-temporal GNNs. This introduction is mostly based on the work of [29] and mainly focuses on GCNs; thus, in the following, the abbreviation GNN is used with semantical equivalence to GCN.

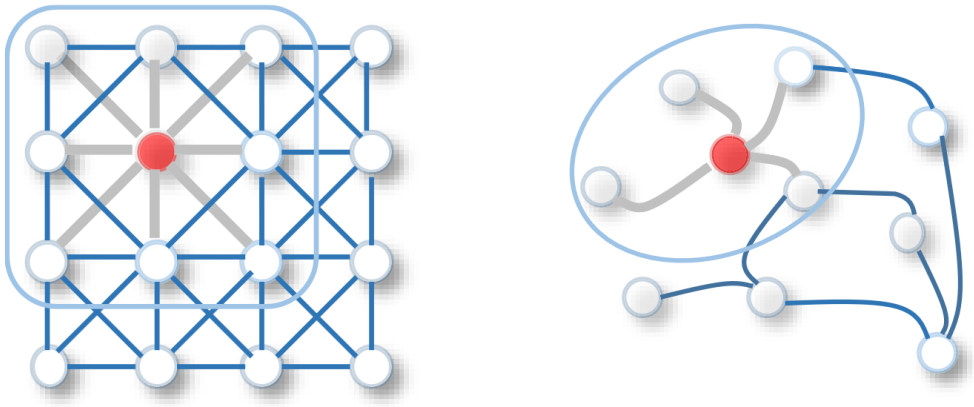

Figure 2.3.: The difference between convolutions in CNNs and GNNs. The left illustrates a convolution for CNNs. The data is grid-structured. During a convolution, a filter of a specific size moves step by step over the grid. The right illustrates a convolution of a single GNN layer for graph-structured input. The notion of a filter does not exist. A neighborhood relation to a node determines the elements to be included in a convolution. The illustration originates from [28].

2.3.1. Message Passing

A GNN comprises multiple layers, each refining a node's representation by including information on more distant neighbors. At a single layer, a GNN aggregates the information for each neighboring node into the representation of the respective node itself (Figure 2.4). The aggregated information passed from the neighbors to a node can be interpreted as a message.

Through repeated iterations, node representations encapsulate information from increasingly distant neighborhoods, progressing from 1-hop to 2-hop and so forth. The embedded information

within these representations manifests in two distinct flavors: structural information and feature information. Structural information denotes the connectivity patterns of nodes within the graph, potentially revealing structural motifs such as cycles or other topological features. On the other hand, feature information encompasses the information associated with the initial features of each node, reflecting specific characteristics or attributes.

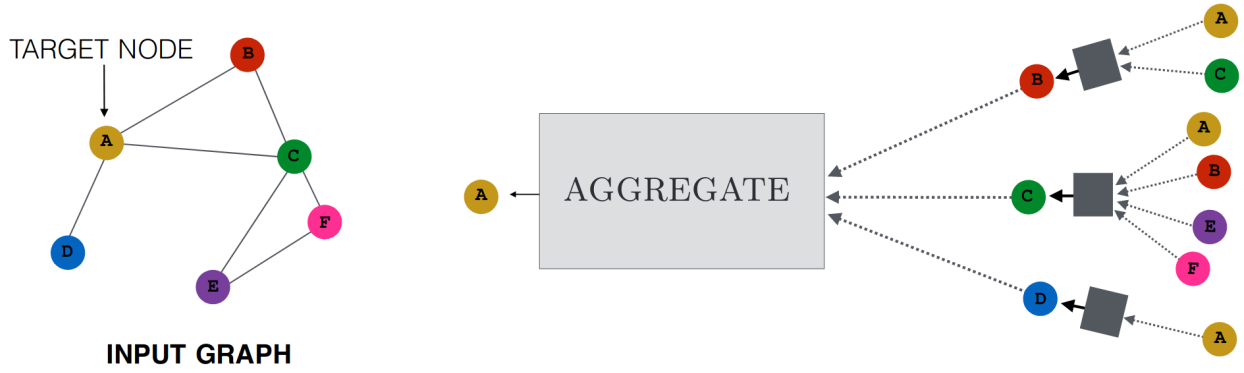

Figure 2.4.: Visualization of a two-layer GNN and its message passing. On the left is a color-coded input graph to the GNN. On the right is the computational graph of a GNN layer for the yellow target node of A. The first layer aggregates the information of the direct neighbors (B,C,D). The second layer also includes the information of the neighbors of the direct neighbors. This illustration originates from [30]

Formally, a GNN has l layers or iterations. The current representation of a node u is presented by $\mathbf{h}_u^{(k)}$. The representation of the next layer is computed by:

$$\begin{aligned}\mathbf{h}_u^{(k+1)} &= \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \text{AGGREGATE}^{(k)}\left(\left\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\right\}\right)\right) \\ &= \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right),\end{aligned} \tag{2.4}$$

The functions of UPDATE and AGGREGATE differ depending on the specific GNN architecture, which is the topic of the next section. In general, the UPDATE step combines the representation and the message of the given iteration. $\mathcal{N}(u)$ denotes the set of neighboring nodes of a node u. From the equation, one can see that the message $\mathbf{m}_{\mathcal{N}(u)}^{(k)}$ is the aggregated representation of the neighborhood, computed by the AGGREGATE operation. $\mathbf{h}_u^{(0)}$ is the initial node embedding. In the case of this work, this will be the textual embedding derived from a BERT model.

2.3.2. GNN Architectures

The previous subsection offered an introductory overview of the message passing paradigm of GCNs and identified two critical steps, UPDATE and AGGREGATE, in a single layer of a GNN. The following delves deeper into the architectures used in this study. These architectures are distinguished by their unique implementations of those mentioned above UPDATE and AGGREGATE processes.

Graph convolution networks

The GCN, first introduced by Kipf and Welling [31], uniquely handles the node update process. Unlike other networks with a distinct update step, GCNs incorporates the node's data directly in the aggregation stage, as seen in equation 2.5. This means the aggregation step also implicitly updates the node. However, a limitation arises since it is hard to differentiate between the node's features (u) and those of its neighbors ($\mathcal{N}(v)$). Aggregation involves summing up symmetric-normalized representations, where this normalization considers the sizes of the node neighborhoods for both the source node u and its neighbor v. The process concludes with a non-linear activation function.

$$\mathbf{h}_u^{(k)} = \sigma\left(\mathbf{W}^{(k)} \sum_{v \in N(u) \cup u} \frac{\mathbf{h}_v^{(k-1)}}{\sqrt{|N(u)||N(v)|}}\right) \tag{2.5}$$

Graph SAGE

With Graph Sample and Aggregate (Graph SAGE) [32] first introduced the abstract notion of an aggregation function and presented an inductive framework to generate node embeddings for unseen data samples. This paper presents three approaches to realize the aggregation function: a mean aggregator, an Long Short-Term Memory Network (LSTM) [33] aggregator, and a pooling-based aggregator. Permutation invariance is a key property of these aggregation functions necessary for GNNs. Permutation invariance means that no matter how the input to these functions, the node neighborhood, is ordered, the same output is produced. The mean aggregator is similar to the GCN model, except having a different normalization (Equation 2.6).

$$\mathbf{h}_u^{(k)} = \sigma\left(\mathbf{W}^{(k)} \sum_{v \in N(u) \cup u} \frac{\mathbf{h}_v^{(k-1)}}{|N(v) \cup u|}\right) \tag{2.6}$$

Additionally to the mean-based aggregator, the authors also present an LSTM and a pooling-based aggregator. The LSTM-based aggregator is a more complex training function, leading to a better expressive capability. The pooling-based aggregator uses a multi-layer perceptron for each representation of neighborhood nodes. For a complete description of these two approaches, refer to [32]. The authors conclude that the LSTM- and pool-based models performed were superior to the mean-based approach. The difference between LSTM- and pool-based models is not statistically relevant. As expected, due to a more complex function, the LSTM-based model has a worse total execution time than the other two approaches.

Graph Attention Networks

Inspired by the recent success of sequence models leveraging self-attention, the authors of GAT [34] introduced this paradigm to GNNs. Unlike previous approaches, Graph Attention Networks (GATs) do not set the weights of the neighborhood nodes during the aggregation to a fixed value but learn them. The following equation (Equation 2.7) shows the general form of the previous GNN architectures. Until now, $a_{v,u}$ was a fixed value determined by the number of nodes in the neighborhood.

$$\mathbf{h}_u^{(k)} = \sigma\left(\mathbf{W}^{(k)} \sum_{v \in N(u) \cup u} a_{u,v} \mathbf{h}_v^{(k-1)}\right) \tag{2.7}$$

In GATs, the weights $a_{u,v}$ are computed as a by-product of an attention mechanism by first generating an attention coefficient $e_{u,v}$ using an attention mechanism α (Equation 2.8) and then normalizing the attention values using a softmax function (Equation 2.9). This allows us to assess the individual importance of each neighborhood node v of a node u.

$$e_{u,v} = \alpha(\mathbf{h}_u, \mathbf{h}_v) \tag{2.8}$$

$$a_{u,v} = \frac{\exp(e_{u,v})}{\sum_{w \in N_u} \exp(e_{u,w})} \tag{2.9}$$

The method of computing the individual weights for the aggregation step is agnostic to the specific attention mechanism. The authors of GAT use a single-layer FNN as an attention strategy. An emphasis on using multi-head attention as a regularization mechanism stabilizes the learning. Multi-head attention is realized by performing the described steps k times using attention strategies with mutually exclusive sets of parameters and applying the gained weights to compute multiple representations of a single node. Those representations are aggregated by summation or concatenation. In their evaluation, the authors report that their proposed GAT model performs state-of-the-art results on four benchmarks containing datasets of citation networks and protein to protein interaction.

The preceding sub-section has outlined the mechanisms of message passing in GNNs, particularly emphasizing the UPDATE and AGGREGATE steps. We explored various architectures, including GCNs, which integrates a node's features in the aggregation step, and Graph SAGE, which introduces an inductive framework with a permutation invariant aggregation function. Finally, we looked at GATs, which leverages self-attention mechanisms to weigh the influence of neighboring nodes, thereby enhancing the model's capacity for feature learning. Each of the presented models can be utilized in different tasks on graph data. Most of the evaluations of the presented models are based on node classification tasks. The task of predicting vertices between nodes is more complicated and relevant to this work.

2.3.3. Relation Prediction

In graph data, tasks generally fall into three broad categories: graph-level prediction, edge-level or relation prediction, and node-level prediction. The architecture of GNNs used for relation prediction is designed to be task-agnostic. Typically, following the forward pass through a GNNs architecture, a task-specific prediction head is applied to compute the outputs for the desired task. In the most basic form for node-level predictions, this involves applying a fully connected layer to transform the node representations into a k-dimensional vector for a k-way classification. In the following, the output of a prediction head is denoted as $\widehat{\boldsymbol{y}}_{u,v}$. For edge-level predictions, two main approaches are commonly employed.

The representations of two nodes are initially obtained from the final GNN layer, denoted as L. These node representations, $\mathbf{h}_u$ and $\mathbf{h}_v$, correspond to the source and target nodes, respectively. For edge-level prediction, two primary methods are employed. The first method involves concatenating the representations of the source and target nodes, followed by applying a fully connected or linear layer to this concatenated vector (2.10). With this first approach, the resulting prediction vector can perform a multi-classification or regression on edge properties.

$$\widehat{\boldsymbol{y}}_{uv} = \text{Linear}\left(\text{Concat}\left(\mathbf{h}_u^{(L)}, \mathbf{h}_v^{(L)}\right)\right) \tag{2.10}$$

The second method computes a scalar value via a dot product of the node representations. This scalar value is particularly useful in binary classification tasks, where the goal is to determine the presence or absence of a relationship between two nodes (Equation 2.11)

$$\widehat{\boldsymbol{y}}_{uv} = (\mathbf{h}_u^{(L)})^T \mathbf{h}_v^{(L)} \tag{2.11}$$

Subsequently, the resulting vector or scalar for each node pair is utilized to compute a loss function. The loss function is an integral part of the learning setup regardless of whether the training setting is supervised or self-supervised. A challenge in relation prediction is the availability of only positive examples. To address this, negative samples are artificially generated by corrupting the tails of the positive samples. A simple yet effective method to generate such negative samples involves randomly selecting n nodes from the current graph. An artificial edge is created between the positive sample's source node and each of the n randomly selected nodes. This approach is known as a uniform sampling strategy.

In this subsection focused on relation prediction on graph data, GNNs displays their adaptability through two primary edge-level prediction heads. The first involves concatenating the source and target node representation, followed by a linear transformation. This method is suitable for multi-classification or regression tasks on edge properties. The second method computes a scalar value through a dot product, which is ideal for binary classification tasks. This section provided a high-level introduction to the relevant concepts of this thesis. The following section contrasts how this work differs from previous in applying GNN and NLP approaches in the space of OM.

2.4. Differentiation of this Work

The two main contributions of this work are the evaluation of different GNN architectures and the comparison of different negative node sampling strategies. Existing work using GNNs for OM typically lacks this rigid analysis and only presents one single GNN architecture without providing the actual possible search space context. In BioHAN, for example, only GATs are used. The authors do not report the negative node sampling strategy. Similar to BioHAN and Medical Data to Ontology Matching (MEDTO), we also use a pre-trained language model to create the initial node representations used for the training of the GNNs. However, despite using the BioBERT language model, this work sticks to the usage of Bio-Clinical BERT like BERTMap. Similar to BERTMap,

we use a token-based index, but in our work, we do not only leverage it to create candidate matchings for the final evaluation; we also use it to create virtual nodes in order to augment the spare graph structure of the ontologies in our training dataset. Regarding the negative node sampling strategies for the relation prediction, this work extends the common strategy of uniform sampling by investigating the impact of a newly proposed similarity- and structural-based negative sampling strategy.

3. Related Work

The previous chapter only briefly introduced traditional matching systems using no machine learning components within their matching process. It provided the conceptual foundation for understanding ML-based systems. In the previous years, systems using language models and GNNs for OM gained traction. We present the most relevant ones for this work in the following.

BERTMap

The BERTMap [18] system uses the OWL class labels of the provided ontologies to fine-tune the pre-trained Bio-Clinical BERT [35]. The authors argue that compared to traditional matching systems using feature engineering based on string similarity measurements, matching scores based on a deep learning model are superior in capturing semantic similarity. The corpus for the fine-tuning contains text from three sources. First, there is an intra-ontology corpus that only contains synonyms and non-synonyms extracted using *rdfs:label* tags within the respective ontology to be aligned. Second, a cross-ontology corpus connects the ontologies to be aligned using known mappings. Third, a corpus is based on complementary sources, using auxiliary, external ontologies of the same domain. The fine-tuning task is designed as a pair-wise sequence classification. For the prediction phase, BERTMap uses the tokenizer BERT model, and two sub-word inverted indexes, one for each input ontology, are constructed. Each index creates one entry for every token within the labels of an input ontology and holds a linked list of class IDs as a value. Using the indexes, BERTMap conducts a candidate selection. For each token within each class of one ontology, the index of the other ontology is queried. Then, the intersection on the token level is built, a ranking within each intersection based on the *inverted document frequency* is conducted, and the top k elements are chosen. After the generation of the candidates, BERTMap creates a mapping score for each candidate. Candidates achieve a score of 1 if there is an exact string match between one of their associated labels and the classification output of the fine-tuned BERT model otherwise. Finally, only the top-scoring candidates are selected. As a refinement, similar to LogMap, the scored mappings are improved by an extension and a repair step. In addition to the already encountered ontologies, BERTMap includes a task matching National Cancer Institute Thesaurus (NCIT) [36]. BERTMap is evaluated based on the FMA-NCIT, the FMA-SNOMED, and an extended version of the FMA-SNOMED task of the OAEI Large BioMed Track. The proposed system is compared to LogMap and AML on these tasks. BERTMap performs best on both FMA-SNOMED (0.883) and FMA-SNOMED(extended version) (0.886) with respect to the F1-score. On the FMA-NCIT BERTMap can not beat LogMap (0.919) and AML (0.918).

MEDTO

The MEDTO [7] framework was first proposed by Hao et al. to address the alignment of medical databases with ontologies. The authors identify a challenging problem regarding the interoperability of those data sources in real-life applications. The issue originates from differences in the terminological and structural properties of manually expert-crafted ontologies and the databases resulting from live production systems. Another problem is the lack of existing mappings between a database and medical ontologies. MEDTO addresses this problem by introducing a framework containing two phases (Figure 3.1).

The first phase derives a semantically rich ontology from an existing database and consists of an ontology creation and an ontology enrichment using a standard ontology like Unified Medical Language System (UMLS) [37]. A database schema is mapped to concepts during ontology creation, and the primary and foreign keys are utilized to derive the structure between concepts. In the ontology enrichment, additional relationships between concepts are introduced via neighborhood augmentation. Neighborhood augmentation involves transferring relationships of the standard ontology to the derived one.

The second phase is dedicated to OM and includes a training on the derived and the raw ontology to generate representations. To learn representations of the concepts, the authors use two types of GNNs, Hyperbolic Graph Convolution Networks [38] and heterogenous graph networks like RGCN [39]. The initial concept embeddings are computed using the concept names by the BioBERT [40] language model. The authors compute the loss for the hyperbolic GCN layers based on the hyperbolic distance. Cross-entropy loss is used for the heterogenous GNN layers. A uniform-based negative sampling method generates samples for cross-entropy loss. The representations of both hyperbolic GCN and the heterogenous GNN layers are then combined in a matching module using the contrastive matching loss. Finally, all loss values are combined for a joint optimization target.

The experimental results on the OAEI datasets show promising results by an overall F1-score improvement of 4.7% compared to the baseline systems. In comparison MEDTO performs best on Precision for the FMA-NCIT task (0.944), best on the mean reciprocal rank for FMA-SNOMED (0.690) and best for Precision (0.901), F1 (0.849) and mean reciprocal rank (0.704) on the more complex NCIT-SNOMED task.

BioHAN

Finally, we would like to take a closer look at BioHAN [8], a system based on MEDTO. BioHAN emphasizes the use of ontology enriching, leverages a hyperbolic graph attention network [41] to compute representations for each class within an ontology and combines features of distant and close neighbor classes within each class. The enrichment of the input ontologies contains two substeps. The first is a preprocessing, and the second is an augmentation. The preprocessing makes implicit knowledge within an ontology more explicit and applies rule-based reasoning to discover additional knowledge. The augmentation phase of the enrichment should resolve the issue of sparse neighborhood structures within ontologies. The ontologies to be aligned are thus extended by information form UMLS. On the one hand, concepts themselves are augmented by inserting *rdfs:label*, *owl:annotation* and *owl:equivalentClass* tags form UMLS. Additionally, missing relationships between classes are added, e.g., if a relationship is present in UMLS but not in the ontology to be augmented. Similar to BERTMap, BioHAN uses a pre-trained language model, BioBERT [40]. Despite fine-tuning BioBERT, the authors create initial embeddings for their GNN by applying BioBERT to the terminological description of each concept. For the GNN, hyperbolic graph attention layers are used, which encode the initial embeddings from the euclidean to the hyperbolic space. This allows for a better representation of hierarchical structures. Two loss functions are combined for learning the GNN; the first is based on the graph reconstruction loss, and the second is on the contrastive matching loss. As a supervised signal, the authors use positive, pre-aligned concepts and sample five negative concepts from the positive ones. Unfortunately, the paper does

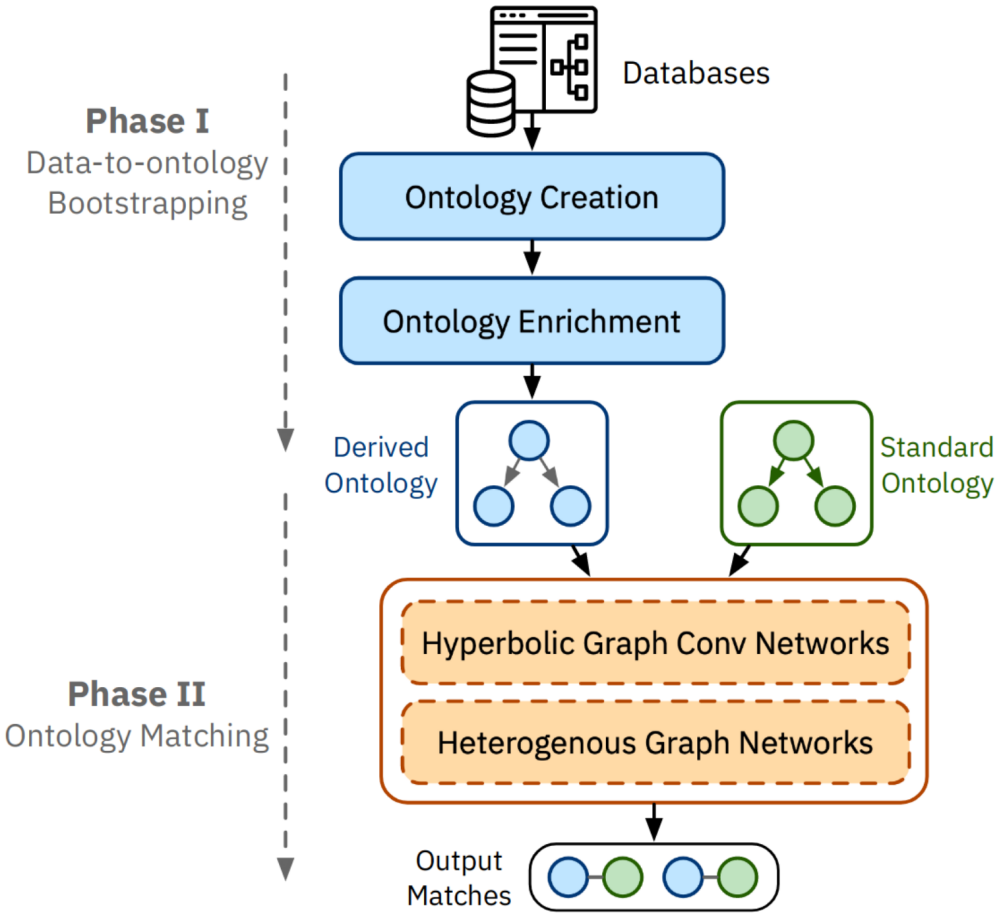

Figure 3.1.: A visualization of the MEDTO [7] matching system. The system consists of two phases. The first one derives an enriched ontology from an existing database. The second phase includes training an GNN based ontology matcher. The matching module is trained simultaneously on two different GNN architectures during the training.

not report how much of these pre-aligned concepts per task were sourced from UMLS. The creators of BioHAN compare their systems matching results to the published evaluations of the Large Biomedical track in OAEI 2021 (three systems) and four other embedding-based matching systems. Compared to the competing systems, BioHAN performs best on recall for the equivalence matching task of FMA-NCIT (0.922) and FMA-SNOMED (0.775) and best with respect on the F1-score on the task of SNOMED-NCIT (0.850).

4. Ontology Matching using Graph Neural Networks

The previous chapter described the work that influenced the creation of our proposed methodology: **O**ntology **M**atching **u**sing graph neural **net**works (OMUNET). OMUNET is a structural/schema-based matching system. This chapter presents the implementation details of our proposed framework. Figure 4.1 provides a visual overview. The framework is comprised of a preprocessing phase and a training phase. The preprocessing phase parses and converts the provided ontologies into a graph structure and further generates artifacts used later in the training phase. This approach reduces the runtime of the overall training. The steps of the preprocessing phase include:

- The creation of the initial embeddings for the GNN using a language model
- The computation of an Annoy[2] index used for the similarity-based negative node sampling
- The creation of a breadth-first search used for the structural-based negative node sampling
- The computation of an inverted token index using the output of the tokenizer created during the initial embedding creation
- The augmentation of the provided ontologies enriching their structural properties by adding virtual nodes based on the inverted token index

The steps of the training phase include:

- The siamese GNN module using different GNN architectures
- The negative graph generation module that derives negative examples from positive ones within the training set
- The combination of node embeddings via a prediction head
- The computation of a loss function serving as the starting point for the gradient descent optimization

Finally, the trained models are evaluated on the test set using three evaluation metrics: F1-score, the Mean Reciprocal Rank (MRR), and HITS@1. section 5.1 provides a detailed explanation of the evaluation scores. Additionally, we have implemented an evaluation that is not only based on the model but also generates candidate matches using the inverted token index of the preprocessing phase. This approach is similar to the evaluation used in BERTMap. Before conducting this final evaluation, we apply a Max1-both postprocessing [42] to enable a one-to-one restriction on the candidate alignments of our system.

[2] https://github.com/spotify/annoy

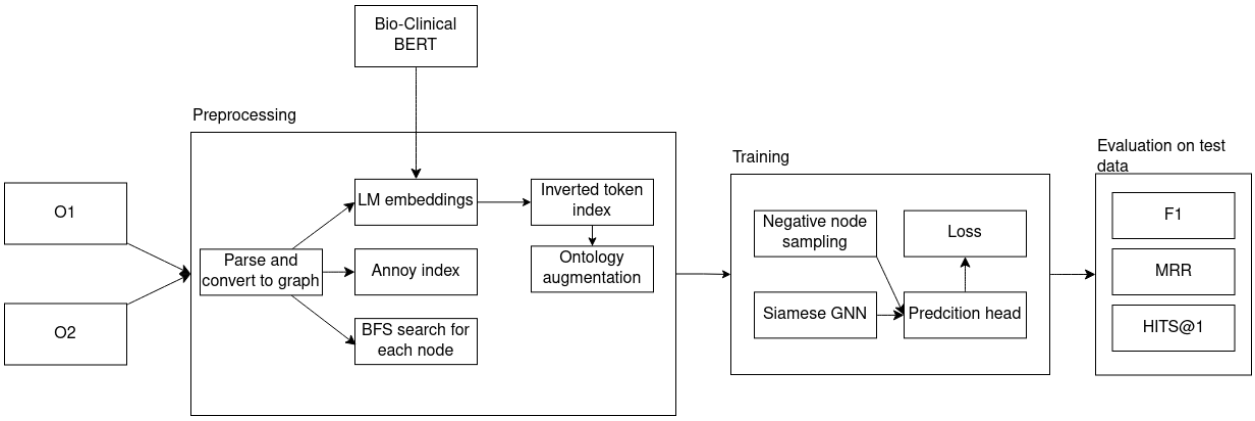

Figure 4.1.: The pipeline of our proposed framework: OMUNET. On a high level, the framework takes two ontologies to match. The first preprocesses them. The second phase uses the artifacts and the augmented ontologies of the preprocessing for training. Finally, we evaluate the trained models on the test set.

4.1. Preprocessing

The preprocessing phase is critical, laying the foundation for our subsequent training and evaluation phases. This section delves into the intricate procedures the OMUNET framework employed during preprocessing. Each step enhances the subsequent training phase's overall efficiency and effectiveness. The ontologies provided to the preprocessing phase originate from the OAEI Bio-ML Track and are given in the OWL format. A detailed description of the used datasets is given in section 5.2

Parsing of the Raw of OWL Files

The read-in of the raw OWL files is facilitated by the Owlready2 [43] python library. From the parsed OWL file, we create tabular data containing the concepts with their textual description and an edge list based on the *is_a* relationship between concepts. This step also resolves any unions or intersections and connects a child with all the concepts found in such a union or intersection. The textual description is derived from the *rdfs:label* properties.

Creation of the Initial Embeddings

The creation of the initial embeddings uses the extracted textual descriptions provided in the previous preprocessing step. We process the textual descriptions in batches using the Bio-Clinical BERT [35] language mode. Each batch is first tokenized, and then a forward pass of the language model uses these tokens as input. We extract the last pooling output layer of the language model as an initial embedding of the respective concepts and save it. Additionally, we save the IDs of the tokens of a single concept to disk.

Computation of Annoy Indices

Our proposed similarity-based negative node sampling approach uses an Annoy [44] index. Annoy allows for an efficient nearest-neighbor search on high-dimensional spaces. The index uses a tree-based index structure, and its distance measurement is based on a heuristic, allowing for fast retrieval at the cost of results precision. As an input, Annoy takes key-value pairs. In our case,

the key is the ID of a node, and the value is the associated word embedding calculated in the previous preprocessing step. Before inserting the ID and its word embedding, we normalize the embedding using the ℓ^2-norm. The normalization enables the cosine similarity to determine the nearest neighbors at retrieval time during the training. We generate separate indices for each input ontology to match.

Creation of Breadth-first Search Mapping

The proposed structural-based negative node sampling approach uses a pre-computed breadth-first search. For the computation of the breadth-first, we use the edge list of the first preprocessing step and an implementation of the graph search algorithm from the NetworkX [45] package for network analysis. We save the results of the breadth-first search in a mapping from node IDs to a two-dimensional list of node IDs; the zeroth element of the list is the 1-hop neighborhood, the first is the 2-hop neighborhood, and so on. Similar to the Annoy indices, this is done twice, once for each input ontology.

Creation of Inverted Token Index

This preprocessing step creates an inverted token index. Two areas within OMUNET use the inverted token index, first, by the following preprocessing step to augment the provided ontologies and second, by the full evaluation (section 5.6) of the trained model. The inverted index is built like standard indices from the information retrieval field. In our case, the keys are the token IDs, and the elements of each associated key are the node IDs. Thus, one token points to the node IDs of the concept it initially occurs in (Algorithm 1). The inverted token index uses the saved tokens of the second preprocessing step.

Algorithm 1 Create Index from node IDs and token IDs

```
tokenIdsAndNodeIds ← loadDf()
index ← new defaultDict(list)
for tokenIds, nodeId in tokenIdsAndNodeIds do
    for tokenId in tokenIds do
        index[token].append(nodeId)
    end for
end for
```

Dataset Augmentation

The first analysis and training results indicated that the provided ontologies are too sparse for stable training in the second phase. We augment the provided ontologies by adding virtual nodes to address this issue. Virtual nodes improve the message passing during the training of GNNs [46, 47]. The virtual nodes are created using the inverted token index of the previous preprocessing step. For each key in the inverted token index, we create a new virtual node and connect it with the node IDs within the list of this key. The evaluation 5.3 analyzes the impact of virtual nodes.

To conclude the preprocessing phase, all artifacts generated in the preceding steps are saved to disk. This step is crucial as it decreases the runtime during the training phase. The saved artifacts encompass the initial embeddings, the Annoy indices, the breadth-first search mappings, and the inverted token index. Furthermore, we enhance the ontologies through augmentation. These

augmented ontologies play a vital role in the training phase, enriching the structural properties of the existing ontologies. The subsequent section will detail the training phase of the OMUNET framework.

4.2. Training Phase

After discussing the prerequisites of the training phase, we will now move forward to the training phase of the OMUNET framework. The model architecture of the training phase is depicted in Figure 4.2. This phase consists of the siamese GNN module containing the respective GNN architecture and leveraging a siamese network approach. Next, we continue with an in-depth description of our negative node sampling approaches. Finally, we explain the prediction head module and the final loss computation.

4.2.1. Siamese Graph Neural Network Model

This subsection explains the siamese GNN model, the core of the OMUNET framework. Besides the siamese network approach, we also use residual connections. The input to this module comprises the initial embeddings of the concepts for each ontology, and the output is the unprocessed representation of those concepts as node embeddings.

Siamese networks were first proposed by Bromley et al. for signature verification [48]. The key idea behind siamese networks is their architecture, which consists of two twin networks joined at the output layer. These twin networks process a different input vector but share the same architecture and weights. The shared weights are a critical feature of siamese networks. This setup means both networks simultaneously learn to embed their respective inputs into a common vector space. This shared embedding space reflects the semantic similarity between the input samples. Thus, similar input vectors are to be close to each other in this space, and dissimilar ones are farther apart.

Another aspect of siamese networks is their use of a specialized loss function, typically a contrastive loss based on cosine similarity. This loss function plays an important role in the training of the networks. It forces the model to minimize the distance between semantically similar pairs of inputs while maximizing the distance between dissimilar pairs. This mechanism allows the network to effectively learn from the relationships between different input pairs, making it particularly suitable for tasks that involve finding similarities or relationships between two distinct inputs, such as signature verification, face recognition, and other applications in the domain of similarity learning [49].

In our architecture, the siamese network approach is utilized in the following way. At first, we provide the initial embeddings of the first ontology to our architecture, performing an iterative application of a respective GNN architecture and a RELU [50] as an activation function. The amount of iterations is configurable as a hyperparameter. The initial input embeddings are saved. At each iteration, we concatenate the output of the RELU activation and use the same output as the input for the next GNN layer. After the iterations, the concatenated result is passed to a postprocessing layer comprised of a fully connected layer. The output of this layer represents the

node embedding of the respective concept within the first input ontology. The same procedure is repeated using the initial embeddings of the second ontology. The weights of the GNN layers are shared between the two input ontologies. In contrast, the postprocessing layer is independent of each input ontology.

Our learning task is set so that node embeddings of concepts that match between both input ontologies must have high similarity and vice versa. Node embeddings whose concepts are no matches between the ontologies must have a low similarity. We theorize that using a siamese style approach benefits the postulated learning task.

Additionally, our architecture uses residual connections [51], originally proposed for CNNs. Using residual connections allows the model to learn a non-fixed computation graph. This is beneficial as the node representations from earlier layers of the GNN might be more important than the later ones. We hypothesize that using residual connections is beneficial for the learning task of OM.

This section provides a detailed explanation of the siamese GNN model, the core of the OMUNET framework. The model architecture uses a shared siamese network model, leveraging residual connections. Our shared network performs n iterations utilizing a GNN layer. The initial embeddings of the concepts from the respective ontology serve as the model's input. Subsequently, the computed node embeddings are integrated and passed to a prediction head, yielding scores that allow for the computation of the loss function. However, the prediction head requires the negative samples, which are the topic of the next section.

4.2.2. Negative Node Sampling

The previous section describes the overall architecture of our GNN and provided the first insights into the learning setup. The GNN learns in a supervised fashion, requiring positive and negative samples. Current research indicates that this area of GNNs needs further discussion and analysis [52]. In our case, the provided datasets contain reference matches between two ontologies, thus only providing positive samples for training. A positive sample is a tuple containing the IDs of the respective concepts within the two ontologies. From a positive sample, we keep the source concept ID in place and replace the target concept ID by picking a new concept ID in the target ontology. The approach to picking the target concept describes the negative node sampling strategy. Per positive sample, we generate k negative samples. This procedure is repeated with target and source swapped, thus generating a total $2k$ negative samples per positive sample. The most basic negative node sampling strategy is random sampling (subsection 2.3.3). However, this work implements two more sophisticated negative node sampling strategies: similarity-based and structural-based negative node sampling. The following sections provide a detailed explanation of each negative node sampling strategy.

Structural-based negative node sampling

The structural-based negative node sampling strategy uses the saved breadth-first search mappings from the preprocessing phase. This approach is inspired by a breadth-fist search-based negative node sampling strategy for the model-specific evaluation presented in [6]. Given a positive matching sample (u_2, v_2) containing a source concept u_2 and a target concept of v_2 of the ontology U and

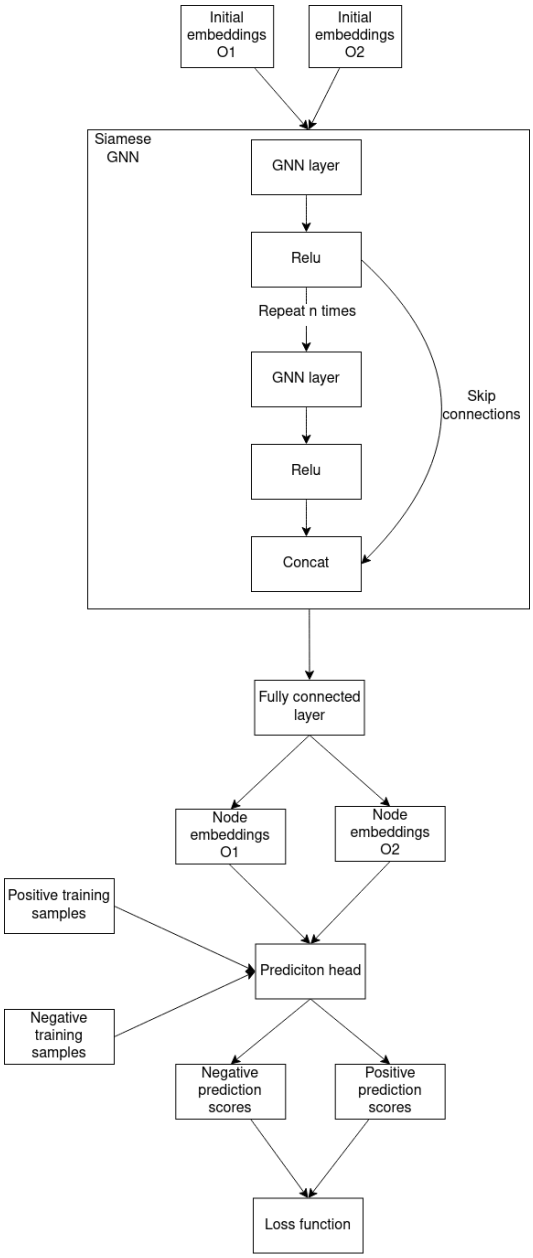

Figure 4.2.: An illustration of the architecture. The core model uses a shared siamese network, augmented with residual connections. This shared network performs n iterations utilizing a GNN. The initial embeddings of the concepts from the respective ontology serve as the model's input. Subsequently, the computed node embeddings are integrated via a prediction head, yielding scores that allow for the computation of the loss function.

the ontology V, we perform a lookup in the precomputed breadth-first search for U using u_2. This returns a collection of u_2's neighbors sorted by their distance with respect to hops from u_2. This list is first restricted to the first l closest neighbors. Next, we randomly sample the k neighbors of u_2 within the list of the l closest neighbors. Finally, we create negative training samples by replacing the source concept u_2 with the k neighbors. To sum up, this approach requires two hyperparameters l and k. l restricts the amount of nearest neighbor and k samples within the l nearest neighbor. During learning, this provides more difficult negative samples for the training process, which potentially allows the GNN to learn better node representations. Our proposed structural-based negative node sampling strategy is illustrated in Figure 4.3.

Similarity-based negative node sampling

Our second proposed negative node sampling strategy leverages the Annoy index, that was first computed for every ontology during the preprocessing. For a positive sample (u_2, v_2), we first look up the associated initial embedding $\mathbf{u}_2$ of u_2. Next we normalize $\mathbf{u}_2$ using the ℓ^2-norm and then use the Annoy index of ontology U to retrieve the k nearest neighbors of $\mathbf{u}_2$. The similarity between the provided query $\mathbf{u}_2$ and the indexed embeddings is determined by the cosine similarity. The result is a list of the k most similar concept IDs. Those are then used to generate k negative samples by corrupting the source concept u_2.

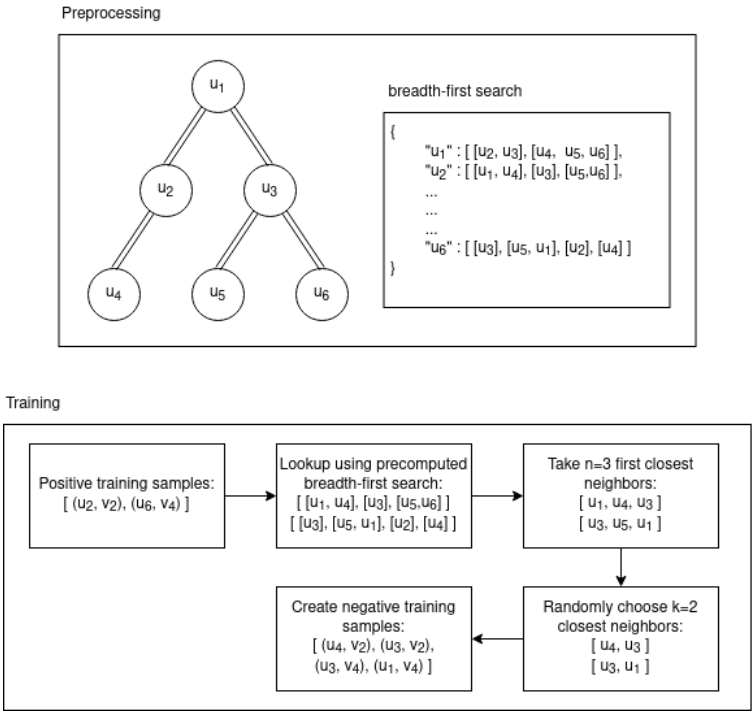

Figure 4.3.: A visualization of the structural-based negative node sampling strategy. The strategy uses the precomputed breadth-first search to sample negative nodes. Intuitively, the strategy is based on the idea that the nearest neighbors of a concept are more likely to be similar to the concept but not a real match, thus pushing the model to potentially better discrimination. The strategy is parameterized by l and k. l restricts the amount of nearest neighbors and k the amount of samples within the l nearest neighbors.

This subsection introduced the negative node sampling strategies used in the OMUNET framework. Our implemented strategies include random sampling, similarity-based sampling, and structural-based sampling. Following the negative node sampling, the next section will explain the prediction head and the loss computation.

4.2.3. Prediction Head and Loss

Having positive and negative samples, we move forward to the prediction head and the loss function. The prediction head is a basic dot product predictor as already described in section 2.3.3. Our model calls the prediction head twice, once for the positive samples and once for the freshly generated negative samples, following a strategy described in the previous subsection. The output of our

prediction head is a scalar value representing a confidence score that the two source and target nodes are either a positive or a negative sample. Next, the scores of all the positive and negative samples are passed to the loss function.

At initial testing, we experimented with different loss functions. The loss functions we tested non-exhaustively were cross-entropy loss, BRP [53] loss and multi-margin loss [54]. From our initial tests, we could not determine performance or convergence differences during the training of our model and thus settled on the multi-margin loss by convention. The multi-margin loss is an extension of the margin-loss first proposed in the context of SVMs [55]. Intuitively, this loss defines a margin that helps separate the provided classes. Having a concept u, v and their positive sample score of $y_{u,v}$ and a process $P_n(v)$ sampling k negative samples from v, the multi-margin loss is defined as follows[3]:

$$\mathcal{L} = \frac{1}{k} \sum_{v_i \sim P_n(v), i=1,\ldots,k} \max(0, M - y_{u,v} + y_{u,v_i}) \tag{4.1}$$

Here, the margin M represents the minimum distance that is required between the positive and negative samples. The loss is small when a single positive sample score $y_{u,v}$ is larger than that of the negative samples y_{u,v_i} by at least the margin M. Finally, we use an average overall k loss values as a reduction method.

4.3. Implementation Details

This section provides further insights into various implementation details of our OMUNET framework that were not covered in the previous sections but are still relevant for completeness and reproducibility. This includes the description of the software libraries used, the execution environment, and the hyperparameters used for the training.

Software Libraries

OMUNET relies on Python 3.11 and exposes a command line interface using the typer package[4]. The first preprocessing phase especially relies on Owlready2 to extract the ontology into a tabular format using the pandas library[5]. Additionally, the preprocessing phase relies on the transformers[6] package loading the Bio-Clinical BERT model from HuggingFace and NetworkX [45] for the breadth-first search. Finally, we use the Deep Graph Library (DGL) [56] leveraging its high-level API to instantiate and train the GNN architectures. For the backend of DGL, we choose PyTorch [54]. We found seaborn[7] creating appealing visualizations for our final evaluation.

[3] Equation adopted from the DGL tutorial on link prediction: https://docs.dgl.ai/guide/training-link.html
[4] https://typer.tiangolo.com/
[5] https://pandas.pydata.org/
[6] https://huggingface.co/docs/transf1rmers/index
[7] https://seaborn.pydata.org/

Execution Environment

The Clara AI cluster of the University of Leipzig[8] determines the hardware environment. Our initial embeddings were derived using a single NVIDIA Tesla V100 graphics card. The training phase of our model was parallelized per matching tasks over multiple nodes in the cluster using a total of 100 CPUs. The parallelization was enabled by using a SLRUM[9] script.

Hyperparameters

Our framework uses multiple hyperparameters during the training phase. The most interesting one within the context of this work is the amount of k negative matches sampled per positive match and ontology during the negative node sampling. During our experiments, we set $k = 5$. Thus, we generate a total of 10 negative matches per positive one. For the optimization of the GNN, we use the Adam optimizer [57] and use the default parameters provided by PyTorch. The training loop runs for 200 epochs, and within each epoch, the trained model performs $n = 5$ iterations of message passing using the GNN architectures provided by DGL. Besides the amount of GNN layers, the size of the hidden embeddings during the iterations and the size of the final output embeddings are also configurable. For the dimension of the hidden embeddings, we use 200, and for the output embeddings, we use 100 to keep the memory requirements reasonable. Finally, we use a margin of 0.5 for the multi-margin loss. Besides the hyperparameter of k, the structural negative node sampling takes a second hyperparameter of l, restricting the amount of nearest neighbors to sample from. We set $l = 20$. All of the hyperparameters were determined non-exhaustively by manual tuning. The listed hyperparameters are all easily modifiable via a central configuration file.

This chapter describes our methodology of the proposed OMUNET framework. The framework is structured into two primary phases: preprocessing and training. The preprocessing phase is initially tasked with generating initial embeddings and computing Annoy indices. It also establishes the breadth-first search, assembles the inverted token index, and augments the provided ontologies. After this, the training phase focuses on training the siamese GNN model. This phase capitalizes on various negative node sampling strategies, essential for calculating the predictive scores and the loss function necessary for the model's learning process. After detailing our methodology, the following chapter will evaluate the OMUNET framework.

[8] https://www.sc.uni-leipzig.de/
[9] https://www.schedmd.com/

5. Evaluation

The following chapter describes the evaluation of our proposed framework. First, we provide an overview of the reported metrics (section 5.1) and the datasets used for the training and the evaluation (section 5.2), originating from the Bio-ML Track of the OAEI [6]. The evaluation contains two different approaches. The first evaluation approach uses a traditional evaluation strategy within the machine learning context by splitting a dataset into train, validation, and test sets. We also refer to this approach as model-specific evaluation or, in the sense of He et al., as global matching [6]. This approach is used by the evaluation of the influence of virtual nodes (section 5.3), the comparison of the different GNN architectures (section 5.4), and the evaluation of the negative node sampling strategies (section 5.5). The results of the model-specific evaluation are each reported on five different random seeds. Each run using a different seed has a different random train, validation, and test set split, providing additional robustness to our reported results. In the second evaluation approach, we follow an evaluation regime more common in the area of OM by first generating candidate matches and then applying our learned node embeddings of the training phase to compute matching scores (see section 5.6). We also refer to this evaluation as a full evaluation or, like He et al., as global matching [6].

5.1. Metrics

This section presents the evaluation metrics that are used for the evaluation of the proposed OMUNET framework. On the one side, we are reporting the performance of our trained models using metrics commonly used in OM like the F1-score, but we also present results based on the MRR and the HITS@k measurement. According to [6], these ranking-based evaluation metrics are better suited for the evaluation of ML-based OM systems as they do not require a computationally expensive generation of all possible output mappings of a system.

The F1-score evaluates the accuracy of a binary classification model. The score is the harmonic mean of precision and recall (Equation 5.1), where precision is the number of true positive results divided by the number of all positive results, including those not identified correctly. Recall is the number of true positive results divided by the number of all samples that should have been identified as positive. The higher the F1-score, the better the model's performance. Especially in setups where the classes are unevenly distributed, the F1-score is superior, as it accounts for false positives and false negatives, compared to the accuracy.

$$F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{5.1}$$

Precision and recall are the standard textbook definitions in the context of our model-specific evaluation. For the full evaluation, applying the trained models on a candidate set and determining only the existence of matches, the precision, and recall are different following the definition of [6].

$$\text{precision} = \frac{|M_{\text{out}} \cap M_{\text{test}}|}{|M_{\text{out}}|}, \quad \text{recall} = \frac{|M_{\text{out}} \cap M_{\text{test}}|}{|M_{\text{test}}|}, \tag{5.2}$$

The precision is the cardinality of the result set of the intersection between the set of matches that are determined as true matches by our system M_{out}, and the cardinality of the test subset of the provided reference matches M_{test} divided by M_{out}. Following the precision, the recall is defined as the division of the same numerator as in the precision but using the cardinality of M_{test} as a denominator.

Moving on to the next evaluation metrics, which are only used for the model-specific evaluation, there are the MRR and HITS@k. For our proposed framework using GNN, these ranking-based metrics are a natural fit as they require negative and positive samples for learning and should provide high-ranking scores for true matches and low-ranking scores for true non-matches. The MRR is given by Equation 5.3. Given a single true positive match from the test set of reference matches, we calculate the reciprocal ranking score $\text{Rank}(m)^{-1}$ of the true positive match within the list of the k negative samples generated by one of our negative node sampling strategies (subsection 4.2.2).

$$MRR = \frac{\sum_{m \in M_{\text{test}}} \text{Rank}(m)^{-1}}{|M_{\text{test}}|} \tag{5.3}$$

Finally, we build the average of the ranking scores over all the elements in the test set. The MRR allows to compare different system configurations concerning the consistency of ranking true matches higher than non-matches. This is especially useful in having a model that should learn to identify the existence of matches rather than the nonexistence of non-matches.

A similar benefit is given by the HITS@k measurement, which is also commonly used in information retrieval ranking tasks. The HITS@k score is less restrictive than the MRR as it counts not only the rank of the first true positive match but accounts for all the true positive matches within the top k scores. The calculation for the HITS@k score is provided in Equation 5.4. It first calculates the true positive test set matches, a subset of the reference matches, whose scores are ranked within the first k positions. The elements participating in a ranking are the true positive match and its derived true negative ones generated by our negative node sampling strategy. Then, the cardinality of this set is divided by the cardinality of all test set reference matches. In our evaluation setting, we set $k = 1$. With the HITS@1 score becomes more restrictive than the MRR. In this setting, the positive test set matches contribute to the score if it has the highest score among its negative counterparts. Otherwise, it does not contribute to the overall increase of the HITS@1 score.

$$\text{Hits@k} = \frac{|\{m \in M_{\text{test}} \mid \text{Rank}(m) \leq k\}|}{|M_{\text{test}}|} \tag{5.4}$$

This section introduced the scoring metrics used within this work and described the different evaluation setups. The first evaluation setup is model-specific, relying on positive and negative matching samples, and the second evaluation setup first generates candidates that scored using one of our trained models and are then evaluated. This second evaluation, more common in OM, only

considers positive matches from the reference matchings. Following the presentation of our utilized metrics, the next section will briefly introduce the datasets used to evaluate OMUNET.

5.2. Datasets

The upcoming section describes the datasets we use to evaluate the models trained within the context of our framework. In total, there are five different matching tasks using seven ontologies. The datasets of the matching tasks are provided by He et al. in the context of the Bio-ML track of the OAEI [6].

There are two assignments for each matching task: subsumption matching and equivalence matching. Our proposed system is designed only to handle the equivalence matching task. Along the seven matching tasks, He et al. provide reference matches for each task that are, with respect to the ontologies of an individual matching task, either sourced from UMLS or Mondo Disease Ontology (Mondo) [58] which both have already integrated knowledge in the biomedical domain.

The following briefly describes the ontologies in use and the matching tasks. First, there are ontologies describing concepts within the domain of human diseases:

- Online Mendelian Inheritance in Man (OMIM) [59]
- Orphanet Rare Disease Ontology (ORDO) [**vasantORDOOntologyConnecting**]
- NCIT
- Human Disease Ontology (DOID) [60]

From these ontologies, the matching tasks between OMIM-ORDO and NCIT-DOID are created using the existing integrated Mondo as a source for the reference matchings. Next, there are ontologies dealing with categories sourced from UMLS: "Body Part, Organ or Organ Components", "Pharmacologic Substances", and "Neoplastic Process":

- FMA
- NCIT
- SNOMED

Using the semantic categories of UMLS the reference matchings for the following matching tasks between these three ontologies are derived: SNOMED-FMA (Body), SNOMED-NCIT (Pharm) and SNOMED-NCIT (Neoplas) [6].

For matching systems using supervised learning, He et al. suggest using a 20% training set, 10% validation set, and 70% test set splitting. The authors argue that such a dataset split is required for the OM use case as there is an immensely skewed distribution of true positives and true negatives. Concretely speaking, between two ontologies to match, there are many more non-matches than actual matches. Unfortunately, we could not produce meaningful results using this little data for our training phase and thus used an 80% training set, 10% validation, and 10% test set split.

Initially, this work started using the 2022 version of the datasets. During the implementation of OMUNET, the 2023 version was released, adding additional concepts within the ontologies. These newly added concepts should only be used during the training and not during the alignment. The artifacts of the preprocessing phase, namely the Annoy index and the precomputed breadth-first search, are used for training and the model-specific evaluation. Thus, these newly added concepts are only taken into consideration by the message passing of the respective GNN and not by our proposed negative node sampling strategies.

The matching tasks greatly vary in the sizes of their ontologies. However, the provided reference matches only sometimes reflect this, making certain matching tasks inherently harder than others (Table 5.1). The matching task of SNOMED-FMA (Body), for example, has a total of 123373 concepts that should be used for the final evaluation and 7256 reference matches for the equivalence matching task. The smallest task OMIM-ORDO has 6.5 times fewer concepts but only half the about of reference matchings, making the SNOMED-FMA harder. Besides the difference in the availability of reference matches between the tasks, the structural complexity of the ontologies also varies. Using the average depth (the average shortest path of the concepts within an ontology to the root node *owl:Thing*), one can observe that, for example, the task of matching SNOMED to NCIT for the Neoplas category has an average depth of 1.15 for SNOMED and 1.68 for NCIT, which potentially makes this task challenging for matching systems relying on structural information. Further, it is interesting to have imbalances between two ontologies that should be matched. In SNOMED-FMA, for example, the average depth has a difference of 4.81, indicating that FMA potentially has a more differentiated concept hierarchy than SNOMED.

Task	**#Src concept**	**#Tgt concept**	**#Ref (equiv)**	**Avg Depth**
NCIT-DOID	15,762 (+8,927)	8,465 (+17)	4,686	2.04-6.85
OMIM-ORDO	9,648 (+6)	9,275 (+437)	3,721	1.44-1.63
SNOMED-FMA (Body)	34,418 (+10,236)	88,955 (+24,229)	7,256	1.86-9.32
SNOMED-NCIT (Neoplas)	22,971 (+11,700)	20,247 (+6291)	3,804	1.15-1.68
SNOMED-NCIT (Pharm)	29,500 (+13,455)	22,136 (+6,886)	5,803	1.09-3.26

Table 5.1.: A table describing the matching tasks. The data describes the OAEI Bio-ML track tasks of 2022. The numbers in parenthesis indicate the number of concepts added in the 2023 version compared to the 2022 version. We use the added concepts only for the training. The average depths of the ontologies are based on the shortest path between a concept and *owl:Thing* and only consider the concepts used during the alignment.

This section introduced the ontologies and matching tasks used to evaluate our framework. We offered insights into the origin of the reference matchings used during the equivalence matching and provided a short overview of the ontologies' size and structural complexity. The next sections will present the evaluation of adding virtual nodes to the initial ontologies of each matching task.

5.3. Evaluation of the Influence of Virtual Nodes

Starting with the first evaluation, we analyze the effect of adding virtual nodes to the training data following the process described in section 4.1. For the experiments, we used the hyperparameter as described in section 4.3 and initially used the Graph SAGE as a GNN-layer and uniform-based negative node sampling. The results of this experiment will serve as our baseline for further comparison in the following evaluations. The results, averaged over the matching tasks and the random seeds of the individual runs, indicate that, in contrast to our initial hypothesis, adding virtual nodes does not improve the model's performance (See Table 5.2).

Experiment Name	**Precision**	**Recall**	**F1-score**	**MRR**	**HITS@1**
With Virtual Nodes	0.705 (0.054)	0.750 (0.131)	0.720 (0.082)	0.909 (0.037)	0.840 (0.063)
Without Virtual Nodes	0.713 (0.057)	0.778 (0.085)	0.742 (0.057)	0.913 (0.038)	0.848 (0.061)

Table 5.2.: A table showing the results the first evaluation using Graph SAGE and uniform-based negative sampling. A single cell contains the mean and the standard deviation (in parenthesis) of the respective metric aggregated over five different random seeds and the matching tasks.

The F1-score without virtual nodes is 0.742, and including the virtual nodes during training, the average F1-score drops to 0.72. The same trend is observed for precision and recall. The MRR and HITS@1 scores are also slightly lower when using virtual nodes. The standard deviations of the reported scores are marginally different for the two setups. Interestingly, this is not the case for the recall, where the standard deviation is 0.046 points higher when using virtual nodes. Overall, one could not determine from averaging over all the matching tasks that the raw or the setup using the virtual nodes is more performant, as the error intervals overlap. The HITS@1 score lies at 0.848 for the setup without virtual nodes and 0.84 for the setup with virtual nodes. This indicates that the model struggles to rank the positive samples higher than the generated negative ones for both setups.

A clearer picture is provided by Figure 5.1, showing an aggregation of the F1-score over the random seeds only. The analysis of the individual matching tasks shows that adding virtual nodes influences the results heterogeneously. The setup without virtual nodes produces a higher average F1-score than with virtual nodes within all matching tasks. The highest value reached by the setup with virtual nodes is for the NCIT-DOID matching task, reaching an average F1-score of 0.804. Conversely, the lowest average F1-score is reached in the OMIM-ORDO task, resulting in only 0.656. More interesting is the observed standard deviation for the NCIT-DOID and SNOMED-FMA (Body) tasks. The F1-score standard deviation for these tasks is 0.0476 and 0.16, much higher than in direct comparison to the setup without virtual nodes in the respective task. This indicates that adding virtual nodes might harm the stability of the training process.

To sum up, the results of evaluating the influence of virtual nodes are not in line with our initial hypothesis. The results show that the raw setup without adding virtual nodes is marginally better. Despite this finding, we keep the setup using the virtual nodes for the following evaluations as

our proposed structural-based negative sampling approach requires them. The next section will evaluate the different GNN layers.

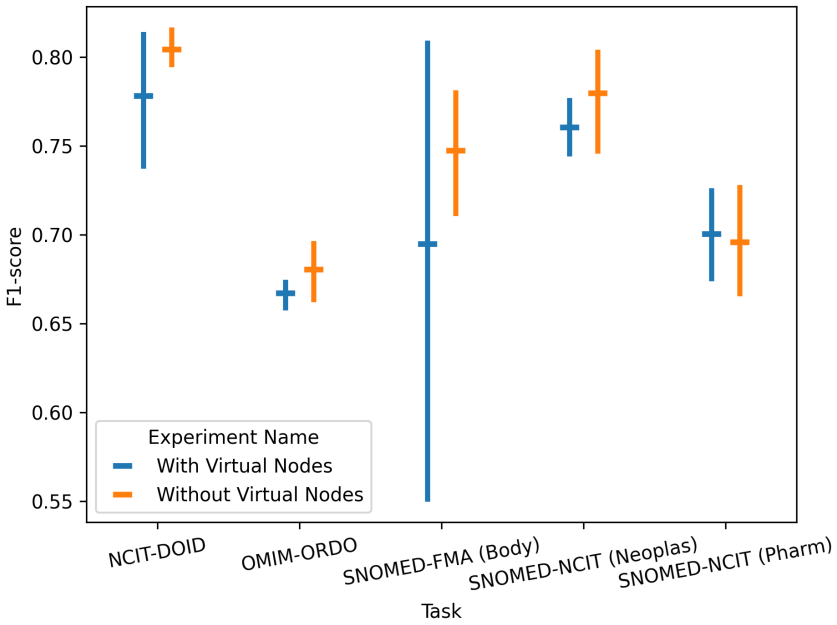

Figure 5.1.: A plot showing the F1-score per matching task. Each task is contains two setups, one with and one without the virtual nodes. The results are aggregated over five random seeds. The horizontal bar shows the average over the seeds and error bars indicate the standard deviation. The averages indicate that the setup without virtual performs marginally better.

5.4. Comparison of Graph Neural Network Architectures

This evaluation analyzes the different GNN architectures. In subsection 2.3.2, we introduced three different architectures, GCN, Graph SAGE, and GAT. The results of the evaluation are presented in Figure 5.2. For this experiment, we use the default configuration of the GNN modules provided by the DGL library. To be consistent with the next experiment comparing the negative node sampling strategies, we decided to include the virtual nodes in this experiment.

A first look at the F1-scores averaged over the different matching tasks and random seeds (Table 5.3) indicates that the GCN model is performing best (0.782). GAT and Graph SAGE perform nearly similar with respect to the averaged F1-score (0.732 and 0.72). The same trend is observed for the ranking-based metrics (MRR and HITS@1). Interestingly, GAT and GCN show a slightly higher recall (0.0801 and 0.816) than the baseline on Graph SAGE (0.705). Similar to the previous evaluation, one could not finally determine a best-performing model by looking at this level of aggregation. One could only assume that GCN performs better.

This assumption solidifies by diving into the per-task aggregation level of the experimental runs (Figure 5.2). All the average F1-scores of the tasks using the GCN model perform better. For the tasks of OMIM-ORDO and SNOMED-FMA (Body), the GCN model does not show an overlap concerning the standard deviation of the F1-score in comparison to the other two architectures.

The highest observed average F1-score achieves the GCN model on the SNOMED-FMA (Body) task with a value of 0.868. The lowest score is reached by the GAT architecture, with a value of 0.638 on the OMIM-ORDO task. Graph SAGE and GAT perform similarly on most of the tasks, except for the SNOMED-FMA (Body) task. Here, the GAT architecture is more performant than the Graph SAGE model. The GAT architecture shows high variance in the F1-score for three tasks, Graph SAGE for two, and GCN for one.

Furthermore, we analyzed the convergence behavior of the different GNN architectures plotting the loss development on the validation set throughout the 200 epochs (Appendix A). This analysis shows that the GCN converges faster than the other two architectures on two of three tasks. Additionally, Graph SAGE and GAT show a high instability in convergence behavior for the tasks SNOMED-FMA, SNOMED-NCIT (Pharm) and SNOMED-NCIT (Neoplas). On these three tasks, the GCN model architecture only shows instabilities during the end of the training. During the OMIM-ORDO task, all architectures struggled to converge properly, which can be attributed to the low average depth of the two ontologies, hampering the message passing of the GNNs.

Experiment Name	**Precision**	**Recall**	**F1-score**	**MRR**	**HITS@1**
GAT	0.679 (0.068)	0.801 (0.127)	0.732 (0.087)	0.915 (0.046)	0.849 (0.078)
GCN	0.758 (0.066)	0.816 (0.127)	0.782 (0.090)	0.940 (0.038)	0.895 (0.064)
Graph SAGE	0.705 (0.054)	0.750 (0.131)	0.720 (0.082)	0.909 (0.038)	0.840 (0.063)

Table 5.3.: A table showing the mean and the standard deviation (in parenthesis) of the reported metrics aggregated over five different random seeds and the matching tasks. Concerning the F1-score, the GCN architecture performs best. The MRR and HITS@1 scores show a similar trend to the F1-score. The standard deviations of the recall are all higher than the ones of precision.

5.5. Evaluation of Negative Node Sampling Strategies

Our final evaluation centers around the three negative node sampling strategies. The previous experiments used uniform-based negative sampling. This section further introduces the results of the structural and similarity-based negative node sampling. Regarding the GNN architecture, we switch to the GCN for this experiment, as the previous one indicates a more stable learning process using the GCN architecture. Further, we are required to include the virtual nodes in this experiment as the structural-based node sampling requires them to expand the neighborhood of the respective concepts. The virtual nodes are not included in any of the artifacts used by the structural or similarity-based approach.

Figure 5.3 indicates that the proposed similarity-based approach performs worse than the other two. Between the baseline (uniform-based) and the structural-based approach, the structural-based one performs slightly better regarding the averaged F1-scores. This is the case for the tasks NCIT-DOID, OMIM-ORDO and SNOMED-NCIT (Pharm). By looking at the standard deviations of the F1-scores, the picture is not so clear anymore. Here, only the OMIM-ORDO task has non-overlapping standard deviation intervals. Despite the structural-based approach being slightly more performant,

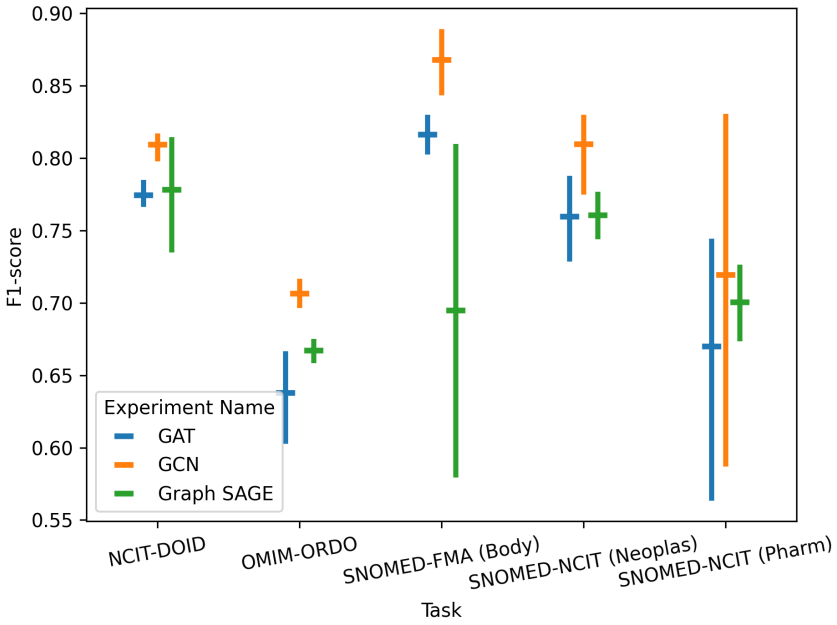

Figure 5.2.: A plot showing the F1-score per matching task. Each task contains three setups, one for each GNN architecture. The results are aggregated over five random seeds. The horizontal bar shows the average over the seeds, and error bars indicate the standard deviation. On average, the GCN architecture performs best.

the uniform-based reaches the highest observed average F1-score of 0.832 on the SNOMED-FMA (Body) task. On the other hand, the lowest observed value is reached by the similarity-based approach on the OMIM-ORDO task, reaching only 0.376.

Similar to the previous evaluation of the architectures, we also analyze the convergence behavior of the negative node sampling strategies (see Appendix B). The similarity-based approach initially converges faster in four tasks and shows less variance in the loss behaviors over the random seeds. Within the context of the NCIT-DOID task, the approach has problems finding a well-performing optimum, having a final loss that is a magnitude higher than the other two approaches. Furthermore, the similarities-based behavior during the SNOMED-FMA (Body) task shows nearly no variance and converges within the first 50 epochs. Despite having a lower validation set loss than the other approaches, its inferior results on the average F1-scores indicate a problem with overfitting. Overall, the structural and uniform-based approaches show similar convergence behavior, with the structural-based approach being more unstable during the end of the training for the tasks of SNOMED-FMA (Body) and SNOMED-NCIT (Pharm) and SNOMED-NCIT (Neoplas). For the OMIM-ORDO task, there is a bigger difference in the loss development between the two approaches. The uniform-based approach converges slower initially but finally reaches a lower loss than the structural and similarity-based approaches.

To summarize this section, the similarity-based approach performs worst, and the structural-based approach is slightly more performant than the uniform-based approach. The next section will present the full evaluation setup, using the best performing GNN architecture and the best performing negative node sampling strategy.

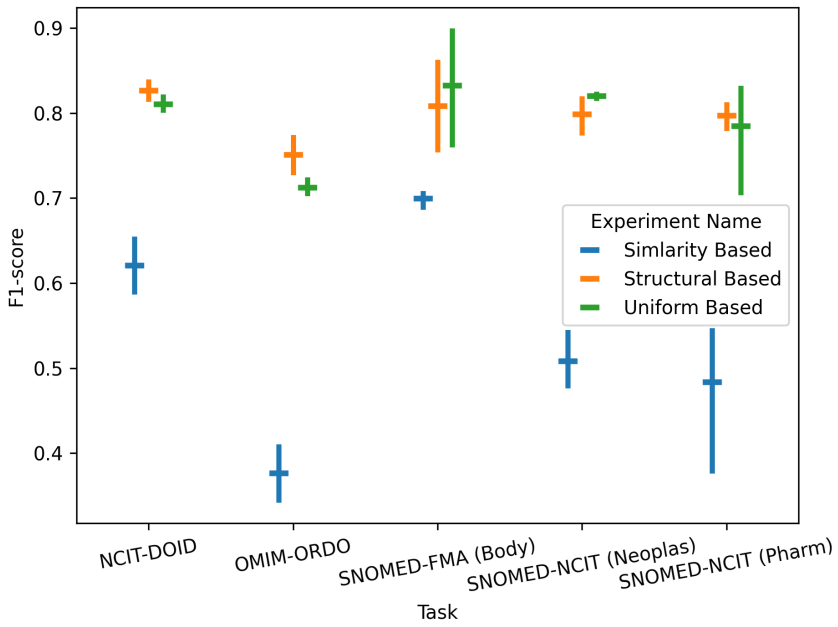

Figure 5.3.: A plot showing the F1-score per matching task. Each task is contains three setups, one for each negative node sampling strategy. The results are aggregated over five random seeds. The horizontal bar shows the average over the seeds. The error bars indicate the standard deviation. Our similarity-based approach performs worst. The structural-based approach performs slightly better than the uniform-based approach in three tasks.

5.6. Full Evaluation Setup

The full evaluation setup only considers positive matches from the test subset of reference matchings. It consists of three extra steps in comparison to the model-specific evaluation. First, before the model training, we conduct a candidate generation step using the inverted token indices of the preprocessing phase and generating 200 candidates per concept within each ontology. Second, after training our model, we save the trained node embeddings and use those to compute a cosine similarity between the potential candidates. Third and finally, we perform a Max1-both [42] postprocessing to ensure a one-to-one relationship between the source and target nodes of the final alignment. Initially, we tried to use the evaluation module of the DeepOnto python-package[10], which is distributed by the organizers of the Bio-ML track. However, we were not able to compile the required dependencies. Thus, we decided to implement the candidate generation and the final evaluation ourselves using the resources in [6] and [18]. We use the GCN architecture and the structural-based negative node sampling strategy for the full evaluation, as they performed best in the previous evaluations.

Table 5.4 shows the full evaluation results. For this experiment, we do not conduct runs over multiple random seeds. Overall, the results are rather low-performing. For the F1-score, we report a best score of 0.183 for the NCIT-DOID task followed by SNOMED-NCIT (Pharm) with an F1-score of 0.156. OMIM-ORDO, SNOMED-FMA (Body) and SNOMED-NCIT (Neoplas) have very low F1-scores all under a value of 0.01, ranging from 0.096 to 0.087. All of our full evaluation runs have

[10]https://krr-oxford.github.io/DeepOnto/

a better recall than precision. The top recall values are subsequently also reached by NCIT-DOID and SNOMED-NCIT (Pharm). The precision is below 0.1 for all the tasks except NCIT-DOID.

Task	**Precision**	**Recall**	**F1-score**
NCIT-DOID	0.110	0.554	0.183
OMIM-ORDO	0.051	0.306	0.087
SNOMED-FMA (Body)	0.045	0.293	0.078
SNOMED-NCIT (Neoplas)	0.056	0.332	0.096
SNOMED-NCIT (Pharm)	0.090	0.561	0.156

Table 5.4.: A table showing the result of the full evaluation using 10% of the reference matches as a test set. The metrics' calculation is based on applying the node embeddings to a set of candidate alignments and a subsequent Max1-both postprocessing. Concerning the F1-score the results could be more satisfying. The model configuration uses a GCN architecture and the structural-based negative node sampling strategy.

This chapter presents the evaluation of our work. The evaluation splits between a model-specific and a full evaluation, relying on candidate generation. We started the section with an explanation of the reported metrics. We have ranking-based metrics that are only used for model-specific evaluations. Further, we described the matching tasks given by the Bio-ML track of the OAEI. Regarding the model-specific evaluations, we observed that adding virtual nodes neither improves nor worsens the model's performance. Our comparison of GNN architectures revealed that the GCN architecture provides the best results while having a more stable convergence behavior. The third model-specific evaluation took a closer look at the different negative node sampling strategies, revealing that the structural-based approach is slightly more performant than the uniform-based one. Finally, our full evaluation setup provided insights that our final model is not performing well on a traditional OM evaluation setup. The next chapter will provide a discussion of the results.

6. Discussion

The previous chapter reports our experimental results without further interpretation. Therefore, this chapter picks up the presented results and discusses them in light of the initially stated research questions. Additionally, we place our findings in the context of the related literature. The discussion finally ends with a reflection of the work's limitations.

The key results of the evaluation of our OMUNET framework are the following:

- Despite our expectations, the influence of adding virtual nodes to the provided ontologies does not influence the reported performance metrics significantly.
- The comparison of the three different GNN architectures indicates that the GCN model outperforms the other two models in terms of F1-score and ranking-based metrics.
- Our evaluation of negative node sampling strategies shows that the proposed similarity-based negative sampling performs worst, and the proposed structural-based negative sampling performs slightly better than the baseline.
- Using a full evaluation utilizing a candidate generation process indicates that the final model of OMUNET is a rather low-performing OM system.

Surprisingly, adding virtual nodes to the ontologies does not influence the results positively. Given the low average depth for ontologies like OMIM (avg. depth of 1.44), SNOMED (Pharm) (1.09) and SNOMED (Neoplas) (1.15), we hypothesized that the structural enrichment of the ontologies would allow the GNN model, that relies on this rich graph structure to learn node representations that are more suitable for the discrimination. This may indicate that adding virtual nodes based on the inverted token index is not the right approach to enable enrichment. We assume that connecting concepts to virtual nodes based on the intersection of the tokens derived from the concepts themselves might add too many edges to the virtual nodes, increasing the direct neighborhood of each concept too much and thus not providing a clear benefit to the GNN during the message passing.

Comparing the different GNN architectures, the GCN model outperforms the other two. Initially, we were motivated by the successful application of GATs in [8], now we were surprised that in our setup, the GAT GNN architecture performs worst. Thus, hyperbolic encoding seems to be essential when applying GATs to ontologies. Together with the unstable convergence behavior depicted in Appendix A, we interpret this finding as a sign that a more complex GNN architecture does not necessarily lead to an increase in performance results. We conclude that the more complex GAT model does not generalize well to the unseen test data due to an over-fitting on the test set. The GCN and Graph SAGE architectures differ in their normalization approach during the message passing (see section 2.3). The GCN model uses a symmetric normalization based on the square root of the multiplication of the neighborhood sizes of the nodes. In contrast, the Graph SAGE model uses a mean aggregation. This difference, a potentially lower normalization factor for big neighborhoods, allows the GCN model to capture the graph structure better and thus learn better node representations, resulting in higher F1-scores.

The last model-specific evaluation (section 4.2.2) shows that the proposed negative nodes sampling strategies significantly outperform the outline, thus only partially answering our second research question (RQ2, section 1.2). The question is which of the proposed negative sampling has the biggest positive influence on the final prediction scores. This question remains unanswered as the proposed structural-based approach, on average, only performs slightly better than the baseline using the standard uniform-based sampling approach. We can say that the proposed similarity-based approach is performing worst. We assume this is because the sampled negative matching pairs that should be scored by the OMUNET framework do not provide a clear signal for learning the discrimination between matches and non-matches during the training phase. This assumption is especially supported by the convergence behavior of the similarity-based approach on the SNOMED-NCIT (Pharm) tasks in Appendix B, showing that the model converges very fast to an over-fitting optimum without exploring the optimization space. Our initial hypothesis that the structural-based approach should perform especially well for ontologies is not fully supported by the results, as it only ranks second concerning the average F1-score on the SNOMED-FMA (Body) and NCIT-DOID task.

Finally, we would like to comment on the results of the full evaluation setup. During the model-specific evaluations, we observed average ranking-based scores ranging from 0.909 to 0.94 for the MRR and from 0.84 to 0.895 for the HITS@1 value. This indicates that the created learning setup has problems consistently providing the highest scoring value to a true positive match. The full evaluation setup ultimately confirms this preliminary indication. Even if it is not directly comparable to our setup as we modified the split of the reference matchings, as a reference, the BERTMap model reports F1-scores ranging from 0.877 to 0.967 on the equivalence matching tasks of the Bio-ML track [6]. Ultimately, this leads to the conclusion that the best configuration of our proposed OMUNET framework is not able to perform higher on the global and local matching scores than the existing methods, thus answering our first research question (RQ1, section 1.2).

Before concluding, we want to discuss further remarks regarding our evaluation setups. Firstly, we ease the matching tasks provided by the authors of the Bio-ML track. Instead of using the proposed 20%, 10%, and 70%, split of the reference mappings into train, validation, and test set, we modify this to 80%, 10%, and 10%, which increases the amount of available training data for our models significantly. Nevertheless, we could not generate matching scores coming close to the ones of systems like BERTMap that report their result on the far more restrictive dataset split. This modification of the dataset split is why we refrain from directly comparing to other systems participating at the Bio-ML track.

Another noteworthy remark of this work is the missing exploration of the hyperparameter of the OMUNET framework, especially for the hyperparameter introduced by our negative node sampling strategies. The imposed hyperparameters were only heuristically tuned by multiple manual tests without relying on a proper automated hyperparameter search. It might be possible that a robust hyperparameter search would have led to different, potentially better, results.

7. Conclusion

In conclusion, the initial motivation to investigate an ML-based OM approach enriching textual representations of concepts leads to an interesting research result. We pose two research questions that guide our analysis. The first one asks if a GNN-based approach performs better than existing ML-based approaches, and the second one asks which of our negative node sampling strategies has the biggest positive influence on the matching score of the implemented GNN-based matching prediction. To answer these questions, we developed OMUNET. A framework that enables the usage of different GNN architectures and negative node sampling strategies for the relation prediction in the context of OM.

The evaluation of our OMUNET framework provides valuable insights into the relationship between the application of GNNs to the equivalence matching Bio-ML tasks of the OAEI and the influence of different negative node sampling strategies on the matching scores. Our proposed method cannot perform higher on the matching scores than existing methods. Regarding our two proposed negative node sampling strategies, we conclude that the similarity-based approach performs worse than the existing standard of using a uniform-based negative node sampling, and the second proposed strategy, the structural-based one, only performs marginally better. Thus, we can only partially answer our second research question.

Even if the results of this work are rather under-performing, we believe that the OMUNET framework provides valuable insights to the research field of OM. This is especially true for the choice of GNN architecture and negative node sampling strategy. Looking ahead, future work on OMUNET should involve a more rigorous hyperparameter search to provide further robustness to our findings. In addition to the hyperparameter search, efforts should focus on a further increase of the comparability of our results. Finally, a study of the relationship between the structural richness of the ontologies being matched and the convergence behavior would also be of valuable interest.

Bibliography

[1] Cornelius Rosse and José L. V. Mejino. "A Reference Ontology for Biomedical Informatics: The Foundational Model of Anatomy". In: *Journal of Biomedical Informatics* 36.6 (2003), pp. 478–500. ISSN: 1532-0464. DOI: 10.1016/j.jbi.2003.11.007. pmid: 14759820.

[2] Kevin Donnelly. "SNOMED-CT: The Advanced Terminology and Coding System for eHealth". In: *Studies in Health Technology and Informatics* 121 (2006), pp. 279–290. ISSN: 0926-9630. pmid: 17095826.

[3] J. J. Cimino and X. Zhu. "The Practical Impact of Ontologies on Biomedical Informatics". In: *Yearbook of Medical Informatics* 15.01 (2006), pp. 124–135. ISSN: 0943-4747, 2364-0502. DOI: 10.1055/s-0038-1638470. URL: http://www.thieme-connect.de/DOI/DOI?10.1055/s-0038-1638470.

[4] Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching, Second Edition*. Springer, 2013. ISBN: 978-3-642-38720-3.

[5] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/V1/N19-1423. URL: https://doi.org/10.18653/v1/n19-1423.

[6] Yuan He et al. "Machine Learning-Friendly Biomedical Datasets for Equivalence and Subsumption Ontology Matching". In: *The Semantic Web - ISWC 2022 - 21st International Semantic Web Conference, Virtual Event, October 23-27, 2022, Proceedings*. Ed. by Ulrike Sattler et al. Vol. 13489. Lecture Notes in Computer Science. Springer, 2022, pp. 575–591. DOI: 10.1007/978-3-031-19433-7_33. URL: https://doi.org/10.1007/978-3-031-19433-7_33.

[7] Junheng Hao et al. "MEDTO: Medical Data to Ontology Matching Using Hybrid Graph Neural Networks". In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. Virtual Event Singapore: ACM, 2021, pp. 2946–2954. ISBN: 978-1-4503-8332-5. DOI: 10.1145/3447548.3467138. URL: https://dl.acm.org/doi/10.1145/3447548.3467138.

[8] Peng Wang and Yunyan Hu. "Matching Biomedical Ontologies via a Hybrid Graph Attention Network". In: *Frontiers in Genetics* 13 (2022). ISSN: 1664-8021. URL: https://www.frontiersin.org/articles/10.3389/fgene.2022.893409.

[9] Oxford English Dictionary. *Ontology, n., Sense 1.a.* In: Oxford University Press, 2023. URL: https://doi.org/10.1093/OED/1029030509.

[10] Nicola Guarino, Daniel Oberle, and Steffen Staab. "What Is an Ontology?" In: *Handbook on Ontologies*. Ed. by Steffen Staab and Rudi Studer. International Handbooks on Information Systems. Springer, 2009, pp. 1–17. URL: https://doi.org/10.1007/978-3-540-92673-3_0.

[11] Michael Uschold and Michael Gruninger. "Ontologies and Semantics for Seamless Connectivity". In: *ACM SIGMOD Record* 33.4 (2004), pp. 58–64. ISSN: 0163-5808. DOI: 10.1145/1041410.1041420. URL: https://dl.acm.org/doi/10.1145/1041410.1041420.

[12] *OWL 2 Web Ontology Language Document Overview (Second Edition)*. URL: https://www.w3.org/TR/owl2-overview/ (visited on 10/26/2023).

[13] Franz Baader et al., eds. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003. ISBN: 0-521-78176-0.

[14] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. "The Even More Irresistible SROIQ". In: *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006*. Ed. by Patrick Doherty, John Mylopoulos, and Christopher A. Welty. AAAI Press, 2006, pp. 57–67. URL: http://www.aaai.org/Library/KR/2006/kr06-009.php.

[15] I Horrocks. "A Description Logic with Transitive and Inverse Roles and Role Hierarchies". In: *Journal of Logic and Computation* 9.3 (1999), pp. 385–410. ISSN: 0955-792X, 1465-363X. DOI: 10.1093/logcom/9.3.385. URL: https://academic.oup.com/logcom/article-lookup/doi/10.1093/logcom/9.3.385.

[16] Manfred Schmidt-Schauß and Gert Smolka. "Attributive Concept Descriptions with Complements". In: *Artificial Intelligence* 48.1 (1991), pp. 1–26. ISSN: 00043702. DOI: 10.1016/0004-3702(91)90078-X. URL: https://linkinghub.elsevier.com/retrieve/pii/000437029190078X.

[17] Erhard Rahm and Philip A. Bernstein. "A Survey of Approaches to Automatic Schema Matching". In: *The VLDB Journal* 10.4 (2001), pp. 334–350. ISSN: 10668888. DOI: 10.1007/s007780100057. URL: http://link.springer.com/10.1007/s007780100057.

[18] Yuan He et al. "BERTMap: A BERT-Based Ontology Alignment System". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.5 (2022), pp. 5684–5691. ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v36i5.20510. URL: https://ojs.aaai.org/index.php/AAAI/article/view/20510.

[19] 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. Red. by David Hutchison et al. Vol. 8185. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 527–541. ISBN: 978-3-642-41029-1 978-3-642-41030-7. DOI: 10.1007/978-3-642-41030-7_38. URL: http://link.springer.com/10.1007/978-3-642-41030-7_38.

[20] Isabel F. Cruz, Flavio Palandri Antonelli, and Cosmin Stroe. "AgreementMaker: Efficient Matching for Large Real-World Schemas and Ontologies". In: *Proceedings of the VLDB Endowment* 2.2 (2009), pp. 1586–1589. ISSN: 2150-8097. DOI: 10.14778/1687553.1687598. URL: https://dl.acm.org/doi/10.14778/1687553.1687598.

[21] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. "LogMap: Logic-Based and Scalable Ontology Matching". In: *The Semantic Web – ISWC 2011*. Ed. by Lora Aroyo et al. Vol. 7031. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 273–288. ISBN: 978-3-642-25072-9 978-3-642-25073-6. DOI: 10.1007/978-3-642-25073-6_18. URL: http://link.springer.com/10.1007/978-3-642-25073-6_18.

[22] Giorgos Stoilos, Giorgos Stamou, and Stefanos Kollias. "A String Metric for Ontology Alignment". In: *The Semantic Web – ISWC 2005*. Ed. by Yolanda Gil et al. Red. by David Hutchison et al. Vol. 3729. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 624–637. ISBN: 978-3-540-29754-3 978-3-540-32082-1. DOI: 10.1007/11574620_45. URL: http://link.springer.com/10.1007/11574620_45.

[23] Dan Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 2nd Edition*. Prentice Hall Series in Artificial Intelligence. Prentice Hall, Pearson Education International, 2009. ISBN: 978-0-13-504196-3. URL: https://www.worldcat.org/oclc/315913020.

[24] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. "A Neural Probabilistic Language Model". In: *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*. Ed. by Todd K. Leen, Thomas G. Dietterich, and Volker Tresp. MIT Press, 2000, pp. 932–938. URL: https://proceedings.neurips.cc/paper/2000/hash/728f206c2a01bf572b5940d7d9a8fa4c-Abstract.html.

[25] Tomás Mikolov et al. "Recurrent Neural Network Based Language Model". In: *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*. Ed. by Takao Kobayashi, Keikichi Hirose, and Satoshi Nakamura. ISCA, 2010, pp. 1045–1048. DOI: 10.21437/INTERSPEECH.2010-343. URL: https://doi.org/10.21437/Interspeech.2010-343.

[26] Ashish Vaswani et al. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon et al. 2017, pp. 5998–6008. URL: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[27] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90. URL: https://doi.org/10.1109/CVPR.2016.90.

[28] Zonghan Wu et al. "A Comprehensive Survey on Graph Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1 (2021), pp. 4–24. ISSN: 2162-237X, 2162-2388. DOI: 10.1109/TNNLS.2020.2978386. URL: https://ieeexplore.ieee.org/document/9046288/.

[29] William L. Hamilton. *Graph Representation Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2020. ISBN: 978-3-031-00460-5. URL: https://doi.org/10.2200/S01045ED1V01Y202009AIM046.

[30] William L. Hamilton, Rex Ying, and Jure Leskovec. "Representation Learning on Graphs: Methods and Applications". In: *IEEE Data Eng. Bull.* 40.3 (2017), pp. 52–74. URL: http://sites.computer.org/debull/A17sept/p52.pdf.

[31] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=SJU4ayYgl.

[32] William L. Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive Representation Learning on Large Graphs". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon et al. 2017, pp. 1024–1034. URL: https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html.

[33] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco.1997.9.8.1735. URL: https://direct.mit.edu/neco/article/9/8/1735-1780/6109.

[34] Petar Velickovic et al. "Graph Attention Networks". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL: https://openreview.net/forum?id=rJXMpikCZ.

[35] Emily Alsentzer et al. "Publicly Available Clinical BERT Embeddings". In: *CoRR* abs/1904.03323 (2019). URL: http://arxiv.org/abs/1904.03323.

[36] Jennifer Golbeck et al. "The National Cancer Institute's Thésaurus and Ontology". In: *Journal of Web Semantics* 1.1 (2003), pp. 75–80. DOI: 10.1016/J.WEBSEM.2003.07.007. URL: https://doi.org/10.1016/j.websem.2003.07.007.

[37] O. Bodenreider. "The Unified Medical Language System (UMLS): Integrating Biomedical Terminology". In: *Nucleic Acids Research* 32.90001 (2004), pp. 267D–270. ISSN: 1362-4962. DOI: 10.1093/nar/gkh061. URL: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkh061.

[38] Ines Chami et al. "Hyperbolic Graph Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al. 2019, pp. 4869–4880. URL: https://proceedings.neurips.cc/paper/2019/hash/0415740eaa4d9decbc8da001d3fd805f-Abstract.html.

[39] Michael Sejr 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*. Ed. by Aldo Gangemi et al. Vol. 10843. Lecture Notes in Computer Science. Springer, 2018, pp. 593–607. DOI: 10.1007/978-3-319-93417-4_38. URL: https://doi.org/10.1007/978-3-319-93417-4_38.

[40] Jinhyuk Lee et al. "BioBERT: A Pre-Trained Biomedical Language Representation Model for Biomedical Text Mining". In: *Bioinformatics* 36.4 (2020). Ed. by Jonathan Wren, pp. 1234–1240. ISSN: 1367-4803, 1367-4811. DOI: 10.1093/bioinformatics/btz682. URL: https://academic.oup.com/bioinformatics/article/36/4/1234/5566506.

[41] Maximilian Nickel and Douwe Kiela. "Poincaré Embeddings for Learning Hierarchical Representations". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon et al. 2017, pp. 6338–6347. URL: https://proceedings.neurips.cc/paper/2017/hash/59dfa2df42d9e3d41f5b02bfc32229dd-Abstract.html.

[42] Martin Franke et al. "Post-Processing Methods for High Quality Privacy-Preserving Record Linkage". In: *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Ed. by Joaquin Garcia-Alfaro et al. Vol. 11025. Cham: Springer International Publishing, 2018, pp. 263–278. ISBN: 978-3-030-00304-3 978-3-030-00305-0. DOI: 10.1007/978-3-030-00305-0_19. URL: http://link.springer.com/10.1007/978-3-030-00305-0_19.

[43] Jean-Baptiste Lamy. "Owlready: Ontology-oriented Programming in Python with Automatic Classification and High Level Constructs for Biomedical Ontologies". In: *Artificial Intelligence in Medicine* 80 (2017), pp. 11–28. ISSN: 09333657. DOI: 10.1016/j.artmed.2017.07.002. URL: https://linkinghub.elsevier.com/retrieve/pii/S0933365717300271.

[44] *Spotify/Annoy*. Spotify, 2023. URL: https://github.com/spotify/annoy (visited on 09/17/2023).

[45] Aric A Hagberg, Daniel A Schult, and Pieter J Swart. "Exploring Network Structure, Dynamics, and Function Using NetworkX". In: (2008).

[46] Trang Pham et al. *Graph Classification via Deep Learning with Virtual Nodes*. 2017. arXiv: 1708.04357 [cs, stat]. URL: http://arxiv.org/abs/1708.04357. preprint.

[47] Katsuhiko Ishiguro, Shin-ichi Maeda, and Masanori Koyama. *Graph Warp Module: An Auxiliary Module for Boosting the Power of Graph Neural Networks in Molecular Graph Analysis*. 2019. arXiv: 1902.01020 [cs, stat]. URL: http://arxiv.org/abs/1902.01020. preprint.

[48] Jane Bromley et al. "Signature Verification Using a Siamese Time Delay Neural Network". In: *Advances in Neural Information Processing Systems 6, [7th NIPS Conference, Denver, Colorado, USA, 1993]*. Ed. by Jack D. Cowan, Gerald Tesauro, and Joshua Alspector. Morgan Kaufmann, 1993, pp. 737–744. URL: http://papers.nips.cc/paper/769-signature-verification-using-a-siamese-time-delay-neural-network.

[49] Hugh M. Cartwright, ed. *Artificial Neural Networks - Third Edition*. Vol. 2190. Methods in Molecular Biology. Springer, 2021. ISBN: 978-1-07-160825-8. URL: https://doi.org/10.1007/978-1-0716-0826-5.

[50] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep Sparse Rectifier Neural Networks". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*. Ed. by Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudík. Vol. 15. JMLR Proceedings. JMLR.org, 2011, pp. 315–323. URL: http://proceedings.mlr.press/v15/glorot11a/glorot11a.pdf.

[51] Andreas Veit, Michael J. Wilber, and Serge J. Belongie. "Residual Networks Behave like Ensembles of Relatively Shallow Networks". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. Ed. by Daniel D. Lee et al. 2016, pp. 550–558. URL: https://proceedings.neurips.cc/paper/2016/hash/37bc2f75bf1bcfe8450a1a41c200364c-Abstract.html.

[52] Zhen Yang et al. "Understanding Negative Sampling in Graph Representation Learning". In: *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*. Ed. by Rajesh Gupta et al. ACM, 2020,

pp. 1666–1676. DOI: 10.1145/3394486.3403218. URL: https://doi.org/10.1145/3394486.3403218.

[53] Steffen Rendle et al. "BPR: Bayesian Personalized Ranking from Implicit Feedback". In: *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*. Ed. by Jeff A. Bilmes and Andrew Y. Ng. AUAI Press, 2009, pp. 452–461. URL: https://www.auai.org/uai2009/papers/UAI2009_0139_48141db02b9f0b02bc7158819ebfa2c7.pdf.

[54] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs, stat]. URL: http://arxiv.org/abs/1912.01703. preprint.

[55] Corinna Cortes and Vladimir Vapnik. "Support-Vector Networks". In: *Machine Learning* 20.3 (1995), pp. 273–297. ISSN: 0885-6125, 1573-0565. DOI: 10.1007/BF00994018. URL: http://link.springer.com/10.1007/BF00994018.

[56] Da Zheng et al. "Scalable Graph Neural Networks with Deep Graph Library". In: *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*. Ed. by Rajesh Gupta et al. ACM, 2020, pp. 3521–3522. DOI: 10.1145/3394486.3406712. URL: https://doi.org/10.1145/3394486.3406712.

[57] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[58] Nicole A Vasilevsky et al. *Mondo: Unifying Diseases for the World, by the World*. preprint. Health Informatics, 2022. DOI: 10.1101/2022.04.13.22273750. URL: http://medrxiv.org/lookup/doi/10.1101/2022.04.13.22273750.

[59] Joanna S. Amberger et al. "OMIM.Org: Online Mendelian Inheritance in Man (OMIM(®)), an Online Catalog of Human Genes and Genetic Disorders". In: *Nucleic Acids Research* 43 (Database-Issue 2015), pp. 789–798. DOI: 10.1093/NAR/GKU1205. URL: https://doi.org/10.1093/nar/gku1205.

[60] Lynn M. Schriml et al. "Human Disease Ontology 2018 Update: Classification, Content and Workflow Expansion". In: *Nucleic Acids Research* 47 (Database-Issue 2019), pp. D955–D962. DOI: 10.1093/NAR/GKY1032. URL: https://doi.org/10.1093/nar/gky1032.

Declaration

I affirm that I have written the present work with the subject:

"Ontology Matching using Graph Neural Networks"

independently and only by using the sources and aids indicated; in particular, verbatim or analogous quotations are marked as such. I am aware that any infringement can also lead to the subsequent withdrawal of the degree. I assure that the electronic copy corresponds to the printed copies.

Leipzig, den 16.02.2024

_____

Jerome Markus Louis Würf

A. Appendix

A. Validation Set Loss per Task for Different Graph Neural Network Architectures

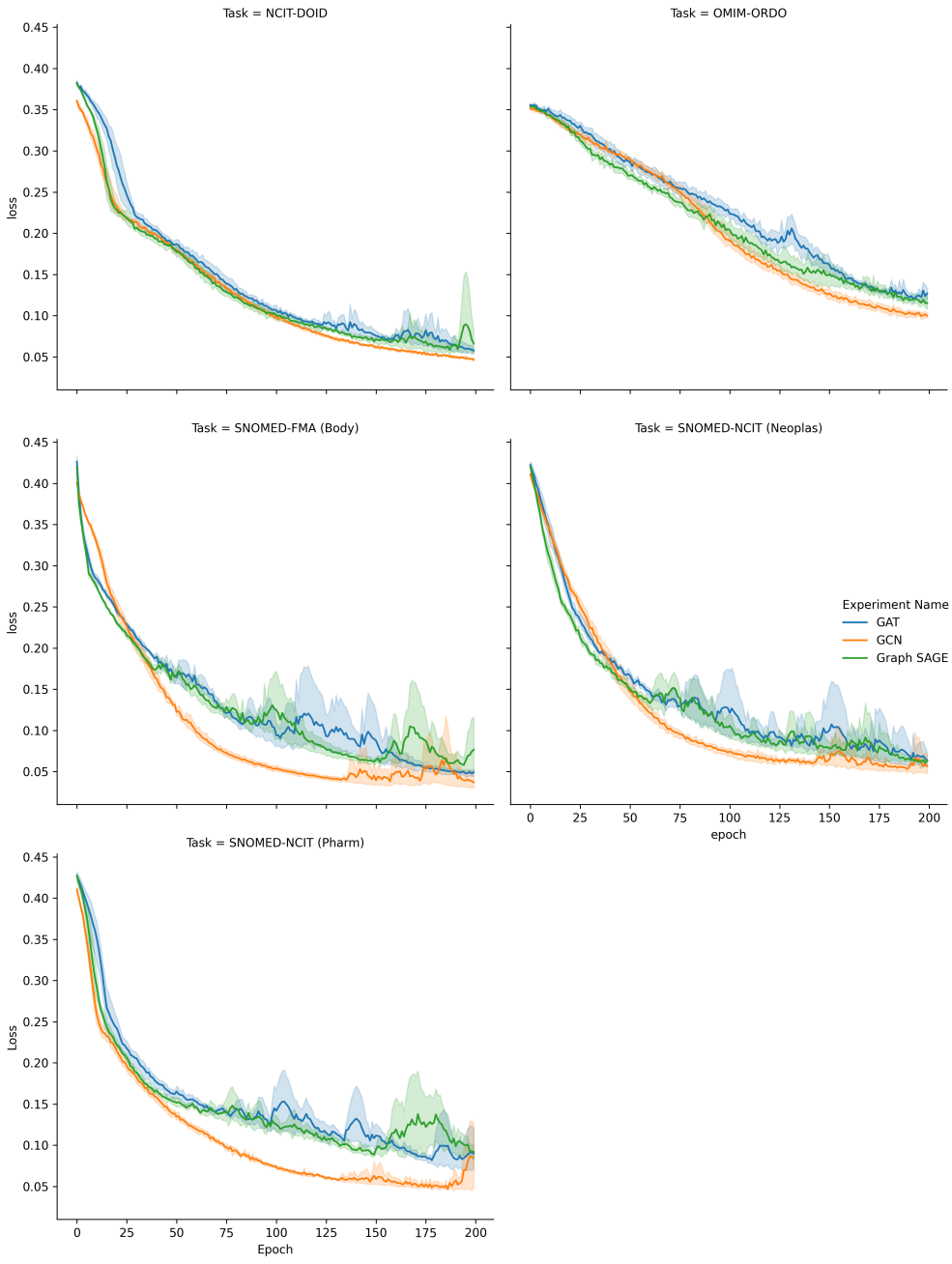

Figure A.1.: Multiple plots displaying the validation set loss development over the course of the epochs. The plots are separated by the different tasks. Within each plot, the loss of the different GNNs is displayed. The GCN architecture shows the smoothest loss development. GAT and Graph SAGE are more volatile.

B. Validation Set Loss per Task for the Negative Node Sampling Strategies

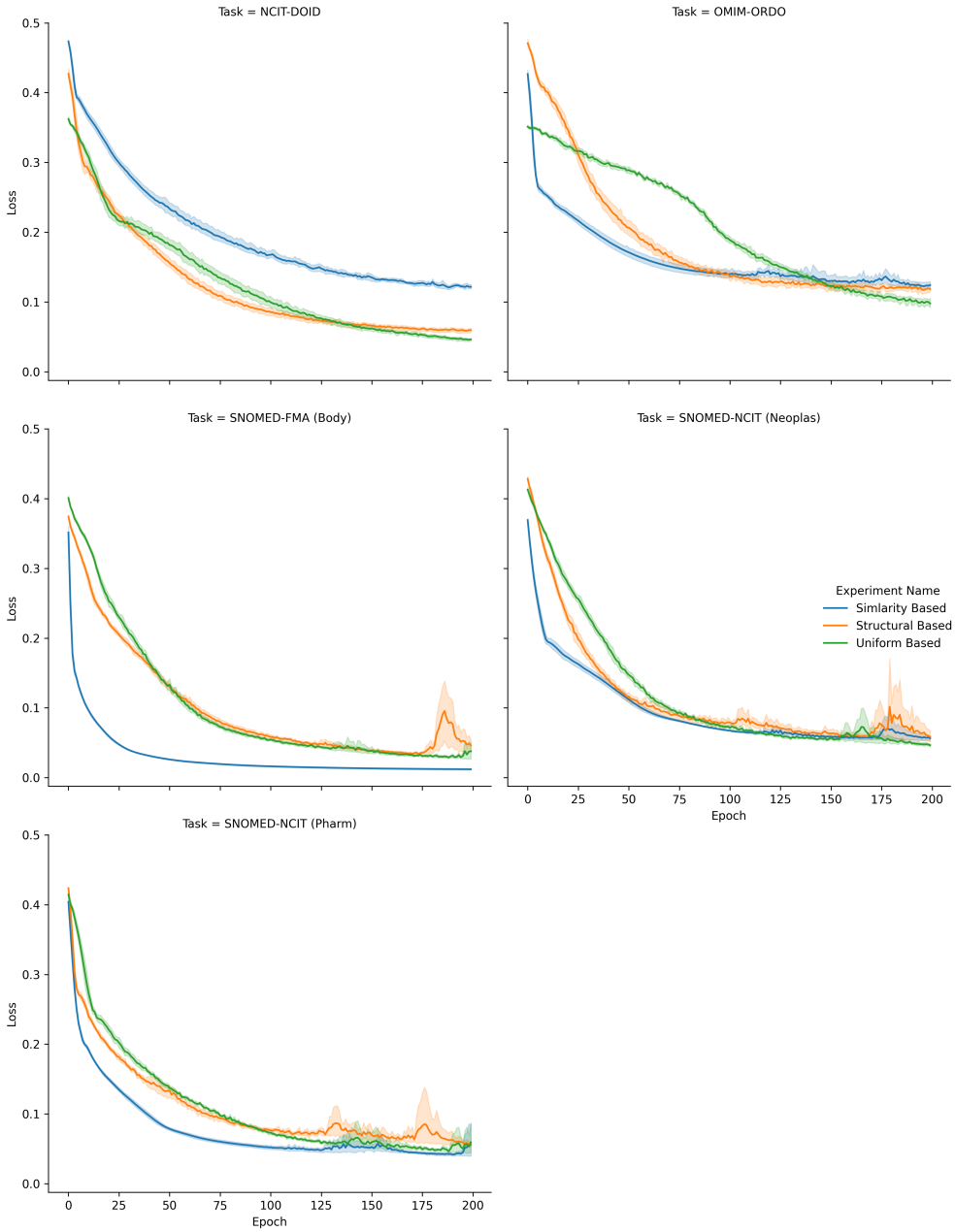

Figure A.2.: Multiple plots displaying the validation set loss development over the course of the epochs. The plots are separated by the different tasks. Within each plot, the loss of the negative node sampling strategies is displayed. For all the tasks the similarity-based strategy converges the fastest. The uniform-based strategy shows the most well behaved loss development. The structural-based strategy shows instabilities during the training at the end of three tasks.