Medical Entity Disambiguation Using Graph Neural Networks

Alina Vretinaris[1*], Chuan Lei[2], Vasilis Efthymiou[3*], Xiao Qin[2], Fatma Özcan[4*]
[1]IBM Germany, Ehningen, Baden-Württemberg, Germany
[2]IBM Research - Almaden, 650 Harry Road, San Jose, CA 95120
[3]FORTH - Institute of Computer Science, Heraklion, Crete, Greece
[4]Google, 1600 Amphitheatre Parkway, Mountain View, CA, 94043
alina.vretinaris|chuan.lei|xiao.qin@ibm.com,vefthym@ics.forth.gr,fozcan@google.com

ABSTRACT

Medical knowledge bases (KBs), distilled from biomedical literature and regulatory actions, are expected to provide high-quality information to facilitate clinical decision making. Entity disambiguation (also referred to as entity linking) is considered as an essential task in unlocking the wealth of such medical KBs. However, existing medical entity disambiguation methods are not adequate due to word discrepancies between the entities in the KB and the text snippets in the source documents. Recently, graph neural networks (GNNs) have proven to be very effective and provide state-of-the-art results for many real-world applications with graph-structured data. In this paper, we introduce ED-GNN based on three representative GNNs (GraphSAGE, R-GCN, and MAGNN) for medical entity disambiguation. We develop two optimization techniques to fine-tune and improve ED-GNN. First, we introduce a novel strategy to represent entities that are mentioned in text snippets as a query graph. Second, we design an effective negative sampling strategy that identifies hard negative samples to improve the model's disambiguation capability. Compared to the best performing state-of-the-art solutions, our ED-GNN offers an average improvement of 7.3% in terms of F1 score on five real-world datasets.

CCS CONCEPTS

• Information systems → Data cleaning; **• Theory of computation → Data integration**; **• Computing methodologies → Neural networks**.

KEYWORDS

Entity disambiguation; graph neural network; medical ontology

ACM Reference Format:
Alina Vretinaris, Chuan Lei, Vasilis Efthymiou, Xiao Qin, and Fatma Özcan. 2021. Medical Entity Disambiguation Using Graph Neural Networks. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD'21), June 18–27, 2021, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3448016.3457328

*Work done while at IBM Research.

1 INTRODUCTION

Recent years have witnessed the rapid growth in medical knowledge bases (KBs), curated from healthcare data, such as clinical resources, electronic health records, and lab tests. Tremendous effort has been put into developing automated medical KB construction [11, 47] and completion [31, 45]. Existing systems often face one major challenge, *entity disambiguation* (ED): how to map entity mentions in text snippets from medical source documents to their corresponding entities in a medical KB.

Text snippets in healthcare data are often collected from heterogeneous data sources. Discrepancies arise for many reasons, including acronyms, abbreviations, typos and colloquial terms. As a result, text snippets may deviate significantly from the canonical descriptions of the entities in the KB that they refer to. For example, an editorial staff member may mention *"renal disorder"* or *"kidney disease"* in a text snippet, with the intention to refer to the entity that is defined as *"nephrosis"* in the KB. Similarly, *"cah"* in a text snippet may refer to the entity defined as *"chronic active hepatitis"*. Such discrepancies make it difficult to link textual entity mentions to the intended entities in a KB, introducing noise, duplicates, and ambiguity, eventually decreasing the value of the KB.

While early works often relied on rule-based [17, 22, 40] and dictionary-based approaches [36, 41], more recent state-of-the-art ED solutions rely on machine learning methods. In particular, deep learning (DL) methods [7, 15, 38, 47] are commonly used due to their powerful feature abstraction and generalization capabilities. A recent study [30] of various DL-based methods for entity matching, concluded that they significantly outperform other solutions (e.g., [15]) for textual entity matching. However, existing DL methods either resolve mentions only relying on textual context information from the surrounding words [5, 7, 47], or merely use entity embeddings for feature extraction and rely on other modules for ED [7, 38, 39]. They do not fully exploit the structural information in text snippets and KBs.

Recently, graph representation learning has emerged as an effective approach to learn vector representations for graph-structured data. Graph Neural Networks (GNNs) [16, 20, 46] have shown promising results in various representation learning tasks on KBs, including link prediction, node classification, as well as node clustering. The foundation of GNNs is a powerful spatial invariant aggregation function that learns how to aggregate rich structural and semantic information from each node's neighborhood to generate node embeddings. Motivated by the observation that entity mentions in a text snippet are likely to share similar or relevant context, we represent these entity mentions as a query graph to capture their interdependence. Then, we model ED as a graph matching problem and propose a simple architecture, ED-GNN, which not only collectively learns the contextual information and structural interdependence of entity mentions in the given text snippets, but also captures discriminative contextual information of entities in a medical KB. We target the medical domain because medical KBs contain deep and fine-grained knowledge, which is reflected by their rich hierarchical structure and vocabularies that can be utilized by our ED-GNN. Note that ED-GNN could be applied to other domain-specific or cross-domain KBs as well, if they contain similar contextual or structural characteristics as the medical ones.

We propose two optimizations for ED-GNN to further improve its disambiguation capability. First, after constructing a *query graph* (representing the entity mentions in a text snippet), ED-GNN augments this graph with domain knowledge from the medical KB. Consider the text snippet *"Aspirin can cause nausea indicating a potential ARF, nephrotoxicity, and proteinuria"*. The abbreviation *"ARF"* is a mention that could refer to the entities *"acute renal failure"* or *"acute respiratory failure"* in the medical KB. Leveraging the domain knowledge from the medical KB (i.e., that *"nephrotoxicity"* and *"proteinuria"* are adverse effects of Aspirin), ED-GNN understands that *"ARF"* is in the context of Aspirin's adverse effects. Hence, *"acute renal failure"* is identified as the matching entity, even though the abbreviation of *"acute respiratory failure"* is also *"ARF"*.

Second, ED-GNN is equipped with an effective negative sampling strategy, which challenges ED-GNN to learn from *difficult* samples to improve the model's disambiguation capability. Assume that we have picked up (*"ARF"*, *"acute renal failure"*) as a positive training example. Following convention [53], we sample negative examples by replacing *"acute renal failure"* from the above positive example. Then (*"ARF"*, *"chronic renal failure"*) is a difficult negative sample as the lexical similarity between *"chronic renal failure"* and *"acute renal failure"* is high. Another difficult negative sample can be (*"ARF"*, *"gastroenteritis"*) since *"gastroenteritis"* shares several common neighbors with *"acute renal failure"* in the medical KB. ED-GNN can more effectively learn from the above negative samples to reach the desired accuracy, compared to the commonly used random negative sampling [20] that replaces *"acute renal failure"* with a random entity (e.g., *"fever"*) in the medical KB.

Contributions. We highlight our main contributions as follows:

- We present ED-GNN, a novel medical ED solution, based on graph neural networks (GNNs) such as GraphSAGE [16], R-GCN [37], and MAGNN [12]. We model ED as a graph matching problem to leverage such GNNs with a simple architecture.
- We develop two optimization techniques to further improve ED-GNN's disambiguation capability. First, we construct the query graph and augment it with domain knowledge from the medical KB. This helps ED-GNN focus on the right structural information from the query graph for making the matching decisions. Second, we design an effective negative sampling strategy, which provides ED-GNN with harder examples, resulting in more discriminative power for entity disambiguation.
- We evaluate the effectiveness of ED-GNN on multiple real-world datasets. Our experimental results show that ED-GNN consistently outperforms the state-of-the-art ED solutions in all datasets by up to 16.4% in F1 score. Furthermore, we evaluate the two optimization techniques in ED-GNN and show that both of them lead to performance improvements.

Outline. The rest of the paper is organized as follows. Section 2 introduces the basic notation, briefly describes a family of GNNs, and overviews the architecture of ED-GNN. Section 3 describes the two optimization techniques designed for ED-GNN. We present our experiments in Section 4, review related work in Section 5, and conclude in Section 6.

2 BACKGROUND AND ARCHITECTURE

Definition 2.1. (Heterogeneous Graph) We define a heterogeneous graph as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ associated with a node type mapping function $\phi : \mathcal{V} \mapsto \mathcal{T}$ and an edge type mapping function $\psi : \mathcal{E} \mapsto \mathcal{R}$, where $\mathcal{T}$ and $\mathcal{R}$ denote the sets of node types and edge types, respectively, with $|\mathcal{T}| + |\mathcal{R}| > 2$.

Figure 1 shows a toy example of a heterogeneous graph constructed from a medical KB. The node types are Drug (blue nodes), AdverseEffect (green nodes), Symptom (purple nodes), and Finding (orange nodes). The edge types are TREAT, CAUSE, INDICATE, as well as HAS. Besides, all these nodes are associated with descriptions (e.g., Aspirin, headache, nausea, and fever). In this work, we model both a medical KB and a text snippet as heterogeneous graphs, such that we cast medical ED as a binary classification problem using the expressive power of heterogeneous GNNs.

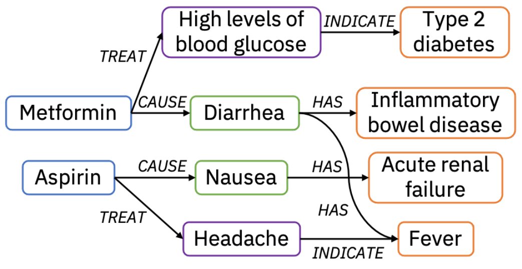


Figure 1: A toy heterogeneous graph (best viewed in color).

Definition 2.2. (Heterogeneous Graph Embedding) Given a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with node attribute matrices $A_{T_i} \in \mathbb{R}^{|\mathcal{V}_{T_i}| \times d_{T_i}}$ for node types $T_i \in \mathcal{T}$, a heterogeneous graph embedding is a d-dimensional node representation (a.k.a. embedding) for all $v \in \mathcal{V}$ with $d \ll |\mathcal{V}|$, which captures the network structural and semantic information in $\mathcal{G}$.

2.1 Graph Neural Networks

In recent years, Graph Neural Networks (GNNs) have been intensively studied and shown effective for various graph mining and analytical tasks, including node classification, link predication, and graph matching. Their ability to combine structural information and semantic features is essential to our ED task. In ED-GNN, we employ three representative approaches, including GraphSAGE [16], R-GCN [37] and MAGNN [12]. GraphSAGE is a seminal message-passing GNN, which employs the general notion of aggregator functions for efficient generation of node embeddings. R-GCN is a

Table 1: Table of notations.

Notation	Description
$\mathcal{G}$	Heterogeneous graph
$\mathcal{G}_{ref}$	Knowledge base (reference graph)
$\mathcal{V}_{ref}$	The set of nodes in $\mathcal{G}_{ref}$
$\mathcal{E}_{ref}$	The set of edges in $\mathcal{G}_{ref}$
v^r	A node in $\mathcal{V}_{ref}$
$\mathcal{G}_{qry}$	Query graph
$\mathcal{V}_{qry}$	The set of nodes in $\mathcal{G}_{qry}$
$\mathcal{E}_{qry}$	The set of edges in $\mathcal{G}_{qry}$
v^q	A node in $\mathcal{V}_{qry}$
$\mathbf{h}_v^{\text{attr}}$	Initial node feature
$\mathbf{h}_v$	Hidden state (embedding) of node v
P	A metapath
$\mathcal{P}$	The set of metapaths $\{P_1, P_2, \cdots, P_M\}$
$P(u, v)$	A metapath instance connecting nodes u and v
$\mathcal{N}_v$	The set of neighbors of node v
$\mathcal{N}_v^P$	The set of neighbors of node v based on P

relation-aware graph convolutional network which handles k-hop message-passing over heterogeneous KBs. MAGNN is the state-of-the-art metapath-based GNN that supports heterogeneous KBs and learns subtle contextual structures in KBs using semantic-aware neighbor aggregation with composite relations. All three GNNs are implemented using Deep Graph Library [43] on top of PyTorch [33]. This makes ED-GNN lightweight and easy to adapt to new KBs. Note that other GNNs can be plugged into our architecture as well. Table 1 summarizes the notations used in these three GNNs.

GraphSAGE. GraphSAGE [16] leverages node features (e.g., text descriptions/labels associated with nodes) in order to learn an embedding function that generalizes to unseen nodes. By incorporating node features, GraphSAGE simultaneously learns the topological structure of each node's neighborhood as well as the distribution of node features in the neighborhood. Formally, the k-th layer of GraphSAGE is:

$$\begin{aligned} \mathbf{h}_{\mathcal{N}_v}^k &= \text{AGGREGATE}(\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}_v), \\ \mathbf{h}_v^k &= \sigma(\mathbf{W}^k \cdot [\mathbf{h}_v^{k-1} || \mathbf{h}_{\mathcal{N}_v}^k]), \end{aligned} \tag{1}$$

where σ is an activation function and $\mathbf{W}^k$ is a set of weight matrices, $\forall k \in \{1, ..., K\}$, which are used to propagate information between different layers of the model. The intuition behind Equation 1 is that at each layer, nodes aggregate information from their local neighbors, and as this process iterates, nodes incrementally gain more and more information from further reaches of the graph.

R-GCN. Unlike GraphSAGE that only considers the node-wise connectivity in a graph and ignores edge labels such as the relations in KBs, R-GCN distinguishes different neighbors with relation-specific weight matrices. In the k-th convolutional layer, each representation vector is updated by accumulating the vectors of neighboring nodes through a normalized sum:

$$\mathbf{h}_v^{(k)} = \sigma(\mathbf{W}_0^k \mathbf{h}_v^{k-1} + \sum_{r \in \mathcal{R}} \sum_{u \in \mathcal{N}_v^r} \frac{1}{c_{v,r}} \mathbf{W}_r^k \mathbf{h}_u^{k-1}), \tag{2}$$

where $\mathbf{W}_0^k$ is the weight matrix for the node itself, $\mathbf{W}_r^k$ is used specifically for the neighbors having relation r, i.e., $\mathcal{N}_v^r$, $\mathcal{R}$ is the relation set, and $c_{v,r}$ is used for normalization. Intuitively, different edge types use different weights and only edges of the same relation type r are associated with the same projection weight $\mathbf{W}_r^k$.

MAGNN. MAGNN aggregates a node v's representation from $\mathcal{N}_v^{\mathcal{P}}$ (i.e., the metapath-aware neighborhood) and the nodes in between, by encoding the metapath instances through a relational rotation encoder. To elaborate, we introduce the following definitions from [12].

Definition 2.3. (Metapath) A metapath P in a heterogeneous graph $\mathcal{G}$ is a path in the form of $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \cdots \xrightarrow{R_m} A_{m+1}$ (abbreviated as $A_1A_2 \cdots A_{m+1}$), where A and R are node types and edge types in $\mathcal{G}$, respectively.

Definition 2.4. (Metapath-based Neighbors) Given a metapath P of a heterogeneous graph, the metapath-based neighbors $\mathcal{N}_v^P$ of a node v are defined as the set of nodes that connect with node v via metapath instances of P.

For example, Drug-AdverseEffect-Finding (DAF) is a metapath representing that drugs cause adverse effects, and these adverse effects can be described by findings. Given the metapath DAF, *"Fever"* and *"Diarrhea"* constitute the metapath-based neighbors of *"Metformin"* in Figure 1. These nodes are connected with *"Metformin"* via the metapath instance *"Metformin-Diarrhea-Fever"*[1].

As defined in [12], during the intra-metapath aggregation, each target node extracts and combines information from the metapath instances connecting the node with its metapath-based neighbors. The intra-metapath aggregation layer is formally defined as:

$$\begin{aligned} e_{vu}^P &= \text{LeakyReLU}(a_P^\intercal \cdot [\mathbf{h}_v || \mathbf{h}_{P(u,v)}]), \\ \alpha_{vu}^P &= \frac{\exp(e_{vu}^P)}{\sum_{s \in \mathcal{N}_v^P} \exp(e_{vs}^P)}, \\ \mathbf{h}_v^P &= \sigma(\sum_{u \in \mathcal{N}_v^P} \alpha_{vu}^P \cdot \mathbf{h}_{P(v,u)}), \end{aligned} \tag{3}$$

where $\mathbf{h}_{P(u,v)}$ represents all the node features along a metapath instance, a_P is the parameterized attention vector for the metapath P, and α_{vu}^P is the normalized importance weight for all $u \in \mathcal{N}_v^P$. Finally, the intra-metapath output goes through an activation function $\sigma(\cdot)$. In this way, MAGNN captures the structural and semantic information of heterogeneous graphs from both neighbor nodes and the metapaths between the target node and its neighbors.

After aggregating the node and edge information within each metapath, MAGNN uses an inter-metapath aggregation layer with the attention mechanism to fuse latent vectors of the node v obtained from multiple metapaths into final node embeddings. The inter-metapath aggregation layer is formally defined as:

$$\begin{aligned} e_{P_i} &= q_A^\intercal \cdot s_{P_i}, \\ \beta_{P_i} &= \frac{\exp(e_{P_i})}{\sum_{P \in \mathcal{P}_A} \exp(e_P)}, \\ \mathbf{h}_v^{\mathcal{P}_A} &= \sum_{P \in \mathcal{P}_A} \beta_P \cdot \mathbf{h}_v^P, \end{aligned} \tag{4}$$

where s_{P_i} denotes the summarized metapath $P_i \in \mathcal{P}$ by averaging the transformed metapath-specific node vectors for all nodes $v \in$

[1]Note that metapath-based neighbors are not limited to 1-hop neighbors.

$\mathcal{V}_A$, q_A is the parameterized attention vector for node type A, β_{P_i} can be interpreted as the relative importance of the metapath P_i to nodes of type A, and $\mathbf{h}_v^{\mathcal{P}_A}$ represents the final node embedding of v, namely a weighted sum of all metapath-specific node vectors of v. By integrating multiple metapaths, MAGNN can learn the comprehensive semantics ingrained in the heterogeneous graph.

2.2 ED-GNN Architecture

We now present an overview of ED-GNN (depicted in Figure 2) for medical entity disambiguation. The basic idea is to represent both a medical KB and a given text snippet as heterogeneous graphs $\mathcal{G}_{ref}$ and $\mathcal{G}_{qry}$, respectively. Following the property graph model [8], we assume that nodes are associated with literal attributes in both $\mathcal{G}_{ref}$ and $\mathcal{G}_{qry}$, where nodes and edges have different types. In $\mathcal{G}_{ref}$, nodes correspond to medical entities and edges correspond to relationships between those entities. The entity mentions and extracted relations from the text snippets are represented as nodes and edges in $\mathcal{G}_{qry}$. Section 3.1 describes the optimized query graph modeling in further details.

Medical KBs are often curated and updated from text corpora in medical literature. The text snippets are collected from these text corpora as well. Hence, the neighboring structures of $\mathcal{G}_{ref}$ and $\mathcal{G}_{qry}$ are expected to be similar. Inspired by Siamese networks [26], ED-GNN uses two identical graph neural networks (one of GraphSAGE, R-GCN, or MAGNN) to generate the graph embeddings that encode all local structural information centered around the nodes in $\mathcal{G}_{qry}$ and $\mathcal{G}_{ref}$, respectively. These two GNNs share the same parameters (i.e., weight matrices) and consume a node list and an edge list from both $\mathcal{G}_{ref}$ and $\mathcal{G}_{qry}$, respectively. In a node list, each row contains a node id, its attribute features, and its type. In an edge list, each row has a source node id (head), a destination node id (tail), and the edge type. More details can be found in [43].

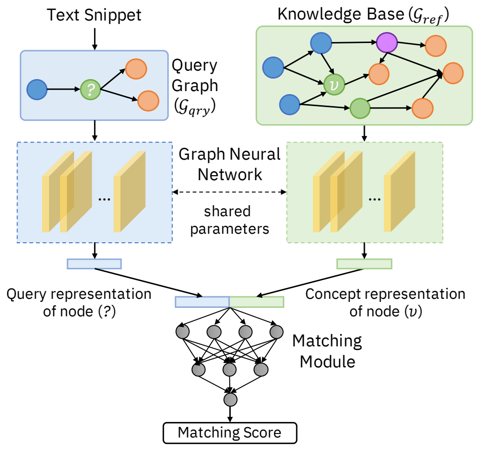

Text Snippet
Knowledge Base ($\mathcal{G}_{ref}$)
Query Graph ($\mathcal{G}_{qry}$)
?
v
Graph Neural Network
shared parameters
Query representation of node (?)
Concept representation of node (v)
Matching Module
Matching Score

Figure 2: ED-GNN architecture (best viewed in color).

Model Training. ED-GNN learns the representation of each node (node 'v' in Figure 2) in $\mathcal{G}_{ref}$ based on its k-hop or metapath-based neighbors and the representation of the query concept (node '?' in Figure 2) in $\mathcal{G}_{qry}$. Such representation captures not only the node features, but also the topological structure of each node's neighborhood in $\mathcal{G}_{ref}$ or $\mathcal{G}_{qry}$. The matching module calculates their matching score, indicating the likelihood of two nodes matching each other. The matching module can be a multi-layer perceptron with one hidden layer, a log-bilinear model, or simply a dot product. We optimize the model weights by minimizing the following loss function through negative sampling:

$$\mathcal{L} = -\sum_{(u,v)\in\Omega} log(\sigma(\boldsymbol{h}_u^\top \boldsymbol{h}_v)) - \sum_{(u,v)\in\Omega^-} log(\sigma(\boldsymbol{h}_u^\top \boldsymbol{h}_{v'})), \tag{5}$$

where $\sigma(\cdot)$ is the sigmoid function, Ω is the set of observed (positive) node pairs, and Ω^- is the set of negative node pairs sampled from all unobserved node pairs. In our entity disambiguation scenario, a positive node pair consists of one node representing an ambiguous entity in the text snippet and one node representing its corresponding matching node in the medical KB, respectively. By default, ED-GNN adopts uniform negative sampling by corrupting one node in the positive node pairs, due to its simplicity and efficiency. An optimized negative sampling strategy is introduced in Section 3.2. The above loss is the cross entropy of classifying the positive pair correctly.

3 OPTIMIZATIONS IN ED-GNN

3.1 Semantic Augmentation for Query Graph

Our first optimization allows domain knowledge from the medical KB to be injected into the query graph $\mathcal{G}_{qry}$ through processing the text snippet to emphasize critical information for entity disambiguation. This processing step includes entity mention extraction and query graph construction.

Augment Entity Mentions with Node Types from $\mathcal{G}_{ref}$. To extract entity mentions from an input text snippet, i.e., named entity recognition (NER), many existing methods are available, including Stanford CoreNLP [28], AllenNLP [13], and PyText [23]. In this work, we choose BioBERT [24], a deep learning-based clinical NER model, fine-tuned on the medical KB. Consider the text snippet in Figure 3: *"Aspirin can cause nausea indicating a potential ARF, nephrotoxicity, and proteinuria"*. In this sentence, we can identify the following terms as entity mentions of medical entities: "*Aspirin*", "*nausea*", "*ARF*", "*nephrotoxicity*" and "*proteinuria*".

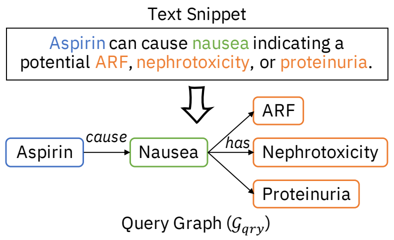

Text Snippet
Aspirin can cause nausea indicating a potential ARF, nephrotoxicity, or proteinuria.
Aspirin
cause
Nausea
has
ARF
Nephrotoxicity
Proteinuria
Query Graph ($\mathcal{G}_{qry}$)

Figure 3: Text snippet to query graph (best viewed in color).

Having entity mentions detected, we try to match them with the nodes in the medical knowledge base $\mathcal{G}_{ref}$. We exploit an inverted index of the entities in $\mathcal{G}_{ref}$ for the matching. Such inverted index includes not only the exact matches of these entities, but also synonyms, acronyms, and abbreviations of the entities in $\mathcal{G}_{ref}$. For the matched entity mentions, we further infer their entity types based on their corresponding entities in $\mathcal{G}_{ref}$. For example, we identify "*aspirin*" as an instance of Drug, "*nausea*" as an instance of Adverse-Effect, and "*nephrotoxicity*" as well as "*proteinuria*" as instances of Finding in $\mathcal{G}_{ref}$. These identified entities can help us disambiguate the remaining entity mentions (e.g., "*ARF*"), for which a match is not found. Then, these identified entity mentions are used as the node set in the query graph $\mathcal{G}_{qry}$. It is possible that an entity mention has multiple matches in $\mathcal{G}_{ref}$. In this case, we associate all entity types of these matches to the entity mention.

Augment Relationships in $\mathcal{G}_{qry}$. One can create a query graph by connecting each node pair with an edge (self-loops are also added in this process) [3, 48]. The resulting query graph can be considered as an undirected graph that describes the dependencies between entity mentions. However, such approach fails to utilize the domain knowledge from the medical knowledge base. Namely, the constructed query graph does not capture different relationships between a pair of entities, which provide critical contextual information to entity disambiguation.

To address this issue, we leverage the domain knowledge from $\mathcal{G}_{ref}$ to augment the query graph $\mathcal{G}_{qry}$. Specifically, we introduce an edge between a pair of nodes u^q and v^q (i.e., entity mentions) in $\mathcal{G}_{qry}$, if there exist two nodes u^r and v^r in $\mathcal{G}_{ref}$, such that u^q matches u^r, v^q matches v^r, and there exists an edge between u^r and v^r in $\mathcal{G}_{ref}$. The type of the newly added edge can be inferred from the corresponding edge in $\mathcal{G}_{ref}$ as well. To continue the above example, the nodes "*Aspirin*" and "*nausea*" are connected by an edge of type CAUSE in $\mathcal{G}_{ref}$ (shown in Figure 1). Hence the newly added edge in $\mathcal{G}_{qry}$ is of type CAUSE as well. For those entity mentions (e.g., "*ARF*") that do not have their matches in $\mathcal{G}_{ref}$, we rely on entity types obtained from NER to find the corresponding node type in $\mathcal{G}_{ref}$ and further identify the edges associated with the node type. Subsequently, we add an edge between the unknown entity and the existing entities if the corresponding node types are connected in $\mathcal{G}_{ref}$. This newly added edge in $\mathcal{G}_{qry}$ is also augmented with the corresponding edge type information from $\mathcal{G}_{ref}$. The overall query graph augmentation method is presented in Algorithm 1.

3.2 Semantic-Driven Negative Sampling

Negative sampling is used in our loss function (Equation 5) as an approximation of the normalization factor of edge likelihood [29]. Random negative sampling is commonly adopted in graph representation learning [16] due to its simplicity and efficiency. However, most of the negative samples are trivial cases from which the model does not gain much discriminative power [53]. Generative adversarial network (GAN), has been introduced in negative sampling [44] to avoid the problem of vanishing gradient and thus to obtain better performance. However, using GAN increases the number of training parameters and leads to instability and degeneracy [53].

To solve the above issues, for every positive training example, we provide *difficult* negative examples for our ED-GNN to learn. Intuitively, these negative examples are very close to the positive entity in the embedding space either due to their lexical or structural features. Hence, we generate them by utilizing two different sources of similarity evidence, which emphasize on both semantic and structural relatedness between positive and negative examples.

Algorithm 1 Query Graph Augmentation

```
Input: A knowledge graph $\mathcal{G}_{ref}$, a text snippet $T$
Output: A query graph $\mathcal{G}_{qry}(\mathcal{V}_{qry}, \mathcal{E}_{qry})$
1:  $\mathcal{G}_{qry} \leftarrow \emptyset$
2:  $EM \leftarrow$ NER($T$) //get all entity mentions
3:  $EM_{match} \leftarrow$ match($EM$, $\mathcal{G}_{ref}$) //get matching entity mentions
4:  $EM_{unknown} \leftarrow EM \setminus EM_{match}$
5:  $\mathcal{V}_{qry}$.addNode($EM_{match}$)
6:  for each pair of nodes $u^q, v^q \in \mathcal{V}_{qry}$ do
7:      $u^r \leftarrow EM_{match}$.getMatch($u^q$)
8:      $v^r \leftarrow EM_{match}$.getMatch($v^q$)
9:      if $e = (u^r, v^r) \in \mathcal{E}_{ref}$ then
10:         $\mathcal{E}_{qry}$.addEdge($u^q$, $v^q$, $e.type$)
11: for each $em \in EM_{unknown}$ do
12:     $et \leftarrow em$.getEntityType()
13:     $EdgeTypeSet \leftarrow \mathcal{G}_{ref}$.getEdgeTypes($et$)
14:     $EntityTypeSet \leftarrow \mathcal{G}_{ref}$.getEntity($EdgeTypeSet$)
15:     $\mathcal{V}_{qry}$.addNode($em$) //add $em$ to $\mathcal{G}_{qry}$
16:     $u^q \leftarrow em$
17:     for each $v^q \in \mathcal{V}_{qry}$ and $v^q \neq u^q$ do
18:         if $v^q$.getEntityType() $\in EntityTypeSet$ then
19:             $edgeType \leftarrow EdgeTypeSet$.get($v^q$.getEntityType(), $et$)
20:             $\mathcal{E}_{qry}$.addEdge($u^q$, $v^q$, $edgeType$)
21: return $\mathcal{G}_{qry}$
```

Semantic Similarity. Difficult negative examples should be semantically similar to the positive entity in $\mathcal{G}_{ref}$. For example, a positive node pair is (*"MH"*, *"Malignant hyperpyrexia"*), in which *"MH"* is the ambiguous entity mention in $\mathcal{G}_{qry}$ and *"Malignant hyperpyrexia"* is the labeled positive entity in $\mathcal{G}_{ref}$. Then, (*"MH"*, *"Malignant hyperthermia"*) can be considered as a difficult negative example since the semantic similarity between these two entities is very high. To find such negative examples, we reuse the initial node (i.e., entity) embeddings in $\mathcal{G}_{ref}$ and compute the cosine similarity between each positive example and other entities in $\mathcal{G}_{ref}$. Note that these initial node embeddings can be obtained using language models such as BERT [9] on each node in both $\mathcal{G}_{qry}$ and $\mathcal{G}_{ref}$.

Structural Similarity. Difficult negative examples should also share many common neighbors with the positive entity in $\mathcal{G}_{ref}$. Intuitively, two entities are similar if they are related to similar entities. Different graph similarity metrics are defined, ranging from graph edit distance (GED) [1], maximum common subgraph [2], to graph kernels [14]. In this work, we choose the commonly used GED to compute the structural similarity of two entities in $\mathcal{G}_{ref}$. Only the local (i.e., 1-hop) neighbors of an entity are used in GED, which substantially reduces the computational cost. Our choice aligns well with the observation that 1-hop neighbors provide the most significant structural information in terms of a node representation.

We integrate the above two measures into the scoring function: $sim = sim_{se} \cdot sim_{st}$, where sim_{se} is the cosine similarity between two entity embeddings and sim_{st} is the normalized GED according to [34]. The resulting similarity score is in the range of [0, 1]. Before training, negative examples are generated by ranking entities in $\mathcal{G}_{ref}$ according to their similarity scores with respect to the ambiguous entities in the labeled training set. The top-ranked examples are randomly sampled. As a result, the hard negative examples are more similar to the query than random negative examples, thus forcing the model to learn to disambiguate entities at a finer granularity. To reduce the computational cost, we only consider the immediate neighbors of an entity in the positive example as candidates for negative examples. These negative examples are guaranteed to be negative, since the KB is a complete graph (no missing nodes/edges) and only one entity matches the ambiguous mention. This is different from link prediction, where a missing positive link can be falsely selected as a negative example.

During training, we adopt a curriculum training scheme [50] where ED-GNN will learn from easy negative examples first, but then gradually focus on difficult ones. Specifically, no difficult examples are used in the first epoch of training such that our ED-GNN can quickly find an area in the parameter space where the loss is relatively small. We then add difficult negative examples in subsequent epochs, focusing the model to learn how to disambiguate highly related entities from only slightly related ones.

4 EXPERIMENTAL EVALUATION

4.1 Datasets

We use the following datasets from the medical domain as heterogeneous graphs to evaluate the performance of our method. Each dataset is used as a KB by itself. There is only one mention to be disambiguated in each text snippet, and the goal is to find its corresponding entity in the KB. Simple statistics of the KBs corresponding to these datasets are summarized in Table 2.

- MDX is a medical KB[2] that contains information about drugs, adverse effects, indications, findings, etc. It is manually curated from medical literature by editorial staff, and the text snippets are extracted from the literature as well. The ground truth for MDX is provided by the editorial staff.
- MIMIC-III [18] is a public data set containing 40,000 anonymized patient health-related records. It includes information such as demographics, laboratory test results, medications, and diagnoses.
- Bio CDR [25] consists of 1,500 PubMed abstracts annotated with mentions of chemicals, diseases, and relations between them.
- NCBI [10] consists of 700 PubMed[3] abstracts annotated with disease mentions and their corresponding concepts in MeSH[4].
- ShARe [32] comprises 433 anonymized clinical notes (400 training and 133 test), obtained from the MIMIC II[5] clinical dataset and annotated with disorder mentions.

In public datasets, ground truths are provided in the following form: `“Text”: “A common human skin tumour is caused by activating mutations.”, “Mentions”: [{“mention”: “skin tumor”,“start_offset”:15, “end_offset”:26, “category”: “Disease”, “link_id”:“C0037286”}]`. In this case, *skin tumor* is the ambiguous mention and its corresponding entity in the KB is *neoplasm of the skin*, which is represented by the concept unique identifier `C0037286` in the medical ontologies (UMLS, MeSH, etc).

Each dataset is split into training (70%), validation (15%), and testing (15%) sets unless otherwise stated. For NCBI, it is split into a training set of 500 abstracts, a validation and a test set of 100 abstracts each. For Bio CDR, it comes with a training set of 1000 and a test set of 500 abstracts. We further split its training set into a training and a validation set of 800 and 200 abstracts. For ED-GNN variants, we add the same number of negative node pairs described in Section 3.2 to the validation and testing sets. These negative samples purposely cover different cases (e.g., abbreviation, synonym, acronym, and simplification).

[2]https://www.ibm.com/products/micromedex-with-watson
[3]https://pubmed.ncbi.nlm.nih.gov/
[4]https://meshb.nlm.nih.gov/search
[5]https://archive.physionet.org/mimic2/

Table 2: Dataset statistics.

Dataset	MDX	MIMIC-III	NCBI	ShARe	Bio CDR
# Nodes	35,028	22,642	753	1,719	1,082
# Edges	74,621	284,542	1,845	12,731	2,857

4.2 Systems

We evaluate our approach ED-GNN using three different GNNs: GraphSAGE [16], R-GCN [37], and MAGNN [12]. We also compare ED-GNN with the state-of-the-art methods DeepMatcher [30], NormCo [47], and NCEL [3], which are briefly described below.

- ED-GNN (GraphSAGE) employs GraphSAGE, designed for homogeneous graphs. It models the graph topology through neighbors aggregation on the node attributes.
- ED-GNN (R-GCN) leverages R-GCN, which handles different relationships between entities in a KB. It learns multiple convolution matrices corresponding to different edge types.
- ED-GNN (MAGNN) adopts MAGNN, which learns the representation of nodes based on their metapath-based neighbors with attention mechanisms at both node and semantic levels.
- DeepMatcher is a supervised deep learning solution designed for entity resolution in a tabular setting. In our setting, an input to DeepMatcher is a tuple containing an ambiguous mention (e.g., *skin tumor*) from a text snippet and an entity (e.g., *neoplasm of the skin*) in the KB. We train and evaluate DeepMatcher with positive and negative tuples. Although the structural information from text snippets and KBs is not available to DeepMatcher, we choose it as an exemplar RNN method focusing on matching entities.
- NormCo uses a deep coherence model for disease entity normalization, which considers the semantics of an entity mention and the topical coherence of the mentions within a text snippet.
- NCEL creates a graph for candidates of mentions and then apply GCN to improve the disambiguation by directly aggregating information from linked nodes.

Implementation Details. For the baseline systems (i.e., DeepMatcher, NormCo, and NCEL), we use the original hyper-parameter settings described in their papers, respectively. For DeepMatcher, we select its attention model since it has been shown effective on textual entity matching tasks in [30]. For all ED-GNN variations, we employ the Adam [19] optimizer with the learning rate set to 0.001, the weight decay set to 0.001, and dropout rate to 0.5. We use the same splits of training, validation, and testing data sets for all models, and train the GNNs for 100 epochs and apply early stopping with a patience of 30. For ED-GNN using R-GCN and MAGNN, we set the dimension of the attention vector to 128. For ED-GNN using MAGNN, we set the number of attention heads to 2; we set the

Table 3: Results of entity disambiguation on five datasets.

Methods	DeepMatcher			NormCo			NCEL		
Datasets	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
MDX	0.656	0.700	0.677	0.687	0.634	0.659	0.673	0.659	0.666
MIMIC-III	0.708	0.567	0.630	0.747	0.692	0.718	0.716	0.624	0.667
NCBI	0.783	0.815	0.799	0.863	0.818	0.840	0.816	0.793	0.804
ShARe	0.694	0.639	0.665	0.726	0.623	0.671	0.753	0.631	0.687
Bio CDR	0.837	0.816	0.826	0.866	0.805	0.834	0.857	0.829	0.843
Methods	ED-GNN (GraphSAGE)			ED-GNN (R-GCN)			ED-GNN (MAGNN)		
Datasets	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
MDX	0.614	0.900	0.730	0.722	0.867	0.788	**0.725**	**0.967**	**0.829**
MIMIC-III	0.786	**0.733**	**0.759**	0.810	0.567	0.667	**0.826**	0.633	0.717
NCBI	**0.924**	**0.856**	**0.889**	0.912	0.823	0.865	0.915	0.861	0.887
ShARe	0.794	0.829	0.811	0.806	0.833	0.819	**0.825**	**0.879**	**0.851**
Bio CDR	0.853	0.845	0.849	**0.896**	**0.867**	**0.881**	0.864	0.853	0.858

dimension of the attention vector in metapath aggregation to 128. For a fair comparison, we set the embedding dimension to 128 for all the above methods.

4.3 Main Results

We measure the performance of all methods using precision, recall, and F1, which are typical metrics for the evaluation of the entity disambiguation task [3, 47]. We report the average measurements of all methods on the test set for 100 repetitions. Table 3 reports the results of ED-GNN and other methods on all five datasets. The major findings are summarized as follows:

- Our ED-GNN variants consistently outperform other solutions in terms of precision, recall, and F1 on all datasets. The best performing ED-GNN variant offers an average improvement of 7.3% in terms of F1 score, compared to the other best performing solutions. Among five datasets, we observe that all models perform better on NCBI and Bio CDR. The reason is that the graph complexity and semantic richness of NCBI and Bio CDR are simpler than the other datasets. The gain is much more significant on MDX (15.2%) and ShARe (16.4%) datasets. This fact manifests the expressive capability of our ED-GNN method to capture rich graph structures from both text snippets and KBs in medical entity disambiguation.
- Among all three ED-GNN variants, ED-GNN (MAGNN) achieves the highest average F1 score on all datasets, despite ED-GNN (GraphSAGE) and ED-GNN (R-GCN) achieve the best performance on MIMIC-III, NCBI, and Bio CDR datasets respectively. It is worth noting that ED-GNN (MAGNN) offers an average improvement of 2.1% and 2.4%, in terms of F1 score compared to ED-GNN (GraphSAGE) and ED-GNN (R-GCN), respectively. The results show that ED-GNN (MAGNN) captures both semantic and structural features by aggregating specific type of neighbors in the KBs, improving the performance of medical entity disambiguation. The other two ED-GNN variants deliver the best results on NCBI and Bio CDR datasets respectively as the complexities of these two datasets are less than the other ones.
- Regarding the use of various graph features, DeepMatcher and NormCo only uses the text attributes of the compared entities, missing the opportunities to leverage more contextual information available in the graphs. NCEL incorporates GCN into its neural network to utilize only a subset of nodes next to the entity mentions but does not take edge types into consideration. ED-GNN (GraphSAGE) does not differentiate the contextual information aggregated via different edge types neither. This can be problematic when information gathered via certain edge types are not equally important. ED-GNN (R-GCN) tackles this issue by introducing an edge-aware aggregation function. ED-GNN (MAGNN) shows the expressive power provided by the metapath-based aggregation to explore the rich structural and semantic information in a KB, which eventually results in the best all-around performance.

4.4 ED-GNN Model Studies

Optimizations in ED-GNN. To make an ablation study on ED-GNN, we first evaluate the performance of our basic ED-GNN without two optimization techniques introduced in Section 3, ED-GNN with semantic augmentation for query graph, and ED-GNN with semantic-driven negative sampling. For each dataset, we choose the best performing ED-GNN variant from Table 3. The major findings are summarized from Table 4.

We observe that the semantic-driven negative sampling improves the basic ED-GNN (GraphSAGE) by 3.5% and 4.5% in terms of F1 score on MIMIC-III and NCBI, respectively. The query graph augmentation does not help at all in this case as GraphSAGE is not a relation-aware GNN. Similarly, ED-GNN (MAGNN) benefits more from the semantic-driven negative sampling strategy on MDX (+6.4%). On the other hand, the query graph augmentation is more effective on BioCDR and ShARe datasets. Compared to the basic ED-GNN, the improvements are 3.3% and 4.3%, respectively. The reason is that the additional semantic information from the augmented query graph is more representative when the KB is simple.

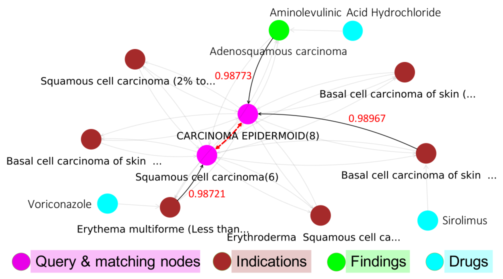
These observations demonstrate that the query graph augmented with domain knowledge from the medical KB helps ED-GNN focus on the right structural information when making the matching decision. The semantic-driven negative sampling strategy, on the other hand, provides ED-GNN with harder examples, resulting in

Table 4: Results of two optimization techniques on ED-GNN.

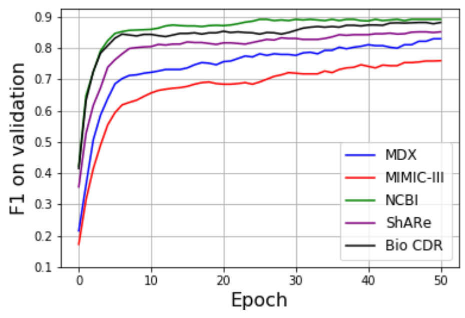
Methods	Datasets	Basic			Query graph augmentation			Negative sampling		
		Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
ED-GNN (GraphSAGE)	MIMIC-III	0.747	0.702	0.724	0.747	0.702	0.724	**0.786**	**0.733**	**0.759**
	NCBI	0.869	0.821	0.844	0.869	0.821	0.844	**0.924**	**0.856**	**0.889**
ED-GNN (R-GCN)	Bio CDR	0.825	0.798	0.811	**0.863**	**0.826**	**0.844**	0.846	0.805	0.825
ED-GNN (MAGNN)	MDX	0.671	0.827	0.741	0.694	0.863	0.769	**0.713**	**0.925**	**0.805**
	ShARe	0.754	0.824	0.787	0.796	**0.868**	**0.830**	**0.813**	0.842	0.827

more discriminative power for entity disambiguation. Together, two optimization techniques improve the ED-GNN's disambiguation capability across a variety of medical datasets.

Furthermore, we employ GNN-Explainer [51] to visualize the important contributions of nodes and edges in KBs when finding the matching entity for the ambiguous mention. Due to the space constraint, we show one example using MDX dataset in Figure 4(a). GNN-Explainer highlights 3 most important (score range [0,1]) edges that contribute the most to matching "squamous cell carcinoma" with "carcinoma epidermoid" by ED-GNN. These edges carry critical information from different types of neighboring nodes, including "adenosquamous carcinoma" (Findings), "basal cell carcinoma of skin" (Indication), and "erythema multiforme (less than 10% epidermal detachment)" (Indication). This indicates that our ED-GNN can learn and leverage the most semantically and structurally meaningful information among different types of entities and relations for entity disambiguation.


Aminolevulinic Acid Hydrochloride
Adenosquamous carcinoma
Squamous cell carcinoma (2% to...
0.98773
Basal cell carcinoma of skin (...
0.98967
CARCINOMA EPIDERMOID(8)
Basal cell carcinoma of skin ...
Squamous cell carcinoma(6)
Basal cell carcinoma of skin ...
0.98721
Voriconazole
Sirolimus
Erythema multiforme (Less than...
Erythroderma
Squamous cell ca...
Query & matching nodes
Indications
Findings
Drugs

(a) Visualization in MDX



(b) Convergence

Figure 4: Model analysis (best viewed in color).

Convergence Analysis. We analyze the convergence properties of ED-GNN, using the best performing ED-GNN variant from Table 3 for each dataset. The results, as shown in Figure 4(b), demonstrate that ED-GNN converges fast and achieves robust performance across all real-world datasets.

Number of Layers in ED-GNN. We also analyze the results of ED-GNN with 1 to 4 graph layers on all five datasets. Again, we choose the best performing ED-GNN variant from Table 3 for each dataset. In Table 5, we observe that the optimal number of graph layers is 2 (for NCBI) or 3 (for MDX, MIMI-III, ShARe, and Bio CDR). When ED-GNN uses more than 3 layers, its performance declines. Although more layers allow ED-GNN to indirectly capture more distant neighborhood information by layer-to-layer propagation, such distant neighbors would introduce much noise and lead to more non-isomorphic neighborhood structures between the query graph and the KB.

Table 5: Number of layers (F1).

# layers	MDX	MIMIC-III	NCBI	ShARe	Bio CDR
1	0.691	0.641	0.815	0.731	0.785
2	0.751	0.704	**0.891**	0.825	0.843
3	**0.829**	**0.759**	0.867	**0.851**	**0.881**
4	0.743	0.727	0.831	0.806	0.829

4.5 Error Analysis

We also provide an error analysis on the entity mentions that are not disambiguated correctly by ED-GNN. Table 6 breaks incorrect results in three categories below.

Table 6: Error analysis (% of each test set).

Error	MDX	MIMIC-III	NCBI	ShARe	Bio CDR
$\mathcal{G}_{qry}$ construction	9.5%	8.7%	1%	3.8%	2.2%
Insufficient structure	4.3%	9.8%	6%	3%	5.2%
Highly similar nodes	8%	4.8%	4%	3%	4.4%

Query Graph Construction Error to $\mathcal{G}_{qry}$. We observed that the semantic augmentation for query graph does not always lead to a correct query graph. The reasons are twofold. First, as described in Section 3.1, an entity mention may be associated with multiple entity types. For example, *"rash"* can be an instance of either Finding or AdverseEffect in MDX. Hence, the query graph may carry ambiguous semantic information that confuses ED-GNN. Second, multiple entity types can also lead to additional relationships in the query graph. These relationships could be irrelevant to the actual text snippet, causing ED-GNN to mismatch the ambiguous entity with incorrect entities in the KB.

Insufficient Structural Information in $\mathcal{G}_{qry}$. We observed that almost 50% of the errors are due to a lack of graph structural information from text snippets. When a text snippet is short, the constructed query graph often contains few nodes and edges. For example, in a text snippet *"Graft failure due to FSGS recurrence"* from MIMIC-III, *"Graft failure"* is the only neighbor entity of *"FSGS recurrence"*. In this case, ED-GNN does not have enough structural information to leverage, and has to primarily rely on the textual features of the ambiguous entity. Consequently, it fails to discover the corresponding entity in the KB's embedding space.

Highly Similar Nodes in $\mathcal{G}_{ref}$. At times, ED-GNN fails to identify the correct entity in the KB (e.g., MIMIC-III), even when the query graph is correctly constructed. In such cases, the entity corresponding to the ambiguous mention is often located in a highly dense area of the KB, where many semantically and structurally similar candidates exist. This essentially corresponds to the difficult negative examples described in Section 3.2. ED-GNN is not able to learn all possible negative examples through the semantic-driven negative sampling.

5 RELATED WORK

Graph Neural Networks. Graph representation learning has been shown to be extremely effective, achieving promising results in various domains over graph-structured data [16, 20, 27, 42, 49]. GCN [20] is a graph convolutional network via a localized first-order approximation of spectral graph convolutions. The seminal GNN framework, GraphSAGE [16], learns node embeddings through aggregating from a node's local neighborhood using inductive learning. Graph attention networks (GAT) [42] are introduced to learn the importance between nodes and their neighbors, and fuse the neighbors to perform node classification.

Heterogeneous graph embedding has also received much research attention recently [4, 12, 37, 46], as many KBs also fall under the general umbrella of heterogeneous graphs. For example, R-GCN [37] distinguishes different neighbors with relation-specific weight matrices. Heterogeneous graph attention network (HAN) [46] leverages a graph attention network architecture to aggregate information from the neighbors and then to combine various metapaths through the attention mechanism. Inspired by HAN, HetGNN [52] encodes the content of each node into a vector and then adopts a node type-aware aggregation function to collect information from the neighbors. HetGNN also uses attention over the node types of the neighborhood node to get the final embedding. MAGNN [12] captures all neighbor nodes and the metapath context using both intra-metapath aggregation and inter-metapath aggregation. Thus, the generated node embeddings preserve the comprehensive semantics in the heterogeneous graphs.

Entity Disambiguation. For many years, entity disambiguation (also referred to as entity linking) has been an active field of research [39]. A related task, entity matching, has also been studied extensively in the context of structured data [6, 21]. Recently, [15, 30] investigated various DL-based methods for entity matching, and concluded that although DL-based techniques do not offer significant advantages for structured data, they outperform current solutions [15] considerably for textual entity matching. DoSeR [54] relies on an RDF KB embedding [35] for KB entities using known entity links to model the context in which those entities are mentioned in the text, which can subsequently be used to predict further mentions of such entities based on the mention's context. NCEL [3] applies graph convolutional network to integrate both local contextual features and global coherence information for entity linking. However, it only considers the immediate neighbors of an entity mention and does not take edge types into consideration. COM-AID [7] introduces a composite attentional encode-decode neural network in healthcare. It encodes a concept into a vector and decodes the vector into a text snippet with the help of textual and structural contexts. NormCo [47] is designed for disease normalization. It models entity mentions using a semantic model, which consists of an entity phrase model using word embeddings and a coherence model of other disease mentions using an RNN. The final model combines both sub-models trained jointly. Unlike existing works in the field, we introduce a simple architecture that leverages state-of-the-art GNNs to encode the latent graph structure of the KB and the input text snippets for medical entity disambiguation.

6 CONCLUSION

In this paper, we study the entity disambiguation problem which plays an important role in medical knowledge graph curation and maintenance processes. We present ED-GNN, a medical entity disambiguation system, based on GNNs. ED-GNN uses a simple architecture to leverage state-of-the-art GNNs, and is further optimized by augmenting the query graph with domain knowledge from the medical KB as well as an effective negative sampling scheme to improve the disambiguation capability. The experimental results on multiple real-world medical KBs demonstrate that ED-GNN is effective and outperforms the state-of-the-art solutions.

REFERENCES

[1] H. Bunke. What is the distance between graphs. *Bulletin of the EATCS*, 20:35–39, 1983.
[2] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recogn. Lett.*, 19(3–4):255–259, 1998.
[3] Y. Cao, L. Hou, J. Li, and Z. Liu. Neural collective entity linking. In *COLING*, pages 675–686, 2018.
[4] Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou, and J. Tang. Representation learning for attributed multiplex heterogeneous network. In *SIGKDD*, page 1358–1368, 2019.
[5] A. Chisholm and B. Hachey. Entity disambiguation with web links. *TACL*, 3:145–156, 2015.
[6] P. Christen. *Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection.* Data-Centric Systems and Applications. Springer, 2012.
[7] J. Dai, M. Zhang, G. Chen, J. Fan, K. Y. Ngiam, and B. C. Ooi. Fine-grained concept linking using neural networks in healthcare. In *SIGMOD*, pages 51–66, 2018.
[8] S. Das, J. Srinivasan, M. Perry, E. I. Chong, and J. Banerjee. A tale of two graphs: Property graphs as RDF in oracle. In *EDBT*, pages 762–773, 2014.
[9] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.
[10] R. I. Dogan, R. Leaman, and Z. Lu. Ncbi disease corpus: A resource for disease name recognition and concept normalization. *Journal of Biomedical Informatics*, 47:1 – 10, 2014.
[11] M. Dredze, P. McNamee, D. Rao, A. Gerber, and T. Finin. Entity disambiguation for knowledge base population. In *COLING*, page 277–285, 2010.

[12] X. Fu, J. Zhang, Z. Meng, and I. King. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *WWW*, page 2331–2341, 2020.
[13] M. Gardner, J. Grus, et al. AllenNLP: A deep semantic natural language processing platform. *CoRR*, abs/1803.07640, 2018.
[14] T. Gärtner, P. A. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *COLT*, volume 2777, pages 129–143, 2003.
[15] Y. Govind, P. Konda, P. S. G. C., P. Martinkus, P. Nagarajan, H. Li, A. Soundararajan, S. Mudgal, J. R. Ballard, H. Zhang, A. Ardalan, S. Das, D. Paulsen, A. S. Saini, E. Paulson, Y. Park, M. Carter, M. Sun, G. M. Fung, and A. Doan. Entity matching meets data science: A progress report from the magellan project. In *SIGMOD*, pages 389–403, 2019.
[16] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *NIPS*, pages 1024–1034, 2017.
[17] W. Hua, K. Zheng, and X. Zhou. Microblog entity linking with social temporal context. In *SIGMOD*, pages 1761–1775, 2015.
[18] A. E. Johnson, T. J. Pollard, L. Shen, et al. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.
[19] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.
[20] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.
[21] H. Köpcke and E. Rahm. Frameworks for entity matching: A comparison. *Data Knowl. Eng.*, 69(2):197–210, 2010.
[22] I. Korkontzelos, D. Piliouras, A. W. Dowsey, and S. Ananiadou. Boosting drug named entity recognition using an aggregate classifier. *Artif. Intell. Medicine*, 65(2):145–153, 2015.
[23] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. Neural architectures for named entity recognition. In *NAACL*, pages 260–270, 2016.
[24] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinform.*, 36(4):1234–1240, 2020.
[25] J. Li, Y. Sun, R. J. Johnson, D. Sciaky, C.-H. Wei, R. Leaman, A. P. Davis, C. J. Mattingly, T. C. Wiegers, and Z. Lu. BioCreative V CDR task corpus: a resource for chemical disease relation extraction. *Database*, 2016, 05 2016.
[26] Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli. Graph matching networks for learning the similarity of graph structured objects. In *ICML*, pages 3835–3845, 2019.
[27] Y. Li, D. Tarlow, M. Brockschmidt, and R. S. Zemel. Gated graph sequence neural networks. In *ICLR*, 2016.
[28] C. D. Manning, M. Surdeanu, J. Bauer, et al. The Stanford CoreNLP natural language processing toolkit. In *ACL*, pages 55–60, 2014.
[29] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.
[30] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra. Deep learning for entity matching: A design space exploration. In *SIGMOD*, page 19–34, 2018.
[31] D. Q. Nguyen, T. D. Nguyen, D. Q. Nguyen, and D. Phung. A novel embedding model for knowledge base completion based on convolutional neural network. In *NAACL*, pages 327–333, 2018.
[32] S. Pradhan, N. Elhadad, W. Chapman, S. Manandhar, and G. Savova. SemEval-2014 task 7: Analysis of clinical text. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 54–62, 2014.
[33] PyTorch. https://pytorch.org/, 2020.
[34] R. J. Qureshi, J. Ramel, and H. Cardot. Graph based shapes representation and recognition. In *International Workshop on Graph-Based Representations in Pattern Recognition*, pages 49–60, 2007.
[35] P. Ristoski, J. Rosati, T. D. Noia, R. D. Leone, and H. Paulheim. Rdf2vec: RDF graph embeddings and their applications. *Semantic Web*, 10(4):721–752, 2019.
[36] G. K. Savova, J. J. Masanz, P. V. Ogren, J. Zheng, S. Sohn, K. K. Schuler, and C. G. Chute. Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications. *J. Am. Medical Informatics Assoc.*, 17(5):507–513, 2010.
[37] M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *ESWC*, pages 593–607, 2018.
[38] W. Shen, J. Han, J. Wang, X. Yuan, and Z. Yang. SHINE+: A general framework for domain-specific entity linking with heterogeneous information networks. *IEEE Trans. Knowl. Data Eng.*, 30(2):353–366, 2018.
[39] W. Shen, J. Wang, and J. Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Trans. Knowl. Data Eng.*, 27(2):443–460, 2015.
[40] D. Tikk and I. Solt. Improving textual medication extraction using combined conditional random fields and rule-based systems. *J. Am. Medical Informatics Assoc.*, 17(5):540–544, 2010.
[41] E. Tseytlin, K. J. Mitchell, E. Legowski, J. Corrigan, G. Chavan, and R. S. Jacobson. NOBLE - flexible concept recognition for large-scale biomedical natural language processing. *BMC Bioinform.*, 17:32, 2016.
[42] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *ICLR*, 2018.
[43] M. Wang, D. Zheng, Z. Ye, et al. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv:1909.01315*, 2019.
[44] P. Wang, S. Li, and R. Pan. Incorporating GAN for negative sampling in knowledge representation learning. In *AAAI*, pages 2005–2012, 2018.
[45] Q. Wang, B. Wang, and L. Guo. Knowledge base completion using embeddings and rules. In *IJCAI*, page 1859–1865, 2015.
[46] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu. Heterogeneous graph attention network. In *WWW*, page 2022–2032, 2019.
[47] D. Wright, Y. Katsis, R. Mehta, and C.-N. Hsu. NormCo: Deep disease normalization for biomedical knowledge base construction. In *AKBC 2019*, 2019.
[48] J. Wu, R. Zhang, Y. Mao, H. Guo, M. Soflaei, and J. Huai. Dynamic graph convolutional networks for entity linking. In *WWW*, pages 1149–1159, 2020.
[49] K. Xu, L. Wu, Z. Wang, Y. Feng, and V. Sheinin. Graph2seq: Graph to sequence learning with attention-based neural networks. *CoRR*, abs/1804.00823, 2018.
[50] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *SIGKDD*, page 974–983, 2018.
[51] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. Gnnexplainer: Generating explanations for graph neural networks. In *NeurIPS*, pages 9240–9251, 2019.
[52] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla. Heterogeneous graph neural network. In *SIGKDD*, page 793–803, 2019.
[53] Y. Zhang, Q. Yao, Y. Shao, and L. Chen. Nscaching: Simple and efficient negative sampling for knowledge graph embedding. In *ICDE*, pages 614–625, 2019.
[54] S. Zwicklbauer, C. Seifert, and M. Granitzer. DoSeR - A knowledge-base-agnostic framework for entity disambiguation using semantic embeddings. In *ESWC*, pages 182–198, 2016.