Potential Energy to Improve Link Prediction with Relational Graph Neural Networks

Simone Colombo[1,2], Dimitrios Alivanistos[1,3] and Michael Cochez[1,3]

[1]*Computer Science, Vrije Universiteit Amsterdam, The Netherlands*
[2]*Triply DB, Amsterdam, The Netherlands*
[3]*Elsevier Discovery Lab, Amsterdam, The Netherlands*

Abstract

Potential Energy (PE) between 2 bodies with mass, refers to the relative gravitational pull between them. Analogously, in the context of a graph, nodes can thought of as objects where a) the product of the degrees of nodes acts as a proxy for mass, b) the clustering coefficients of common neighbours as a proxy for gravitational acceleration, and c) the inverse of the shortest distance between nodes as a proxy for distance in space, which allows for PE calculation as introduced in prior work. In this work, we are investigating the effects of incorporating PE in Link Prediction (LP) with Relational Graph Convolutional Networks (R-GCN). Specifically, we explore the benefits of including PE calculation as an informative prior to the LP task and in a follow-up experiment as a learnable feature to predict. We performed several experiments and show that considering PE in the LP process has certain advantages and find that the information PE provides was not captured by the embeddings produced by the R-GCN.

Keywords

Machine Learning, Potential Energy, Link Prediction, Graph Neural Networks

1. Introduction

Nowadays, Knowledge Graphs (KGs) are widely used in a variety of different fields [1], and have become the backbone of different AI-driven applications which are employed in diverse domains [2, 3, 4]. They have a long history, but after the term Knowledge Graphs was used by Google [5], other companies followed the trend [6, 7, 8, 9, 10]. The information encoded in Knowledge Graphs can be used as an input to solve various tasks, including Link Prediction (LP), Question Answering, Recommender Systems and more [4].

Traditional LP employed topological features of graphs to predict edges between nodes that are not present, yet are true. More recently proposed LP approaches employ advancements in Machine Learning and specifically in the area of Graph Neural Networks to learn latent representations for entities and relations in a graph. These representations are then used to

In A. Martin, K. Hinkelmann, H.-G. Fill, A. Gerber, D. Lenat, R. Stolle, F. van Harmelen (Eds.), Proceedings of the AAAI 2022 Spring Symposium on Machine Learning and Knowledge Engineering for Hybrid Intelligence (AAAI-MAKE 2022), Stanford University, Palo Alto, California, USA, March 21–23, 2022.

s2.colombo@student.vu.nl (S. Colombo); d.alivanistos@vu.nl (D. Alivanistos); m.cochez@vu.nl (M. Cochez)
https://github.com/simoneVU (S. Colombo); https://dimitrisalivas.github.io/ (D. Alivanistos); https://www.cochez.nl/ (M. Cochez)
0000-0002-0090-2069 (D. Alivanistos); 0000-0002-9421-8566 (M. Cochez)

perform the LP task. However, there are cases where the underlying topological features of the graph get lost in the transition to vector space in the form of multi-dimensional embeddings.

Potential Energy in graphs, as introduced by Kumar and Sharma [11], refers to the potential energy between two nodes in the graph. It is a measure that takes into account topological characteristics of the graph; specifically node degrees, clustering coefficients, and the inverse shortest distance between two nodes of interest.

In this work, we evaluate PE as an additional signal for performing LP in a co-authorship network. We further investigate whether PE can be predicted by the vanilla R-GCN architecture [12]; if that would be the case, then we would not necessarily need to add the pre-calculated PE as an additional signal, but would just have to incorporate its prediction in the link prediction component. With the intent of answering said questions, we developed a GNN-based architecture and specifically employed an R-GCN [12], which we further on enhance with the additional information of PE in our experiments. Our initial experimental results suggest that PE improves the accuracy of LP when included as an additional signal to the R-GCN. Furthermore, our results suggest that the information given by the PE cannot be obtained from the embeddings generated by the R-GCN, implying loss of topological information.

2. Building Blocks

We consider link prediction (recovering of missing triples) in a transductive setting, as our task. Missing pieces of information in graphs can be predicted using the neighborhood information of nodes. For example, knowing that *Michael* works with a person with whom *Wouter* works as well, implies that it is likely that the triple (*Michael*, *friend_with*, *Wouter*) belongs to the Knowledge Graph. Our link prediction model follows the work of Schlichtkrull et al. [13]. The model consists of a) an encoder, the R-GCN, and b) an enhanced decoder version of DistMult [14]. The enhancement lies in the proposed decoder that leverages the Potential Energy [11] between two nodes as an additional feature.

2.1. Relational Graph Convolutional Networks

The Relational Graph Convolutional Network (R-GCN) is Graph Neural Network that operates on relational data. This means that the data is a graph with directed typed edges. The update for layer $l+1$ and node i is:

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}W_r^{(l)}h_j^{(l)} + W_0^{(l)}h_i^{(l)}\right), \tag{1}$$

where $\mathcal{N}_i^r$ denotes the set of neighbor indices of node i under relation $r\in\mathcal{R}$. $c_{i,r}$ is the number of incoming edges of type r to node i, W_r is the weight matrix of relation type r, and W_0 the weight matrix used to sure the next update of a node retains the information from the previous state (simulating a self loop). This update accumulates feature vectors of neighbors via a normalized sum after performing a relation specific transformation. Executing the neural network means evaluating Equation (1) for each node in the graph, for each layer (sequentially).

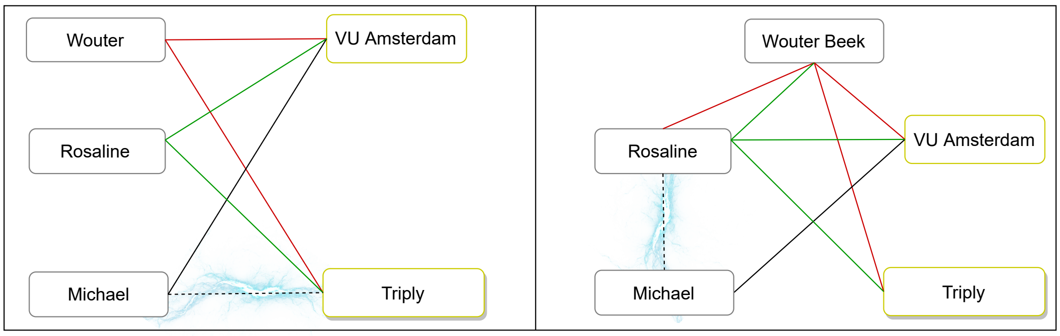

Figure 1: On the left: Depiction of the potential energy, represented as a light blue lightning , for nodes A and B. in a Bipartite graph. On the right: Depiction of the potential energy, represented as a light blue lightning , for nodes A and B in a Non-bipartite graph. The green and red links represent the triangles used in the calculation of the clustering coefficients for the common neighbours of node A (i.e. *VU Amsterdam*) and B (i.e. *Triply*).

2.1.1. R-GCN for Link Prediction

Link prediction is the task in which we predict either missing edges in the graph, when the graph is static, or potential future links that need to be added when the graph is dynamic. Formally, let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$ be our graph with vertices $\mathcal{V}$, edges $\mathcal{E}$, and relation types $\mathcal{R}$, which we assume to be complete. Then, given an incomplete set of edges, $\hat{\mathcal{E}}$, the objective is to assign scores to potential existing edges (s, r, o) in order to determine how likely they are to belong to $\mathcal{E}$. Following [13], we can tackle this problem using the R-GCN as an entity encoder and the DistMult decoder as a scoring function. The output of the R-GCN for one node $i \in \mathcal{V}$ is a real-valued vector $e_i \in \mathbb{R}^d$. DistMult uses the representation two nodes s and o and the learnt representation of the relations types R_r to compute the score for a triple as:[1]

$$f(s, r, o) = e_s^T R_r e_o, \tag{2}$$

Cross-entropy loss is used as the optimization objective for this task.

2.1.2. Input Features for the R-GCN

The nodes at the first layer of the R-GCN can be initialized with features. We initialize them with a feature vector which represents the textual information we have available about the node. To do this, we encode string attributes with a pre-trained BERT model [15]. If a textual attribute has multiple tokens, we sum the outputs of the BERT model to obtain one embedding. If an attribute is multi-valued, we sum the embeddings of the individual values.

[1]This representation is a diagonal matrix and learnt *separately* from the representations W_r in the R-GCN.

2.2. Potential Energy

In this work, we want to see the effect of including a topological feature as additional information for the link prediction of the R-GCN. We chose the Potential Energy (PE) as introduced by Kumar and Sharma [11] for link prediction in a bipartite graph[2]. A bipartite graph is a graph $\mathcal{G}_{\mathcal{B}} = (\mathcal{V}_{\mathcal{A}}, \mathcal{V}_{\mathcal{B}}, \mathcal{E})$, where $\mathcal{V}_{\mathcal{A}}$ and $\mathcal{V}_{\mathcal{B}}$ disjoint sets of vertices, $\mathcal{E}$ a set of undirected edges connecting a node from $\mathcal{V}_{\mathcal{A}}$ with a node of $\mathcal{V}_{\mathcal{B}}$. The potential energy PE_{AB} between two nodes $A \in \mathcal{V}_{\mathcal{A}}$ and $B \in \mathcal{V}_{\mathcal{B}}$ can be computed, and is the product of three factors:

a) $d_A d_B$, the product of the degree of the nodes,

b) $\sum_{z \in \Gamma(A) \cap \Gamma(B)} cl_z$, the sum of the clustering coefficients of common neighbors , where $\Gamma(X)$ the set of neighbors of X, cl_z the clustering coefficient of node z. If there are no common neighbors, cl_z is defined as 0.1.

c) $\frac{1}{sd(A,B)}$, the inverse of the length of the shortest distance (path) between the nodes.

In this definition we assume that if there is no path between two nodes, $sd(A, B) = +\infty$, and hence the potential energy becomes zero.

2.2.1. Potential Energy for a Knowledge Graph

In this paper, we do not work with a bipartite graph, but rather a Knowledge Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$, as defined in Section 2.1.1. Hence, we have to (re-)define what Potential Energy means. We make the following modifications, when computing PE_{AB}:

1. We only have one set of nodes, and hence the nodes $A, B \in \mathcal{V}$.
2. To compute the shortest path, we ignore the relation types and regard the graph as undirected.

Otherwise, we use the PE formula as stated above. In a Knowledge Graph, if the nodes are in disconnected components, their PE is zero. We illustrate the PE in Figure 1.

3. Experimental Setup and Outcomes

3.1. Dataset

We evaluate our model on a Knowledge Graph with scholarly data; namely the IOS LD Connect Scholarly Knowledge Graph[3]. All the metadata about the papers are serialized and published as Linked Data following the bibliographic ontology[4] and is accessible through a SPARQL endpoint[5].

[2]In that work, the authors do not combine this measure with a learning approach.
[3]http://ld.iospress.nl
[4]http://bibliontology.com/#sec-sioc-rdf
[5]http://ld.iospress.nl:3030

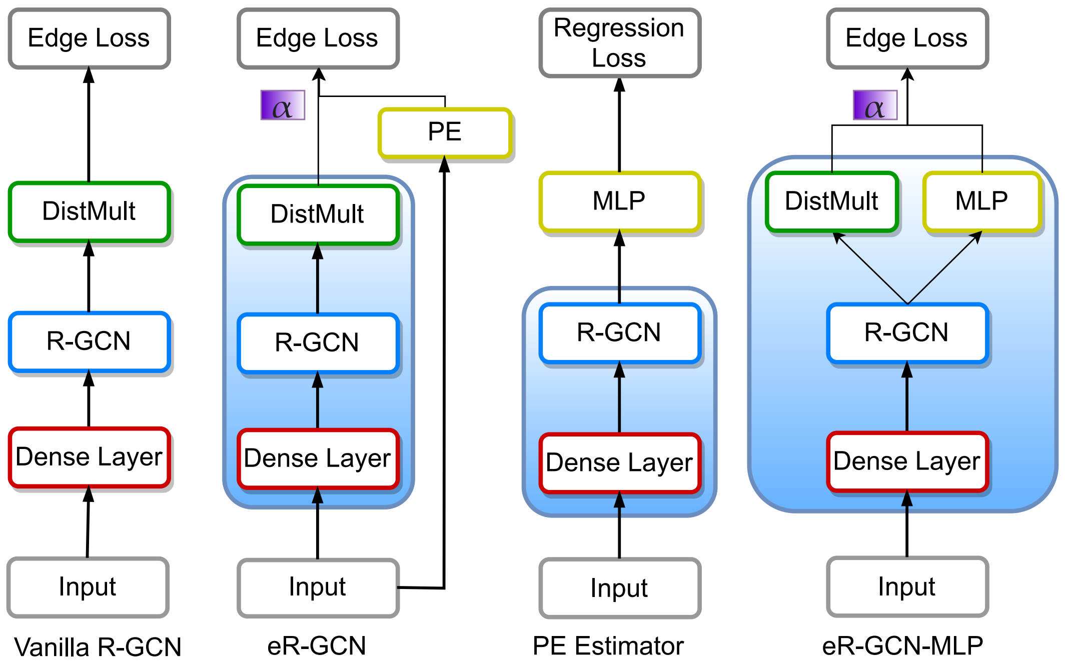

Figure 2: Vanilla R-GCN: The R-GCN for LP with an additional dense layer to reduce the dimension of the textual embeddings, **eR-GCN:** The energized R-GCN - utilizing the additional signal of PE for prediction, note that PE comes directly from the input graph topology, also note the trainable parameter α that represents signal weight. **PE Estimator:** An MLP-Regressor trained to predict the PE between nodes. **eR-GCN-MLP:** The combination of eR-GCN and PE estimator that optimizes α based on both the R-GCN and the PE Estimator. The light blue box indicates frozen parameters, which allows us to isolate the effects of specific parts in the architecture.

Preprocessing. As a first data cleaning step, we removed all triples related to geometrical information. Then, we gather nodes relevant for co-authorship link prediction; namely entities of the following types with their respective textual attributes: **authors** with their full name, **articles** with their title and keywords, and **affiliations** with their names. Then, we extract the relations which link authors to the papers they wrote, co-authors of a specific paper to each other, and authors to their affiliations. In parallel, we merged nodes if there were multiple nodes representing the same entity, for example two nodes for the exact same author, as indicated by the `owl:sameAs` relation. The resulting graph has 17748 edges, which we split in 65% (13330) for training, 20% (2525) for validation and 15% (1893) for testing.

3.2. Model Architecture

To investigate whether the potential energy is beneficial to link prediction with an R-GCN, we modify the decoder part of our architecture to incorporate the potential energy between the nodes. We perform two experiments. In both, we initially train the R-GCN with the link prediction objective, see *Vanilla R-GCN* on the far left side of Figure 2.

eR-GCN. For the first experiment, we incorporate the potential energy into the decoder by taking a linear combination of the output of the original decoder and the calculated potential energy amongst the specific pair of nodes in the triple, as such:

$$f(s,r,o) = (1-\alpha)e_s^T R_r e_o + \alpha PE_{so}, \tag{3}$$

We then optimize the parameter α via the link prediction objective. The reason behind these experiments lies in what we would like to observe. First of all, our goal is to showcase whether PE improves the LP task and quantify said improvement. Hence, we would like to observe the behaviour of the α parameter and specifically whether post-training it is greater than zero. If that is the case, it implies that PE as a metric holds valuable information for the LP task, that either the R-GCN or the DistMult decoder failed to capture. We refer to the resulting model as energized R-GCN or eR-GCN.

eR-GCN-MLP. In order to deduce whether it is solely the DistMult decoder which is unable to decode the PE information, or whether it is an inherent weakness to the R-GCN, we setup a second experiment. In the first stage of this experiment, we train a MLP which estimates the potential energy from the embeddings created by the R-GCN, see *PE Estimator* in Figure 2. The hypothesis is that if the embeddings contain the topological information necessary to predict PE, then the MLP should be able to estimate it (the MLP is a universal function approximator). We then utilise the output of the MLP to replace the actual computation of the PE and again observe both the performance and the value of α, see *eR-GCN-MLP* in Figure 2. If the value of α is similarly high compared to the first experiment, then this is a sign that a different decoder would be able to capture the information contained in the PE. If α is lower, then this means that the topological information contained in the PE calculation is not captured by the embeddings.

3.3. Training and Parameters

The pre-trained BERT embeddings used have a dimension of 768, which is too computationally expensive for our model to use, so we employed an MLP to reduce the dimensions to 64. This is then also the input dimension of the nodes of our 3-layer R-GCN. On the first hidden layer, the dimension is 14, the second and the output layer have a dimension of 32. In the R-GCN we made use of basis decomposition, with 2 bases. The internal dimensions and number of bases were optimized using a hyper-parameter search on the vanilla R-GCN , and then kept the same for the other experiments. We used Adam as an optimizer for 300 epochs and a learning rate of 0.01. Regarding the training objectives, the link prediction models are all trained using the cross-entropy loss while for training the regression model to predict the PE, we employ the L1 loss (we also experimented with L2 loss, with comparable results).

Negative Sampling. Link prediction models are trained on both positive and negative samples. In the context of KGs, every triple that exists in the graph is considered true. However, negative triples are not present. Hence, in order to train our model we generated negative samples by corrupting the subject or object of existing triples. We did this corruption by replacing the head or tail of an existing triple, with another entity from the graph sampled uniform at random. In every split, we included one negative triple for every positive one.

3.4. Results

Model	Training loss	Validation AUC-ROC	Test AUC-ROC	Scaling factor (α)
R-GCN	0.3782 ± 0.0252	0.8923 ± 0.0191	0.8912 ± 0.0173	n.a.
eR-GCN	0.2014 ± 0.0352	0.9186 ± 0.0230	0.9209 ± 0.0221	0.2925 ± 0.0097
eR-GCN-MLP	0.3665 ± 0.0001	0.8997 ± 0.0001	0.8993 ± 0.0001	≈ 0

Table 1
The outcomes of the experiments on the vanilla R-GCN, the eR-GCN and the eR-GCN-MLP.

The results obtained for the two baseline and the two experiments are in Table 1. These are the mean and standard deviations of the training loss, validation and test accuracy and scaling factor over 30 runs.

Baseline. As a baseline of our experiments, we compare against the standard R-GCN version for link prediction without the enhanced DistMult decoder. The results of this model are on the first row on Table 1, denoted by R-GCN. We introduce and compare the baseline against two variants of our introduced eR-GCN model.

eR-GCN. The first experiment utilises the pre-calculated PE as an additional feature for performing LP. The α parameter remains trainable, allowing for further insights on the trade-off between utilising and not utilising PE. We observe that that the eR-GCN the scaling factor is relatively large, meaning that the potential energy is incorporated, and provides useful information which the standard decoder cannot capture.

eR-GCN-MLP The second experiment employs an additional MLP to perform a regression task over the PE. Intuitively, we would like to see whether the latent node representations retain some original graph topological features, by trying to predict the PE from them. We observe that the scaling factor becomes zero, meaning that it does not contribute positively to the predictive power of the model and thus the information provided by the potential energy was not captured in the embeddings which the R-GCN gave when trained with the DistMult objective.

4. Related Work

eR-GCN and Link Prediction. Our encoder-decoder model approach to LP is based on DistMult in the decoder [14], enhanced by the Potential Energy [11]. There is a plethora of different approaches to perform LP, categorized in 4 large clusters i.e Similarity-based, Probability-based, Dimensionality Reduction-based, and "other" [16]. Given the definition in Kumar et al. [16], we can see that our eR-GCN approach is combining elements from both the first and the third cluster (similarity and dimensionality reduction). Specifically, we utilise embeddings (that fall under dimensionality reduction) and local similarity indices (that fall under similarity-based). In addition to that, we employ *Learning-based* methodology (under category *other*), since we train a neural network architecture to perform the task.

Enhancing Graph Neural Networks. In terms of our neural architecture, our eR-GCN is closely related to the R-GCN by Kipf and Welling [12] and the underlying Message-Passing mechanism behind it. Previous research on enhancing GNNs employed graph features in two flavours; 1) by enhancing the GCNs directly with the use of topological graph features as done in [17] with the use of specific graph motifs and 2) by incorporating topological graph information in the embeddings creation process as in [18], where they employ an attention mechanism to recognize positional and centrality qualities. However, to the best of our knowledge, none of these directions considered investigating whether topological features of high-complexity such as PE can be captured by an R-GCN and aid in the LP task.

5. Conclusion & Future Work

We have introduced the energized relational graph convolutional network (eR-GCN) and investigated the impact of employing Potential Energy between nodes as an enriching factor in LP. Furthermore, our findings suggest that the vanilla R-GCN architecture cannot capture PE, at least not when trained with a DistMult decoder. Moreover, enriching the factorization model with the PE metric proved valuable for the link prediction task, yielding a small improvement of the AUC-ROC over the R-GCN baseline.

There are several ways in which our work could be extended. First, we can probe into adding more layers or residual connections to the R-GCN, aiming to increase its generalization power towards topological graph features, such as the PE. Additionally, we would like to investigate if the lack in ability for R-GCNs to capture PE, is true for other topological features of the graph. Lastly, in order to solidify our findings further, it would be necessary to run more extensive experimentation using a wider selection of datasets, to measure the effect of the properties of the graph on the reported improvements. Finally, a detailed theoretical analysis of the GNN used in our experiments could provide further insights on why the network is unable to capture the Potential Energy. Besides, there might be other GNN architectures which can capture complex topological features, such as PE, of the graph. Further investigation should be carried out to determine whether this is only a restriction of the R-GCN specifically.

Acknowledgments

We would like to thank Wouter Beek and Rosaline de Haan from Triply, Amsterdam and Peter Bloem from the Vrije Universiteit Amsterdam, for the interesting discussions on this work and their overall support. The work presented here overlaps with the thesis of the first author.

References

[1] B. Abu-Salih, Domain-specific knowledge graphs: A survey, 2021. arXiv:2011.00235.

[2] Y. Gao, Y.-F. Li, Y. Lin, H. Gao, L. Khan, Deep learning on knowledge graph for recommender system: A survey, 2020. arXiv:2004.00387.

[3] M. Nickel, K. Murphy, V. Tresp, E. Gabrilovich, A review of relational machine learning for knowledge graphs, Proceedings of the IEEE 104 (2016) 11–33. doi:10.1109/JPROC.2015.2483592.

[4] S. Ji, S. Pan, E. Cambria, P. Marttinen, S. Y. Philip, A survey on knowledge graphs: Representation, acquisition, and applications, IEEE Transactions on Neural Networks and Learning Systems (2021). URL: 10.1007/978-3-030-86520-7_34.

[5] A. Singhal, Introducing the knowledge graph: things, not strings, Google Blog (2012). Https://blog.google/products/search/introducing-knowledge-graph-things-not/.

[6] S. H. R.J. Pittman, Amit Srivastava, Cracking the code on conversational commerce, eBay Blog (2017). Https://blog.aboutamazon.com/innovation/making-search-easier.

[7] A. Krishnan, Making search easier: How amazon's product graph is helping customers find products more easily, Amazon Blog (2018). URL: https://blog.aboutamazon.com/innovation/making-search-easier.

[8] R. Socher, D. Chen, C. D. Manning, A. Y. Ng, Reasoning with neural tensor networks for knowledge base completion, in: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1, NIPS'13, Curran Associates Inc., Red Hook, NY, USA, 2013, p. 926–934.

[9] N. Noy, Y. Gao, A. Jain, A. Narayanan, A. Patterson, J. Taylor, Industry-scale knowledge graphs: Lessons and challenges, Commun. ACM 62 (2019) 36–43. URL: https://doi.org/10.1145/3331166. doi:10.1145/3331166.

[10] D. Devarajan, Happy birthday watson discovery, IBM Cloud Blog (2017). URL: https://www.ibm.com/cloud/blog/announcements/happy-birthday-watson-discovery.

[11] P. Kumar, D. Sharma, A potential energy and mutual information based link prediction approach for bipartite networks, Scientific Reports 10 (2020).

[12] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2017. arXiv:1609.02907.

[13] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, M. Welling, Modeling relational data with graph convolutional networks, 2017. arXiv:1703.06103.

[14] B. Yang, W. tau Yih, X. He, J. Gao, L. Deng, Embedding entities and relations for learning and inference in knowledge bases, 2015. arXiv:1412.6575.

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

[16] A. Kumar, S. S. Singh, K. Singh, B. Biswas, Link prediction techniques, applications, and performance: A survey, Physica A: Statistical Mechanics and its Applications 553 (2020) 124289.

[17] X. Li, W. Wei, X. Feng, X. Liu, Z. Zheng, Representation learning of graphs using graph convolutional multilayer networks based on motifs, 2020. arXiv:2007.15838.

[18] A. Sadeghi, D. Collarana, D. Graux, J. Lehmann, Embedding knowledge graphs attentive to positional and centrality qualities, 2021.