Rewiring Knowledge Graphs by Graph Neural Network Link Predictions

Alex Romanova[a]
Melenar, LLC, McLean, VA, US, 22101, U.S.A.

Keywords: Link Prediction, Knowledge Graph, Graph Neural Network, Graph Topology, Deep Learning.

Abstract: Knowledge Graphs recently received increasing attention from academia and industry as a new era in data-driven technology. By building relationships graphs are 'connecting the dots' and moving data from zero-dimensional to multi-dimensional space. Emerging Graph Neural Network (GNN) models are building a bridge between graph topology and deep learning. In this study we examine how to use GNN link prediction models to rewire knowledge graphs and detect unexplored relationships between graph nodes. We investigate diverse advantages of using highly connected and highly disconnected node pairs for graph mining techniques.

1 INTRODUCTION

On his keynote presentation on Semantics 2017 conference in Amsterdam, Aaron Bradley declared that "Semantic Web has died but in 2012 it was reincarnated by Google Knowledge Graph" (Bradley, 2017). Knowledge graph became essential in both academia and industry as a new era in data integration and data management. Knowledge graphs provide the structured data and factual knowledge that drive many products and make them more intelligent and "magical" (Noy et al., 2019).

Google Knowledge Graph conceptually is similar to Semantic Web and in many cases knowledge graphs are built based on Semantic Web fundamental techniques, in particular on Sparql language. In our previous paper (Romanova, 2020) we examined limitations of Sparql language and demonstrated how knowledge graphs can be build by non-Sparql methods. Also we demonstrated that knowledge graph abilities are much wider than search and data integration.

Methods that we used in that study were based on traditional property graph techniques. In this study we will show how to rewire knowledge graphs through emerging Graph Neural Network (GNN) link prediction models.

The year when Google Knowledge Graph was introduced was a breakthrough year for Deep Learning: in 2012 the evolutionary model AlexNet was created (Krizhevsky et al., 2012). Convolutional Neural Network (CNN) image classification techniques demonstrated great success outperforming previous state-of-the-art machine learning techniques in various domains (LeCun et al., 2015). For several years deep learning and knowledge graph were growing in parallel until in the late 2010s GNN bridged the gap between them (Bronstein et al., 2021).

Table 1: Numbers of Words in Wikipedia Articles about Modern Art Artists.

Artist	Number of Words
Vincent van Gogh	13677
Paul Gauguin	13249
Marc Chagall	12627
Paul Cézanne	8609
Claude Monet	7852
Pablo Picasso	6713
Vasily Kandinsky	6491
Paul Klee	6314
Henri Matisse	5188
Piet Mondrian	5148
Jackson Pollock	4626
Joan Miró	3959
Oskar Kokoschka	3247
Kazimir Malevich	3097
Egon Schiele	3048
Paul Signac	2290
Natalia Goncharova	1897
Max Beckmann	1850
Georges Braque	1639
Franz Marc	1324

CNN and GNN models have a lot in common: both CNN and GNN models are realizations of Geometric Deep Learning. What is peculiar for GNN is the fact that in GNN node features are not just arbitrary vectors but coordinates of geometric entities

[a] https://orcid.org/0000-0002-5927-2129

Romanova, A.
Rewiring Knowledge Graphs by Graph Neural Network Link Predictions.
DOI: 10.5220/0011664400003393
In *Proceedings of the 15th International Conference on Agents and Artificial Intelligence (ICAART 2023) - Volume 2*, pages 149-156
ISBN: 978-989-758-623-1; ISSN: 2184-433X

(Bronstein et al., 2021). Based on this, GNN link prediction models allow to combine node features with graph topology.

In this study we will use GNN link prediction models to find missing links in knowledge graphs. Finding missing links for knowledge graphs helps to solve numerous problems, in particular knowledge graph incompleteness. Also adding links to knowledge graphs allows to detect unknown relationships between graph nodes.

Experiments of our previous knowledge graph study (Romanova, 2020) were based on finding unknown relationships between modern art artists. As data for experiments we used artist biographies, known relationships between artists and data about modern art movements. For experiments of this study we will use Wikipedia articles about the same 20 modern art artists (please see Table 1).

We will examine two different scenarios: one scenario is based on artist names and full text of Wikipedia articles and another scenario is based on distribution of co-located words within and across the articles.

For the first scenario we will build initial knowledge graph on artist names and Wikipedia text as nodes and relationships between artists and corresponding articles as edges. Then we will embed node features through transformer models and generate additional edges for artist pairs if their corresponding Wikipedia article vectors will have high cosine similarities. Modified knowledge graph will be used as input data to GNN link prediction model.

For the second scenario we will build initial knowledge graph with nodes as pairs of co-located words and edges as pairs of nodes with common words. That knowledge graph will represent not only word sequences within articles but also chains of words across Wikipedia articles about different artists.

After running GNN link prediction models on top of both knowledge graphs, we will rewire initial knowledge graphs through similarities of re-embedded nodes.

In this paper we will demonstrate the following:

- Describe related work.
- Examine raw data analysis.
- Describe methods of data preparation, model training and interpreting model results.
- Explain in different scenarios how to rewire knowledge graphs based on interpreting the model results.
- Illustrate applications of highly similar and highly dissimilar artist pairs for recommender systems.
- Emphasize that pairs of dissimilar nodes provide for graph mining quite different values that pairs of similar nodes.

2 RELATED WORK

After it was introduced by Google, knowledge graph was adapted by many companies as a powerful way to integrate and search various data such as structured, unstructured or semi-structured data taken from a variety of sources. Knowledge graphs combine internal data with public knowledge, drive a variety of data products and make them more intelligent (Noy et al., 2019).

Knowledge graph organizes various data types and data volumes to highlight relationships between data points. Relationship is one of the main reasons of knowledge graph popularity but in practice in existing knowledge graphs it is often incomplete.

Also real-world data are often dynamic and evolving, which leads to difficulty in constructing correct and complete knowledge graphs and it is a challenging task to automatically construct complete dynamic knowledge graphs. Link prediction is one of ways to solve these challenging problems (Wang et al., 2021).

Link prediction is a fundamental problem that attempts to estimate a likelihood of existence of a link between two nodes, which makes it easier to understand associations between two specific nodes and how the entire network evolves (Wu et al., 2022). The problem of link prediction over complex networks can be categorized into two classes. One is to reveal the missing links. The other is to predict the links that may exist in the future as the network evolves.

Various types of link predictions has been widely applied to a variety of fields. In social networks link predictions support potential collaborations and help to find assistants. In biology and medicine link predictions provide ability to foresee hidden associations like protein–protein interactions. (Zhou, 2021).

In recent years, link predictions are extensively used in social networks, citation networks, biological networks, recommender systems, security and so on and link prediction models attract more and more studies.

Before GNN became an emerging research area link prediction techniques were based either on graph topology or on node features (Zhou et al., 2009). There has been a surge of algorithms that make link prediction through representation learning that learns low dimensional embeddings such as DeepWalk (Grover and Leskovec, 2016), node2vec (Perozzi et al., 2014), etc. Over the years many link prediction methods have been developed (Wang and Vinel, 2021).

As the Graph Neural Networks have been an emerging research area in recent years, significant advances and various architectures were proposed and developed (Wu et al., 2022).

For this study we will use GraphSAGE link prediction model (Hamilton et al., 2017), an inductive learning algorithm for GNNs which instead of applying the whole adjacency matrix information among all nodes, learns aggregator functions that can induce the embedding of a new node given its features and neighborhood information without retraining of the entire model (Wang and Vinel, 2021).

3 METHODS

We will describe data processing, model training and interpreting model results in the following order:

- We will start with description of node embedding process. In the second scenario node embedding will be used only for GNN link prediction model, but in the first scenario it also will be used to add edges to the input knowledge graph.
- Then we will describe the first scenario: knowledge graph based on artist names and Wikipedia full text as nodes and connections between artists and corresponding Wikipedia articles as edges.
- Next we will describe the second scenario: knowledge graph based on co-located word pairs as nodes and pairs joint through common words as edges.
- Next we will illustrate how to prepare and train GNN link prediction models.
- And finally we will define how to rewire knowledge graphs based on the model results interpretations.

For data processing, model training and interpreting the results we will use techniques that are described in details in our technical blog (sparklingdataocean.com, 2022a; sparklingdataocean.com, 2022b).

3.1 Node Embedding

For both knowledge graph scenarios to translate text to vectors we will use the 'all-MiniLM-L6-v2' transformer model from Hugging Face. This is a sentence-transformers model that maps text to a 384 dimensional dense vector space.

There are two advantages of embedding text nodes:

- Vectors generated by transformers can be used for GNN link prediction model as node features.
- Graphs can get additional edges on highly connected vector pairs.

We will use the first technique for both knowledge graph scenarios and the second technique only for the first scenario - knowledge graph built on artist names and full text of Wikipedia articles. For the first scenario we will calculate cosine similarity matrix for vectors generated by transformers, select from that matrix highly connected pairs and generate on those pairs additional graph edges.

3.2 Build Initial Knowledge Graph on Names and Full Text

For the first scenario to build a knowledge graph on artist names and full text of Wikipedia articles we will do the following:

- Define nodes as artist names and full text of Wikipedia articles.
- Define edges as pairs of artist names and corresponding articles.
- Embed nodes through transformer model.
- Calculate cosine similarity matrix for pairs of vectors and add highly connected pairs of nodes as edges to the graph.
- Build a knowledge graph on these nodes and edges.

Detail information about the first scenario is described in our technical blog (sparklingdataocean.com, 2022b)

3.3 Build Initial Knowledge Graph on Co-Located Word Pairs

For the second scenario to build a knowledge graph on co-located word pairs we will do the following:

- Tokenize Wikipedia text and exclude stop words.
- Get nodes as co-located word pairs.
- Get edges between nodes.
- Build a knowledge graph.

To generate edges we will find pair to pair neighbors following text sequences within articles and joint pairs that have common words.

```
if pair1=[leftWord1, rightWord1],
   pair2=[leftWord2, rightWord2]
   and rightWord1=leftWord2,
then there is edge12={pair1, pair2}
```

Graph edges built based of these rules will cover word to word sequences and word to word chains within articles. More important, they will connect different articles by covering word to word chains across articles.

Description of the second scenario and code can be checked in our technical blog (sparklingdataocean.com, 2022a)

3.4 Training the GNN Link Prediction Model

For this study we will use GraphSAGE link prediction model GraphSAGE (Hamilton et al., 2017). This algorithm is based on learning aggregator functions that can induce the embedding of a new node given its features and neighborhood information without retraining of the entire model. The concatecated vector will be passed through a GNN layer to update the node embedding.

As Graph Neural Networks (GNN) link prediction model we used a model from Deep Graph Library (DGL) (DGL, 2018). The model is built on two GraphSAGE layers and computes node representations by averaging neighbor information.

For data preparation and model training we used the code provided by DGL tutorial. In our code we only had to transform input graph data to DGL data format. Coding techniques are available in our technical blog (sparklingdataocean.com, 2022b).

3.5 Interpreting Results of the GNN Link Prediction Model

The results of GNN link prediction model are re-embedded nodes that can be used for further data mining such as node classification, k-means clustering, link prediction and so on.

The goal of this study is to find unknown connections between modern art artists. To do it in the first scenario we will use the results of the model in traditional way: we will estimate cosine similarities between re-embedded node pairs and select graph edges based on cosine threshold.

In the second scenario we will use a non-traditional approach. We will aggregate re-embedded nodes by artists and estimate link predictions by cosine similarities between aggregated vectors.

4 EXPERIMENTS

In this section we will present results of our experiments.

- First we will introduce the process of building knowledge graphs.
- Then we will show how to prepare training data for GNN link prediction model.
- Finally we will illustrate applications of this model for rewiring knowledge graphs.

Table 2: Scenario 1: Artist Pairs with High Cosine Similarities.

Artist1	Artist2	score
Georges Braque	Pablo Picasso	0.97
Paul Signac	Paul Cézanne	0.93
Paul Cézanne	Claude Monet	0.83
Paul Signac	Paul Klee	0.82
Vincent van Gogh	Claude Monet	0.79
Franz Marc	Vincent van Gogh	0.79
Natalia Goncharova	Vasily Kandinsky	0.79
Paul Cézanne	Paul Klee	0.78
Vincent van Gogh	Paul Gauguin	0.76
Vincent van Gogh	Paul Cézanne	0.7
Vincent van Gogh	Paul Klee	0.7
Marc Chagall	Paul Klee	0.75
Kazimir Malevich	Oskar Kokoschka	0.72
Marc Chagall	Vasily Kandinsky	0.72
Paul Cézanne	Paul Gauguin	0.71
Paul Klee	Paul Gauguin	0.66
Paul Signac	Vincent van Gogh	0.64
Paul Signac	Paul Gauguin	0.63
Claude Monet	Paul Gauguin	0.62
Henri Matisse	Paul Gauguin	0.62
Paul Signac	Claude Monet	0.61
Paul Klee	Claude Monet	0.61
Pablo Picasso	Henri Matisse	0.60

4.1 Data Source

As the data source for this study we used text data from Wikipedia articles about 20 modern art artists - the list of artists is represented in Table 1.

To compare sizes of Wikipedia articles we tokenized text data and calculated counts of words. Based on text size distribution (Table 1), the most well known artist in this list is Vincent van Gogh and the most unknown artist is Franz Marc. The size of Wikipedia article about Franz Marc is less than 10 percent of the size of Wikipedia article about Vincent van Gogh.

4.2 Knowledge Graph on Full Wikipedia Article Text

4.2.1 Building Initial Knowledge Graph

For knowledge graph of the first scenario, as nodes we used artist names and full text of Wikipedia articles and as edges we used connections between artist names and corresponding articles.

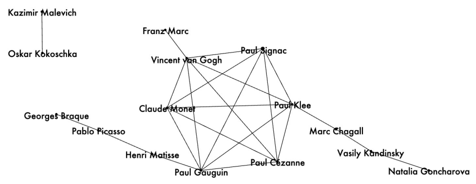

Figure 1: Scenario 1: Rewired Knowledge Graph.

To enrich the graph by adding edges between artists with semantically similar articles, first, we embedded node features, i.e. transformed artist names and full text of Wikipedia articles to vectors.

For text to vector translation we used 'all-MiniLM-L6-v2' transformer model from Hugging Face. As input we used text information for 40 nodes and as a result of node embedding model we received a tensor of size [40, 384], i.e. 40 vectors of size 384.

For pairs of embedded full text nodes we calculated cosine similarity matrix. In that matrix we found 21 node pairs with cosine similarities greater than 0.6 and added corresponding 21 edges to the knowledge graph.

4.2.2 Training GNN Model

As a GNN link prediction model we used a GraphSAGE model from Deep Graph Library (DGL). The model code was provided by DGL tutorial (DGL, 2018) and we only had to transform nodes and edges data from our data format to DGL data format. Coding techniques for data preparation and encoding data to DGL data format are available on our technical blog (sparklingdataocean.com, 2022b).

We used the GNN link prediction Graph-SAGE model with the following parameters:

- 40 nodes: 20 artist names and 20 Wikipedia articles.
- 41 edges: 20 edges between artist names and corresponding Wikipedia articles plus 21 edges on pairs with cosine similarities greater than 0.6.
- PyTorch tensor of size [40, 384] for embedded nodes.
- For GraphSAGE model output vector we selected size 64:

```
model =
GraphSAGE(train_g.ndata['feat']
.shape[1], 64)
```

To estimate the model results we calculated accuracy metrics as Area Under Curve (AUC). The model accuracy metric was about 88.5 percents.

4.2.3 Rewiring Knowledge Graph

To estimate predicted links between artists we looked at cosine similarities for pairs of re-embedded nodes. In the Table 2 you can see pairs of artists with cosine similarities highest scores and in the Table 3 you can see pairs of artists with cosine similarity lowest scores. On Figure 1 you can see graph visualization for pairs of artists with scores more than 0.6.

In Observations subsection of Experiments section we will examine how the results of this scenario can be applied to recommender systems and to graph mining techniques.

More examples and coding techniques are described in our technical blog (sparklingdataocean.com, 2022b).

Table 3: Scenario 1: Artist Pairs with Lowest Cosine Similarities.

Artist1	Artist2	score
Paul Klee	Joan Miró	-0.66
Natalia Goncharova	Claude Monet	-0.64
Pablo Picasso	Paul Signac	-0.63
Paul Signac	Max Beckmann	-0.61
Georges Braque	Paul Signac	-0.57
Claude Monet	Joan Miró	-0.56
Pablo Picasso	Paul Klee	-0.56
Paul Cézanne	Joan Miró	-0.5
Natalia Goncharova	Henri Matisse	-0.56
Natalia Goncharova	Piet Mondrian	-0.55
Pablo Picasso	Paul Cézanne	-0.54
Georges Braque	Franz Marc	-0.52
Kazimir Malevich	Marc Chagall	-0.52
Georges Braque	Paul Klee	-0.51
Paul Signac	Joan Miró	-0.51

4.3 Knowledge Graph on Co-Located Word Pairs

4.3.1 Building Initial Knowledge Graph

The second scenario is based on a knowledge graph that is built on co-located word pairs as nodes and word chains within and across the articles as edges. As we illustrated in Table 1, artists have Wikipedia articles of very different sizes and if we use full

Wikipedia text data, well-known artists, i.e. artists with longest articles will get more word pairs and much more connections than unknown artists.

To balance artist to artist relationship distribution we selected subsets of articles with similar word pair counts. As all selected Wikipedia articles about artists start with high level artist biography descriptions, from each article we selected the first 800 words.

To generate initial knowledge graph we used the following steps:

- Tokenized Wikipedia text and excluded stop words.
- Selected the first 800 words from Wikipedia articles.
- Generated nodes as co-located word pairs.
- Calculated edges as pair to pair neighbors following text sequences within articles.
- Calculated edges as joint pairs that have common words. These edges will represent word chains within articles and connect different articles through word chains across them.
- Built an initial knowledge graph.

Coding techniques for building initial knowledge graph for this scenario are described in our technical blog (sparklingdataocean.com, 2022a).

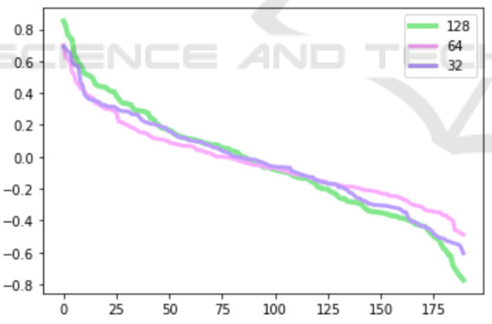

Figure 2: Cosine similarity distributions for GraphSAGE link prediction model outputs of sizes 128, 64 and 32.

4.3.2 Training GNN Model

As a GNN link prediction model we used the same GraphSAGE model as in the first scenario: DGL link prediction model (DGL, 2018). Coding techniques for data preparation and encoding data format to DGL data format are available on our technical blog (sparklingdataocean.com, 2022a).

We used the model with the following parameters:

- 14933 nodes.
- 231699 edges.
- PyTorch tensor of size [14933, 384] for embedded nodes.
- For GraphSAGE model output vector size we experimented with sizes 32, 64 and 128:

```
model =
GraphSAGE(train_g.ndata['feat']
.shape[1], 128)
```

To estimate the model results we calculated Area Under the Curve (AUC) accuracy metrics. Accuracy metrics for models of different output vector sizes are similar and they are represented in Table 4.

Table 4: AUC Accuracy Metrics for GNN Link Prediction Graph-SAGE Model.

Output Vector Size	AUC
32	96.6 percents
64	96.8 percents
129	96.3 percents

4.3.3 Rewiring Knowledge Graph

The results of the GraphSAGE model from DGL library are not actually 'predicted links' but node vectors re-embedded by the model. Those vectors can be used for further analysis steps to predict graph edges.

The results of this scenario are 14933 re-embedded nodes and to detect relationships between artists we calculated average node vectors by artists and estimated link predictions by cosine similarities between them.

As we mentioned above, we experimented with GraphSAGE model output vector sizes of 32, 64 and 128 and compared distributions of cosine similarities between artist pairs.

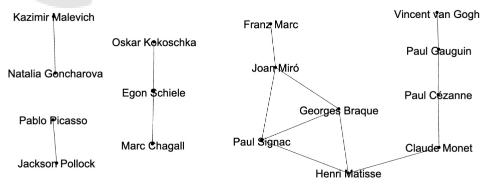

Figure 3: Scenario 2: Rewired Knowledge Graph.

The number of cosine similarity pairs for 20 artists is 190 and the Figure 2 illustrates cosine similarity distributions for model outputs of sizes 128, 64 and 32. For knowledge graph rewiring we selected the model results with output size 128 that reflect a smooth cosine similarity distribution.

In the Table 5 you can see pairs of artists with highest scores of cosine similarities and in the Table 6 - pairs of artists with cosine similarity lowest scores.

On Figure 3 you can see graph visualization for pairs of artists with cosine similarity scores more than 0.5.

More examples and coding techniques are described in our technical blog (sparklingdataocean.com, 2022a).

Table 5: Scenario 2: Artist Pairs with Highest Cosine Similarities.

Artist1	Artist2	score
Paul Signac	Henri Matisse	0.85
Egon Schiele	Marc Chagall	0.82
Paul Cézanne	Paul Gauguin	0.77
Kazimir Malevich	Natalia Goncharova	0.75
Georges Braque	Henri Matisse	0.74
Georges Braque	Joan Miró	0.64
Pablo Picasso	Jackson Pollock	0.62
Georges Braque	Paul Signac	0.59
Paul Signac	Joan Miró	0.58
Vincent van Gogh	Paul Gauguin	0.55
Henri Matisse	Claude Monet	0.52
Paul Cézanne	Claude Monet	0.52
Egon Schiele	Oskar Kokoschka	0.51
Franz Marc	Joan Miró	0.50

4.4 Observations

Node pairs with high cosine similarities, also known as high weight edges, are actively used for graph mining techniques such as node classification, community detection or for analyzing node relationships.

In experiments of this study artist pairs with high cosine similarities can be considered as artist pairs with high semantic relationships through corresponding Wikipedia articles. Some of these relationships are well known: both Pablo Picasso and Georges Braque were pioneers of cubism art movement. Specialists in biographies of Paul Gauguin or Vincent van Gogh will not be surprised to find that these artists had high relationship regardless of different art styles. Some unknown artist semantic connections such as between Egon Schiele and Marc Chagall might be interesting for modern art researchers.

Rewiring knowledge graph and finding high weight links between artists can be applied to recommender systems. If a customer is interested in Pablo Picasso art, it might be interesting for this customer to look at Georges Braque paintings or if a customer is interested in biography of Vincent van Gogh the recommender system can suggest to look at Paul Gauguin biography.

Applications of node pairs with high cosine similarities (or high weight edges) for graph mining techniques are well known: they are widely used for node classification, community detection and so on. On the other hand, node pairs with low cosine similarities (or negative weight edges) are not actively used. Based on our observations, dissimilar node pairs can be used for graph mining techniques in completely different way that similar node pairs or weakly connected node pairs.

For community detection validation strongly dissimilar node pairs act as more reliable indicators than weakly dissimilar node pairs: negative weight edges can validate that corresponding node pairs should belong to different communities.

Graphs with very dissimilar node pairs cover much bigger spaces that graphs with similar or weakly connected node pairs. For example, we found low cosine similarities between key artists from not overlapping modern art movements: Futurism - Natalia Goncharova, Impressionism - Claude Monet and De Stijl - Piet Mondrian.

Links with very low cosine similarities can be used by recommender systems. If a customer is very familiar with Claude Monet's style and is interested in learning about different modern art movements the recommender system might suggest to look at Piet Mondrian's paintings or Natalia Goncharova's paintings.

Table 6: Scenario 2: Artist Pairs with Lowest Cosine Similarities.

Artist1	Artist2	score
Egon Schiele	Henri Matisse	-0.77
Marc Chagall	Henri Matisse	-0.76
Georges Braque	Egon Schiele	-0.74
Kazimir Malevich	Claude Monet	-0.72
Egon Schiele	Paul Signac	-0.70
Marc Chagall	Paul Signac	-0.68
Georges Braque	Marc Chagall	-0.62
Paul Cézanne	Vasily Kandinsky	-0.62
Paul Klee	Joan Miró	-0.59
Natalia Goncharova	Claude Monet	-0.58
Vasily Kandinsky	Claude Monet	-0.56

5 CONCLUSIONS

In this study we propose methods of rewiring knowledge graphs to detect hidden relationships between graph nodes by using GNN link prediction models.

In our experiments we looked at semantic similarities and dissimilarities between biographies of modern art artists by applying traditional and novel methods to corresponding Wikipedia articles. Traditional method was implemented on full test of articles and cosine similarities between re-embedded nodes.

The novel method was constructed based on distribution of co-located words within and across articles. The output vectors from GNN link prediction model were aggregated by artists and link predictions were estimated by cosine similarities between them.

We explored advantages for graph mining techniques of using not only highly connected node pairs but also highly disconnected node pairs.

We denoted that level of disconnected word pairs can be used to define boundaries of a space covered by knowledge graph: existence of node pairs with very low cosine similarities shows that a graph covers much bigger space than a graph with only high and medium cosine similarities. Also highly disconnected node pairs are good indicators for validation of community detection.

We demonstrated applications of rewired knowledge graphs for recommender systems. Based on high similarity pairs recommender systems can suggest to look at paintings on biographies of artists that are similar to the artist of interest. Based on high dissimilarity pairs recommender systems can advice to look at very different art movements.

REFERENCES

Bradley, A. (2017). Semantics conference, 2017.

Bronstein, M., Bruna, J., Cohen, T., and Veličković, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges.

DGL (2018). Link prediction using graph neural networks.

Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. *In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*.

Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*.

Krizhevsky, A., Sutskever, I., and Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, vol. 521, no. 7553, pp. 436–444.

Noy, N., Gao, Y., Jain, A., Narayanan, A., Patterson, A., and Taylor, J. (2019). Industry-scale knowledge graphs: Lessons and challenges. In *acmqueue*.

Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deep-walk: Online learning of social representations. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Romanova, A. (2020). Building knowledge graph in spark without sparql. *CCIS*, vol. 1285, pp 96–102.

sparklingdataocean.com (2022a). Find semantic similarities by gnn link predictions.

sparklingdataocean.com (2022b). Rewiring knowledge graphs by link predictions.

Wang, M., Qiu, L., and Wang, X. (2021). A survey on knowledge graph embeddings for link prediction. In *Symmetry*.

Wang, X. and Vinel, A. (2021). Benchmarking graph neural networks on link prediction.

Wu, H., Song, C., Ge, Y., and Ge, T. (2022). Link prediction on complex networks: An experimental survey. In *Data Science and Engineering*. Springer.

Zhou, T. (2021). Progresses and challenges in link prediction.

Zhou, T., Lu, L., and Zhang, Y.-C. (2009). Predicting missing links via local information. In *Eur. Phys. J. B 71 (2009) 623-630*.