Graph Neural Networks for Entity Matching

Evgeny Krivosheev[1], Katsiaryna Mirylenka[2], Mattia Atzeni[3], Paolo Scotton[2]
Jumio Corp[1], *IBM Research Europe*[2], *EPFL*[3] {*kmi,psc*}@*zurich.ibm.com*

2023 IEEE International Conference on Big Data (BigData) | 979-8-3503-2445-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/BigData59044.2023.10386187

Abstract—**Data integration still remains largely rule-driven and lacks universal automation. In this work, we propose a general approach to modeling and integrating entities from structured data, such as relational databases, as well as unstructured sources, such as free text from news articles. Our approach is designed to explicitly model and leverage relations between entities, thereby using all available information and preserving as much context as possible. This is achieved by combining siamese and graph neural networks to propagate information between connected entities. We evaluate our method on the task of integrating data about business entities and demonstrate that it outperforms standard rule-based systems, as well as other deep-learning approaches that do not leverage any inductive bias towards the use of graph-based representations.**

I. Introduction

This paper introduces a methodology for leveraging graph-structured information in entity matching. The main idea consists of exploiting the structure of Knowledge Graphs (KGs) to create distributed representations of the nodes, so that entities related to each other (e.g., having the same entity type) in the KGs are closer in the embedding space. We assume that the data is either readily represented as a KG, or that a KG can easily be extracted from the data source. For example, in our use case, we experiment on data about companies and their relations (e.g., in terms of ownership and subsidiaries). This means that companies and their connections can be represented as a graph and stored accordingly.

Our approach relies on a Siamese Graph Convolutional Network (S-GCN) to learn a distance function between the nodes. At training time, the network is optimized to produce small distances for pairs of nodes belonging to the same entity, and large distances for unrelated nodes. Therefore, the trained network can then be used to assess the similarity between a query node and any reference node. S-GCNs are especially useful when the task involves a large number of classes, but only a few examples for each label are available.

The main contributions of this paper are as follows: (a) we investigate how graph modeling and GNNs can be applied to data-integration problems; (b)) we proposed a general approach for modeling hidden representations of entities using Siamese Graph Convolutional Networks; (c) we build a record linkage system and evaluate the performance of the model for entity matching of business entities. A demonstration of the system can be seen in [1].

II. Problem Statement

We assume to have two different databases R and Q, defined over a set of entities E, such that each record $r \in R$ and $q \in Q$ can be regarded as an entity mention and uniquely associated to an entity $e \in E$. Each entity represents a distinct real-world object, and we denote with e_r and e_q the entities associated to record $r \in R$ and record $q \in Q$ respectively. The goal of record linkage is to find a function $\mathcal{F}$ that takes as input the databases R and Q, and outputs a set of matches $\mathcal{M} \subset Q \times R$, such that $(q, r) \in \mathcal{M} \iff e_q = e_r$. Assuming we are matching records in Q against records in R, we will henceforth refer to R as the reference database.

We propose to extract graph representations of the databases and exploit the self-supervision provided by the graph structure to train a machine learning (ML) model for data integration, without the need for human-labeled data.

III. Proposed Approach

We approach entity matching as a two-stage process: (1) obtaining a graph representation R^* of the reference database R; (2) designing and training a GNN on R^* to perform entity-matching tasks on graphs.

GNNs for Data Integration. The reference graph R^* is defined as a directed multigraph, and we denote with $|R^*|$ the number of nodes. We assume to have a feature descriptor $x_r \in \mathbb{R}^D$ for every node $r^* \in R^*$, and we aggregate all these features in a matrix $X_R \in \mathbb{R}^{|R^*| \times D}$. Edges and relations for every pair of connected nodes are defined as triples of the form (r_i^*, ρ, r_j^*), where ρ is a relation type and r_i^* and r_j^* are nodes of the graph. On the other hand, a query graph Q^* consists of nodes $q^* \in Q^*$ that have to be matched to nodes in R^*.

Our goal is to match a node $q^* \in Q^*$ to a node $r^* \in R^*$ such that $e_q = e_r$. To achieve this, we propose to process the graph with GNNs and exploit the self-supervision provided by the structure of the reference graph. This process can be implemented in several ways. The first approach is to adopt the conventional classification setup. However, this usually requires a large number of training examples for each class label. In contrast, for record linkage, the number of unique entities (classes) is typically much larger than in classification problems, and the number of records belonging to the same entity is usually small. Also, this approach requires re-training the model when new entities are added to the database.

The methodology we propose is instead based on learning distributed representations for every node in R^*, so that nodes representing the same entity are closer in the embedding space and far from irrelevant nodes. This can be achieved by applying *deep metric learning* methods, such as Siamese networks [2]. R^* is used as training data for a GNN that produces an embedding vector of size M for each node $r^* \in R^*$. We denote as γ_r the embedding of the node $r^* \in R^*$, whereas Γ_R represents the set of all the embeddings for every node in R^*. At inference time, for a node $q^* \in Q^*$ we compute an embedding vector $\gamma_q \in \mathbb{R}^M$ by applying the already trained GNN model. At this point, we have three potential scenarios: *(i)* q^* cannot be matched successfully to any node in R^*; *(ii)* q^* can be matched to a single node $r^* \in R^*$; *(iii)* q^* can be matched to multiple nodes that refer to the same entity.

For the first two cases, the rule to link q^* to a node in the reference graph is given by: $r_c^* = \arg\min_{r^* \in R^*} dist(\gamma_q, \gamma_r)$,

$$\mathcal{N}(q^*, R^*) = \begin{cases} r_c^* & \text{if } dist(\gamma_q, \gamma_{r_c^*}) < t \\ \perp & \text{otherwise,} \end{cases} \tag{1}$$

where $dist(\gamma_q, \gamma_r)$ is the distance (e.g., Euclidean) between vectors γ_q and γ_r, $r_c^* \in R^*$ is the closest node to q^* in the embedding space Γ_R, $t \in \mathbb{R}$ is a threshold on the distance, and $\perp$ means no matches were provided according to the given threshold t. If multiple nodes refer to the same entity, the predicted probability of a match to an entity e is proportional to the sum of the similarities between the embeddings of the k nearest neighbors $\Gamma_q^k \subseteq \Gamma_R$ and the query embedding γ_q:

$$P_k(e_q = e \mid q^*, R^*) \propto \sum_{\gamma_r \in \Gamma_q^k} \delta(e, e_r) \cdot (1 - dist(\gamma_q, \gamma_r)), \tag{2}$$

where $\delta(e, e_r)$ is 1 if $e = e_r$ and 0 otherwise. The workflow described in this section is depicted in Figure 1.

Siamese GCN for Record Linkage. The proposed architecture consists of a Siamese Graph Convolutional Network (S-GCN) [3], [4]. It has the primary objective of learning discriminative hidden representations of nodes in a graph R^*, in such a way that they can then be used for further entity matching with previously unobserved data. The model is made of several GCN layers [3], each computing a non-linear function: $Z^{(l+1)} = \phi(Z^{(l)}, A)$, where l is the layer index and A is the adjacency matrix of the graph R^*. $Z^{(0)}$ is given by the initial feature matrix X_R. In our experiments, we employed a pre-trained BERT model [5] as the input layer to create input features for nodes. The last GCN layer L produces the output embedding matrix for all nodes, which forms the set of embeddings Γ_R of size z.

Our Siamese network is based on two identical GCNs. During the training period, the first GCN focuses on a given node r_1^* and its local neighborhood nodes $R_{r_1}^* \subset R^*$. This means that the GCN takes a subgraph around r_1^* as input and produces a vector of size z that finally forms the embedding vector γ_{r_1} of node r_1^*. The same procedure is repeated with the second GCN for node r_2^*, as described in [1]. Siamese networks are optimized with respect to the loss between their two outputs, namely γ_{r_1} and γ_{r_2}. The loss will be small for two nodes representing the same class (e.g., nodes of the same entity) and will be greater for nodes belonging to different classes. An instance of such a loss is the contrastive loss [6]:

$$ContrastiveLoss(\gamma_{r_1}, \gamma_{r_2}) = 0.5 \cdot (1 + y) \cdot dist(\gamma_{r_1}, \gamma_{r_2})^2 + \\ +0.5 \cdot (1 - y) \cdot \{max(0, m - dist(\gamma_{r_1}, \gamma_{r_2}))\}^2$$

where $y = 1$ if nodes (r_1^*, r_2^*) are from same class, and $y = -1$ otherwise, while m is a margin that indicates that dissimilar node pairs whose distance is beyond the margin will not penalize the network.

When the model is trained, one of the two Siamese networks (identical GCNs) is used for computing the output embedding matrix and forms Γ_R. It then sends the whole graph R^* as input to the GCN. Similarly, using the trained GCN, we can compute Γ_Q for an unobserved graph Q^* and perform entity matching in accordance to (1) and (2).

IV. Datasets and Preprocessing

We evaluate our approach on a database where each record corresponds to a different company business location. Company entities represented in the database include different branches, subsidiaries and headquarters, which can be grouped together and represented hierarchically as a single family. We are interested in linking a company name to a family in the database. We represent families in the database as a graph, where each node is a company name. Hence, the resulting graph consists of several disjoint connected components, each corresponding to a different family.

Data Preprocessing and Partitioning. The database we used contains around 700k business entities. As the record-linkage task we have set up works at the family level, we do not have an exact match at the record level from an entity in the test set to an equivalent entity in the training set. Moreover, some families contain completely non-overlapping company names. As an example, the companies *DIAMAS*, *Peter Instruments SAGL* and *Blaum GmbH* belong to the same family. To address both issues, we partition the dataset in such a way that names in the training set overlap with names in the test and validation sets.

We produce two different variants of the datasets, which we denote as overlap-1 and overlap-all. In the overlap-1 variant, we impose the constraint that each name in the validation and test sets has at least one representative word in common with at least one training example that belongs to the same family. This dataset addresses the first problem of not having an exact match for company entities at the record level. In the overlap-all version, we identify a representative token for each family as the most frequent token among the names in that family. Next, we filter out all names that do not contain the representative word and we partition the remaining ones into non-overlapping training, validation, and test sets. We also generate additional names that are short or normalized versions of a company name as described in [7].

V. Experimental Evaluation

We compare the approach described against two baselines. The first is a record linkage system (RLS) for company entities, introduced in [7]. This system relies on a rule-based approach to score entities with different attributes, including the name of a company, the precise location and the short company names. Hybrid rule- and ML-based approach to match product entities is introduced in [8], [9].

The second baseline is a multilayer perceptron (MLP) with 1 to 3 fully connected layers (depending on the dataset) and

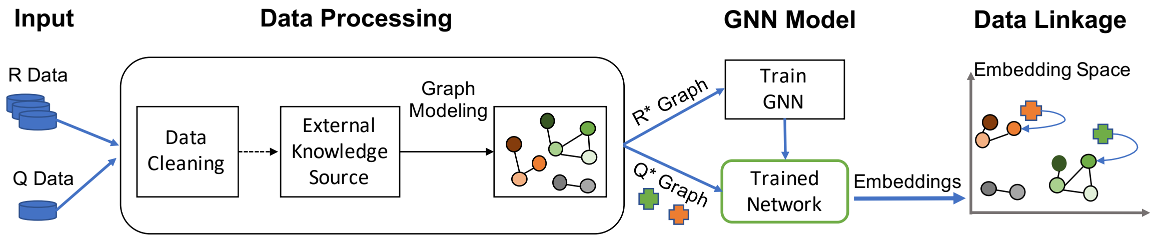

Fig. 1: Entity matching workflow with graph neural networks.

TABLE I: Accuracy on the overlap-1 and overlap-all datasets

Algorithm	**overlap-1**	**overlap-all**
RLS	0.71	0.80
MLP	0.71	0.90
GCN-CLF	0.56	0.90
S-GCN	0.70	0.91
RLS + short names	0.78	0.87
MLP + short names	0.75	0.91
GCN-CLF + short names	0.70	**0.94**
S-GCN + short names	**0.83**	0.93

a softmax classification layer on top, optimized with a cross-entropy loss function. The MLP takes BERT features as input for company names and produces a company family as output.

We use the accuracy score, i.e. the proportion of correctly matched records in Q, to assess the performance of the proposed graph algorithms. As GCN is a well-established choice for classifying graph nodes, we implement the data-linkage algorithm on graphs, called GCN-CLF, while keeping a classification softmax layer with the number of outputs equal to the number of company families as described in Section III.

We compare GCN-CLF and S-GCN against our baselines, namely RLS and MLP, on overlap-1 and overlap-all datasets (see Table I). For both datasets, the proposed graph-based algorithms outperform the baselines. After company names have been enriched by their short names, performance improves by a large margin, e.g., for S-GCN on overlap-1, accuracy rises 13% from 70% to 83%. Adding short names increases the variability of training data, and this allows GNNs to propagate information from short names to their original names.

It is worth mentioning that, for neural networks, overlap-1 is inherently difficult data to learn as it contains only a very small number of similar examples to train at. On overlap-1, MLP performs better than GCN-CLF, i.e. 0.75 vs. 0.7, respectively, and S-GCN has the best accuracy value of 0.83. One explanation of GCN-CLF's poor performance is due to the fact that non-overlapping company names might be connected in the KG. Therefore, the network propagates this high variance information through edges to overlapping companies, distorting their original samples. For MLP we still have non-overlapping companies in the training dataset as MLP does not rely on the graph structure (only on samples), therefore, the name variation for non-overlapping companies does not distort the original names samples.

S-GCN is less sensitive to the variations within a company family than GCN-CLF and it outperforms GCN-CLF by more than 10% on overlap-1. Recall that S-GCN is not trained directly for classification. Instead, it aims to put two similar nodes from R^* closer within the embedding space Γ_R. Even though some variation might come from neighboring nodes, S-GCN *does not explicitly force grouping* of all entities of the same family, i.e., S-GCN does not aim to learn a common pattern among all these entities at once (only for two adjacent nodes), unlike MLP and GCN-CLF, which try to learn a pattern to classify these entities. The overlap-all dataset is much "cleaner" as it does not contain non-overlapping companies in R^*, Q^*, which leads to better performance. Accuracy levels for S-GCN and GCN-CLF are comparable, namely 0.93 and 0.94 respectively, and exceed the results by MLP and RLS by 3% and 7%, respectively. More details about the training settings and results can be found in [10].

VI. Conclusion

We proposed a general GNN-based framework for learning hidden representations of entities and linking the entities represented in databases. The key takeaways are: *(i)* modeling relational data with an explicit graph structure improves entity representation and *(ii)* using graph neural networks can improve data integration.

References

[1] E. Krivosheev, M. Atzeni, K. Mirylenka, P. Scotton, C. Miksovic, and A. Zorin, "Business entity matching with siamese graph convolutional networks," in *AAAI'21*, vol. 35, no. 18, 2021, pp. 16 054–16 056.

[2] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," ser. CVPR, 2005.

[3] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR 2017*, 2017.

[4] J. Bromley and et al, "Signature verification using a "siamese" time delay neural network," ser. NIPS, 1993.

[5] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186.

[6] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," ser. CVPR '06, 2006.

[7] T. Gschwind, C. Miksovic, K. Mirylenka, and P. Scotton, "Fast record linkage for company entities," in *IEEE BigData*, 2019.

[8] K. Mirylenka, P. Scotton, C. A. M. Czasch, and A. Schade, "Similarity matching systems and methods for record linkage," 2021, uS Patent 11,182,395.

[9] K. Mirylenka, P. Scotton, C. Miksovic, and S.-E. B. Alaoui, "Linking it product records," in *ECML PKDD 2019*, 2020, pp. 101–111.

[10] E. Krivosheev, M. Atzeni, K. Mirylenka, P. Scotton, and F. Casati, "Siamese graph neural networks for data integration," *arXiv preprint arXiv:2001.06543*, 2020.