Link Prediction using Graph Neural Networks for Master Data Management

Balaji Ganesan
IBM Research, India
bganesa1@in.ibm.com

Gayatri Mishra
IBM Data and AI
gayamish@in.ibm.com

Srinivasa Parkala
IBM Data and AI
shsriniv@in.ibm.com

Neeraj R Singh
IBM Data and AI
sneeraj@in.ibm.com

Hima Patel
IBM Research, India
himapatel@in.ibm.com

Somashekar Naganna
IBM Data and AI
soma.shekar@in.ibm.com

ABSTRACT

Learning graph representations of n-ary relational data has a number of real world applications like anti-money laundering, fraud detection, risk assessment etc. Graph Neural Networks have been shown to be effective in predicting links with few or no node features. While a number of datasets exist for link prediction, their features are considerably different from real world applications. Temporal information on entities and relations are often unavailable. We introduce a new dataset with 10 subgraphs, 20912 nodes, 67564 links, 70 attributes and 9 relation types. We also present novel improvements to graph models to adapt them for industry scale applications.

1. INTRODUCTION

Relational Data consists of tuples where each tuple is a set of attribute/value pairs. A set of tuples that all share the same attributes is a relation. Such relational data can be presented in a table, json arrays among other forms.

In Master Data Management, one or more tuples in relational data can be resolved to an Entity. Typically, an Entity in this setting is a person or an organization, but there can be other types too. These entities may share explicit and implicit relations between them. A common way to represent these entities and relationships is a graph, where each entity is a node, and the relationships are links (edges) between the nodes.

Link Prediction is the task of finding missing links in a graph.

Master Data Management (MDM) includes tasks like Entity Resolution, Entity Matching, Non-Obvious Relation Extraction. While the enterprise customer data is predominantly stored as relational data, Graphs stores are widely used for visualization and analytics use-cases. In recent years, Graph Neural Networks are being used for link prediction and node classification on such graphs.

Graphs in Master Data Management could be considered as property graphs which differ from Knowledge Graphs and Social Networks. In particular, MDM is focused on entities like people, organization, and location, which constitute nodes in the graph. In this work, well focus on Master Data Management on People Graphs. Other entities like numerical ids, demographic information, business terms constitute the attributes of the nodes. Links between people in the real world constitute the edges of the graph. The type of people to people relations form the link type and details of the relationship like duration constitute the edge properties.

Given this organization of Master Data, Link Prediction on people graphs presents a unique set of challenges in terms of model performance, data availability, fairness, privacy, and data protection. Further, Link Prediction among people has a number of societal implications irrespective of the use-cases. While link prediction of financial fraud detection, law enforcement, and advertisement go through ethical scrutiny, the recent use of social networks for political targeting and job opportunities also require ethical awareness on the part of the model developers and system designers. In this work we discuss a number of use-case for link prediction on property graphs, their specific dataset requirements and practical insights in designing the infrastructure for industrial scale deployment.

Throughout this paper, well discuss the creation of a synthetic dataset in the context of a number of tasks, namely link prediction, node classification, human in the loop verification, explainability, temporal graphs, and non-obvious links prediction.

Dataset requirements

- Attributes that capture demographic data should be present.

- Predicted links and the node classes should be verifiable and explainable.

- The graph should be of sufficient size, sparsity and connectedness to capture the real world.
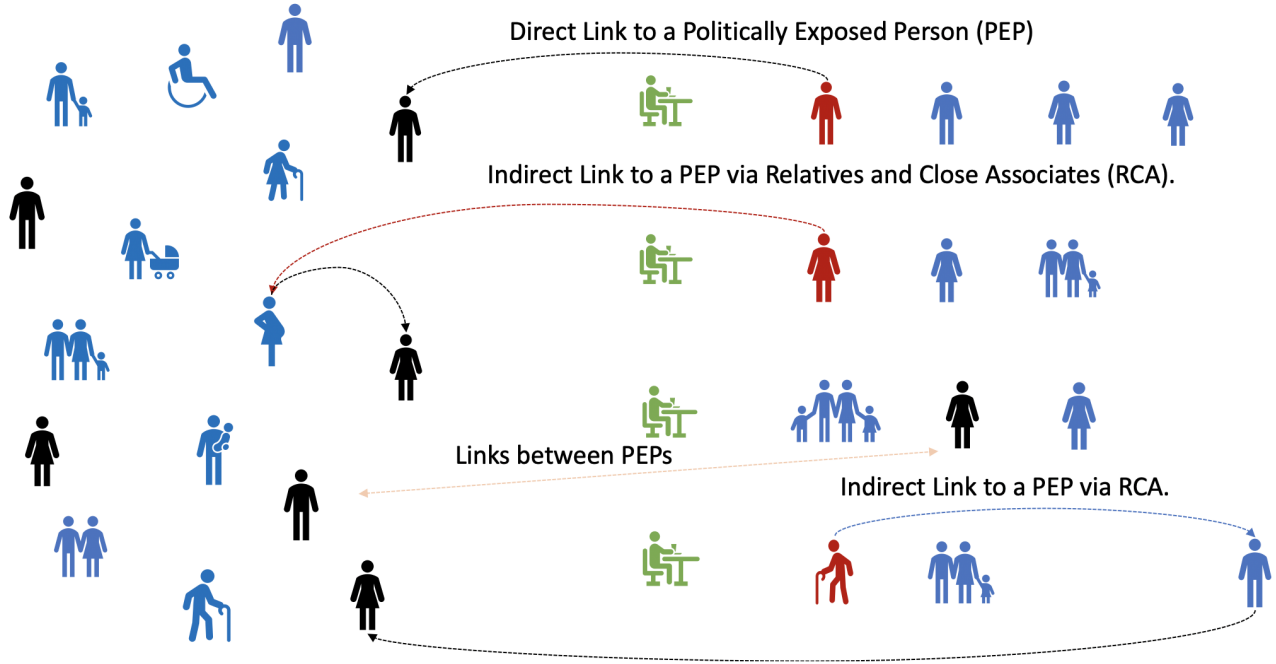
Figure 1: Link Prediction in Property Graphs

- The graph should have temporal information.

- Graph should support provenance and lineage data.

The main contributions of this work are:

1. We introduce a new dataset similar to the PPI graph dataset, for property graphs on people which satisfies the requirements of many real-world applications in Master Data Management. 2. We present our experiments to train GNNs on similar sub-graphs and infer on unseen subgraphs.

2. RELATED WORK

[10] showed that a Relational Graph Convolutional Network can outperform direct optimization of the factorization (ex: DistMult). They used an autoencoder model consisting of (1) An encoder an R-GCN producing latent feature representations of entities. (2) a decoder a tensor factorization model exploiting these representations to predict labeled edges.

Since then, several graph embedding methods have been proposed including node2vec, DeepWalk, LINE, MNMF etc. These graph embedding methods outperform heuristic based approaches like Adamic Adar, Common Neighbors, Jaccard Index. But these heuristic based approaches remain popular on commercial solutions like Neo4J.

Recently, libraries for generating graph embeddings on large graphs and in enterprise settings have been made publicly available. PyTorch BigGraph provides several knowledge graph embedding methods including TransE, RESCAL, DistMult and ComplEx.

[7] introduced GraphSAGE (SAmple and AggreGatE) an inductive framework that leverages node feature information (ex: text attributes, node degrees) to efficiently generate node embeddings for previously unseen data or entirely new (sub)graphs. In this inductive framework, we learn a function that generates embeddings by sampling and aggregating features from a nodes local neighborhood.

[14] proposed a Position Aware Graph Neural Network that significantly improves performance on the Link Prediction task over the Graph Convolutional Networks.

[5] introduced a dataset based on Wikidata. We have incorporated Wikidata in our dataset too. Except that Wikidata does not have contact details and can be incomplete like the DBPedia (UDBMS) dataset.

[3] also introduced a dataset based on Wikidata. Again, for afore mentioned reasons, its not different from DBPedia dataset.

[15] introduced the Protein-Protein Interaction dataset which has been used in a number of recent works in graph neural networks. [cite] presented Graph Convolutional networks. [7] introduced the idea of inductive representation of nodes in a graph. [14] added anchor nodes to improve the representation of nodes in the graph.

[cite] introduced many aspects of entity resolution using a probabilistic matching engine. DeepMatcher [9] presents a neural model for Entity Matching.

[13] proposed GNNExplainer for explaining link prediction.

Existing Datasets

The Protein-Protein Interaction (PPI) dataset introduced by [15] consists of 24 human tissues and hence has 24 subgraphs of roughly 2400 nodes each and their edges. Having similar subgraphs helps to average the performance of the model across subgraphs. This is a desirable property we wish to retain in our dataset.

Wikipeople dataset introduced by [5] is based on the publicly available Wikidata and hence can be used without violating the privacy of the people in the resulting graphs.
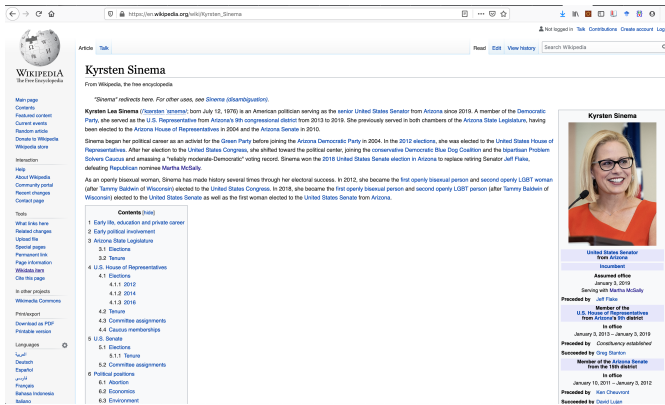
(a) Wikipedia Text
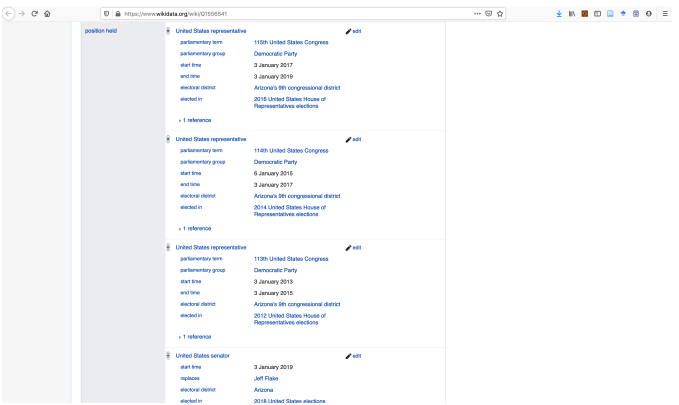
(b) Wikidata page

Figure 3: Crawled page

However, Wikidata, Wikipedia and crowd sourced data in general have the false False positives problem that we discuss in section 5.2. The general idea is that if we tr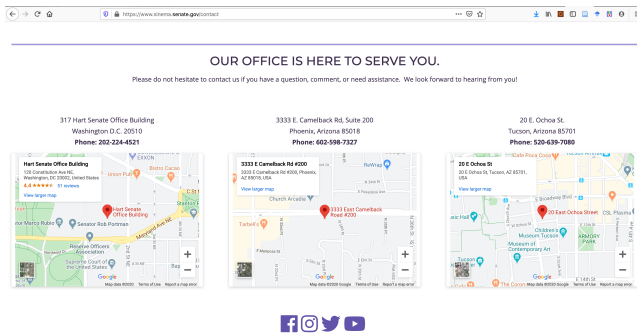ain a model on such crowd sourced data and the model predicts new links, we will not know if the predicted links are missing links correctly predicted by the model or false positives. Manual evaluation of the correctness of these models is not possible at scale. Hence we need automated, even if approximate methods to verify the links proposed by these models.

Open Graph Benchmark [8] initiative by SNAP Group at Stanford is trying to come up large benchmark datasets for research in Graph Neural Models. [1] is an effort to create a database of Politically Exposed Persons in Ukraine.

3. POLITICALLY EXPOSED PERSONS

[4] define politically exposed persons, or PEPs, to denote individuals who are or have been entrusted with prominent public functionssuch as heads of state or governmentand their family members and close associates. It is commonly used in the context of anti-money-laundering standards that target the laundering of proceeds of corruption.

[6] by FATF expanded the definition of PEP to relatives and close associates (RCAs) of the individuals listed above, who may become involved in financial crimes like money laundering because of family, professional, and social connections.

In this section, we present few applications of link prediction that necessitate the creation of a new dataset. These are real-world applications where Master Data Management solutions are already being used on large graphs (few million nodes to close to a billion nodes). However, the dataset that we present is a reasonably small graph that can be used to experiment on a single GPU. We leave distribute learning on large graphs as future work.

PEP Links

Given a list of people or organizations (nodes) on the watchlist, predict whether a new node (or a subgraph) is linked to any of the nodes on the watchlist. For example, an airline may want to know if a customer at the checkin or boarding gate is related to a person on the watch list. PEP a company may want to know if a vendor is related to a Politically Exposed Person (PEP). We'll use this application throughout this work.

Link Type Prediction

While type of the relation is very common in Relation Extraction datasets, a number of graph datasets do not have link types. Being able to predict the type of a link (householder, tenant, colleague, contract parties), has a number of enterprise applications. While DBPedia and Wikidata based relations have a number of relation types, the number of such relation types is limited. Some of the link types like a householder who is not family member are hard to predict but are often needed in enterprise deployments. We however do not attempt to generate training data for such link types in this work.

Explainability

Explainability methods in Graph Neural Networks tend to follow similar methods in text and images, namely identifying features that are most significant for the predictions. However, in enterprise applications, there is a need for explanations for non-technical users. Path-based explanations that help to visualize the neighborhood of the predicted links will require training data with medium sized and multiple paths between different people, as is common in the real world. The popular six degrees of separation motivates having an average path length of 6 between nodes of a graph.

Temporal Graph Embedding

[3] introduced a new dataset for their work on temporal graph embeddings. Use-cases in financial fraud prevention

```
Kyrsten Sinema  wiki_title      Kyrsten Sinema
Kyrsten Sinema  wiki_url        https://en.wikipedia.org/wiki/Kyrsten_Sinema
Kyrsten Sinema  name    Kyrsten Sinema
Kyrsten Sinema  subgraph        US
Kyrsten Sinema  birth_year      1976
Kyrsten Sinema  educated_at     ASU College of Public Service & Community Solutions
Kyrsten Sinema  educated_at     Arizona State University
Kyrsten Sinema  educated_at     Brigham Young University
Kyrsten Sinema  educated_at     Sandra Day O'Connor College of Law
Kyrsten Sinema  nationality     United States
Kyrsten Sinema  occupation      athlete
Kyrsten Sinema  occupation      worker
Kyrsten Sinema  occupation      academic
Kyrsten Sinema  occupation      lawyer
Kyrsten Sinema  occupation      faculty
Kyrsten Sinema  occupation      politician
Kyrsten Sinema  occupation      Sports
Kyrsten Sinema  occupation      Lawyer
Kyrsten Sinema  gender  Female
Kyrsten Sinema  religion        Christianity
Kyrsten Sinema  political_party Democrats
Kyrsten Sinema  position_held   senator
Kyrsten Sinema  sexual_orientation      LGBT
Kyrsten Sinema  location        the United States
Kyrsten Sinema  ideology        feminism
Kyrsten Sinema  ideology        socialism
Kyrsten Sinema  state   Arizona
Kyrsten Sinema  relation        Jeff Flake
Kyrsten Sinema  relation        Tammy Baldwin
Kyrsten Sinema  relation        Ralph Nader
Kyrsten Sinema  relation        David Lujan
Kyrsten Sinema  relation        Bob Thomas
Kyrsten Sinema  relation        Ed Pastor
Kyrsten Sinema  relation        David Schweikert
Kyrsten Sinema  relation        Vernon Parker
Kyrsten Sinema  relation        David Schapira
Kyrsten Sinema  relation        Andrei Cherny
Kyrsten Sinema  relation        Harry Mitchell
Kyrsten Sinema  relation        Wendy Rogers
Kyrsten Sinema  relation        Barack Obama
Kyrsten Sinema  relation        Dave Giles
Kyrsten Sinema  relation        Ed Buck
Kyrsten Sinema  relation        Nancy Pelosi
Kyrsten Sinema  relation        Donald Trump
Kyrsten Sinema  relation        Justin Amash
Kyrsten Sinema  relation        Bill Clinton
Kyrsten Sinema  relation        Jay Goodlik
Kyrsten Sinema  relation        John Katko
Kyrsten Sinema  relation        Blake Dain
Kyrsten Sinema  relation        Jeff Merkley
Kyrsten Sinema  relation        Thomas Charles O'Halleran
Kyrsten Sinema  relation        Greg Stanton
Kyrsten Sinema  relation        Bill Cassidy
Kyrsten Sinema  last_name       Sinema
Kyrsten Sinema  first_name      Kyrsten
```

Figure 4: Extracted Triples for the example Node

require a dataset where the demographic attributes of a person change, sometimes quite drastically. Even otherwise, changes to attributes of people nodes is quite common in the real world (people moving residences, jobs, change in marital status or partners). Hence well propose a dataset that captures events in the life of people.

Provenance and Lineage

A closely related capability in enterprise graphs is the ability to maintain lineage and provide provenance of nodes, attributes and relations. On Wikipedia based datasets, we can use the edit history to capture some of the changes to the entity (Person) described in the Wikipedia page.

Verifiability

In addition to Explainability, it is desirable to have additional data that can be used by Data Stewards to verify the predicted links. This idea is similar to the efforts in fact verification task. A piece of text that strengthens a predicted link, a heuristic measure like Jaccard coefficient, a measure of similarity (or lack thereof) between nodes involved in a link can all be used to verify the predicted links.

Fairness

Several use-cases where link prediction is used on people graphs have significant societal impact and require an acute awareness of privacy, fairness and data protection. A dataset with reasonable diversity and fairer representation of underprivileged groups is also a desired requirement in our dataset. [12] present methods to measure and sample the dataset. These methods in turn have been based on the AIF 360 toolkit [2].

4. PEP LINKS DATASET

In this section, we'll present our method to create a PEP Links dataset using the publicly available information about political leaders and those linked to these leaders. A similar approach can be used to generate datasets for other kinds of links.

Typically, data about people in the Master Data of a company can come from multiple sources before they are combined to form the master data. As shown in Figure 2a, we create the PEP Links Dataset from 3 sources, namely the Wikipedia page of a person, their corresponding Wikidata page and their web pages crawled by us. Within Wikipedia, we can extract entities, attributes and relations both from the unstructured text data and from the Wikipedia API. We use the method described in [?] to extract information from unstructured text.

Desired Dataset Characteristics

- Personal data should be there (name, email, address, even SSN if it can be generated artificially).

- The ratio of nodes and edges should be configurable or typically be 1:5. UDBMS (DBPedia) and many other datasets have lot less edges than nodes.

- For non obvious link prediction. the dataset requires several hops.

- The Entity re-resolution problem will need a dataset where entities like US President change or a persons job_title changes. This can also be attempted in the PEP Dataset.

- People can have different names (legal name, business name etc), addresses (permanent, business, contact etc), phone numbers.

- The number of attributes is above 30 on average. The dataset they use for scale testing has 145 attributes.

- The datasets they work with have millions of rows.

- Ids are weighted higher than other columns. (We now use Wikidata_id as unique id in people_dataset).

[15] introduced the Protein Protein Interaction dataset that has 20 subgraphs each representing a human tissue. Having similar subgraphs presents a number of advantages over other graphs.

We start with the idea of creating a property graph of people consisting of multiple similar subgraphs albeit with different sizes. As we shown in experiments in [ref section], this allows us to train and infer on different subgraphs without affecting the connected-ness and hence node representation in the original data.

We have created this dataset from the Wikipedia, Wikidata and the web pages of elected representatives. We obtained the names of 3983 elected representatives from the listings of 10 legislatures. These listings provide the names of the elected representatives and other details like contact
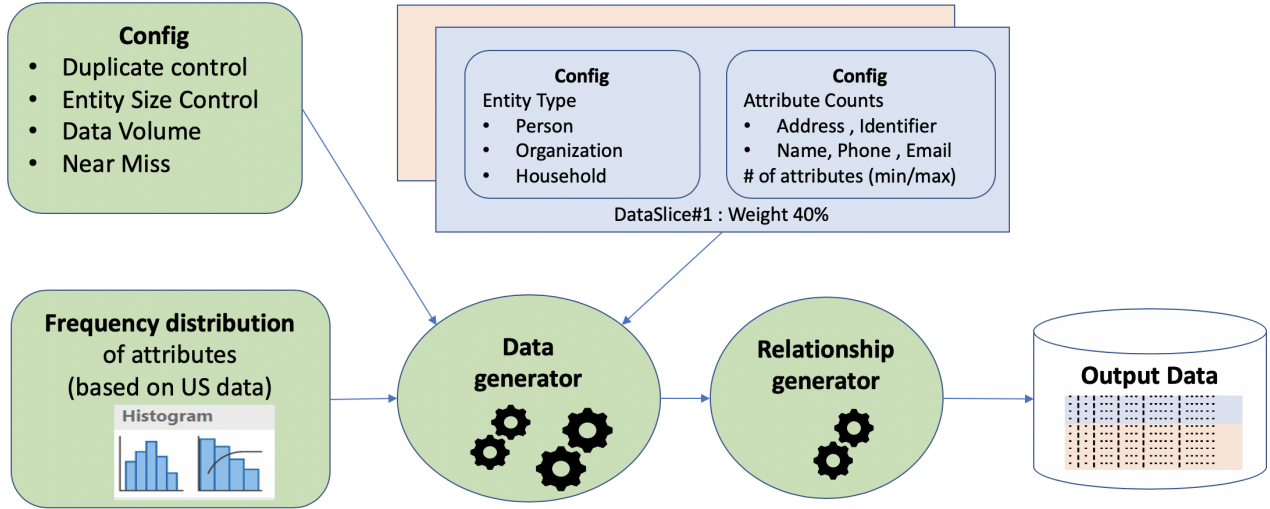
Figure 5: MDM Data Generator

information. However this semi-structured data by itself cannot be used for training a neural model on unstructured data.

Hence, we first obtained the Wikipedia pages of elected representatives. We used the method described in [cite].

```
|
Comparing proband 10047 with candidate 10048, score= 7.2

XNM: BURNS|JOSH|            -- BURNS|JOSH|                , wgt= +6.90, mcc=E
PADDRESS: VIC|              -- 219|BARKLY|ST|ST|KILDA|V, wgt= -0.04, mcc=D
BADDRESS:                   --                          , wgt= +0.00, mcc=M
SSN:                        --                          , wgt= +0.00, mcc=M
DOB:                        --                          , wgt= +0.00, mcc=M
SEX: M                      -- M                        , wgt= +0.35, mcc=E
SIN:                        --                          , wgt= +0.00, mcc=M

XNM Detail:
  XNM[01]: BURNS            -- BURNS                    , wgt= +3.39, mcc=X
  XNM[02]: JOSH             -- JOSH                     , wgt= +4.26, mcc=X


PADDRESS Detail:
  PADDRESS[01]: VIC         -- VIC                      , wgt= +8.71, mcc=X
```

Figure 6: Entity Resolution using Infosphere MDM

4.1 MDM Data Generator

In the context of MDM, an entity is quite often a person or an organisation. The key attributes that define a person entity include fields like first name, last name, gender, date of birth, residential address, phone number, identifiers like SSN number, passport number etc. In addition there are fields like references or links to other entities which could be other person entity, an Org entity or a contract entity.

For the purpose of testing functional correctness, performance evaluations, evaluation and training of Deep Learning/AI models, we need real looking datasets. Obviously, such dataset should not include real life data for the reasons like privacy and confidentiality, which we have utmost respect for. This is also a legal requirement in many countries. Hence we needed a way to generate the real-looking synthetic data.

We developed an in-house tool that is designed to provide data as per requirements. The names and addresses are as one would typically observe in the US population. The frequency distribution of various tokens are taken into account to simulate the real life scenario. We introduce near-misses in attributes to simulate typos. The synthetic data also contains predefined number of suspected duplicates which are created with partial set of attributes matching with that of original entity such that the size of entities distributed as per Zipf's law.

In addition to the attributes for an entity, the synthetic data also contains stated relationships between entities. The data provides the degree of separation between given entities, which can be used to evaluate the capability of the Deep Learning/AI model to find remotely-related entities.

4.2 Entity Resolution

Entity resolution is being done by using the Probabilistic Matching engine. Probabilistic matching measures the statistical likelihood that two records are the same. By rating the degree of matching of the two records, the probabilistic method is able to find non-obvious correlations between data. With PME we first need to bootstrap the database before data can be loaded. Bootstrapping is the process of creating the core database tables that the PME will reference.

- Step 1 - Standardize: Optimizes data for statistical comparisons Normalizes & compacts data, creates derived data layer, source data remains intact Phonetic equivalences, tokenization, nicknames, etc.

- Step 2 - Bucketing: Finds all the potential matches Casts a wide net all matches on current or historical attributes, prevents misses Partial matches, reversals, anonymous values, etc.
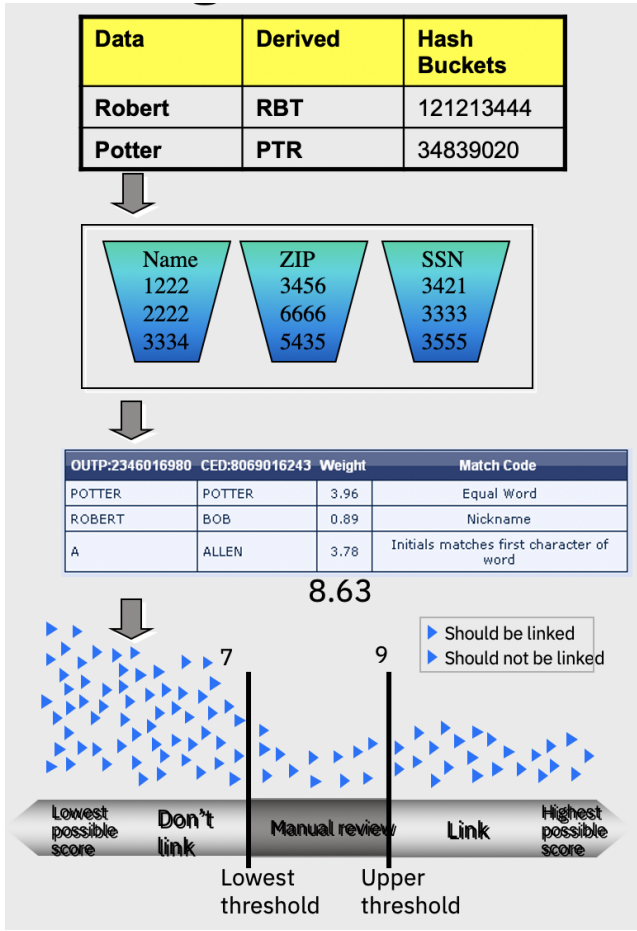
Figure 7: Probabilistic Matching Engine

- Step 3 - Compare: Scores accurately via probabilistic statistics Compares attributes one-by-one and produces a weighted score (likelihood ratio) for each pair of records Frequency weights specific to your business Edit distance, proximity of match

- Step 4 - Score: Custom threshold settings Single or dual threshold models Link, dont link, dont know

Configure Data Model - The PME member model defines the way that the data is stored managed, and validated . For our data set we are using a predefined dictionary offered by PME.

Configure Algorithm - An algorithm will need to be configured to address the attributes we are using, the comparisons that we would like to use, and the bucketing strategy that would be employed. We are using the predefined algorithm

Deploy Instance - Setting up the PME instance involces 3 steps. Here we a.) Create an empty database. b.) Create hub instance. c.)Bootstrap database.

Deploy configuration - Once we have the entire model ready , the model needs to be deployed into the PME engine .The dictionary tables control validation rules, application properties, attributes, sources, nicknames, and the core algorithm settings. The dictionary can be populated by using a combination of PME engine utilities and Workbench jobs.
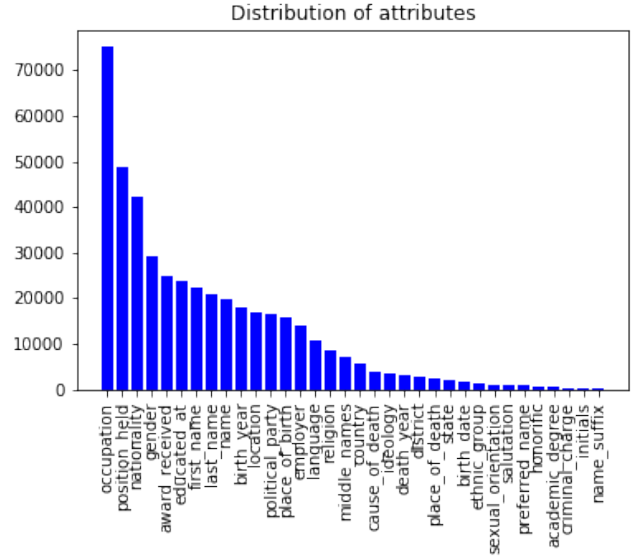

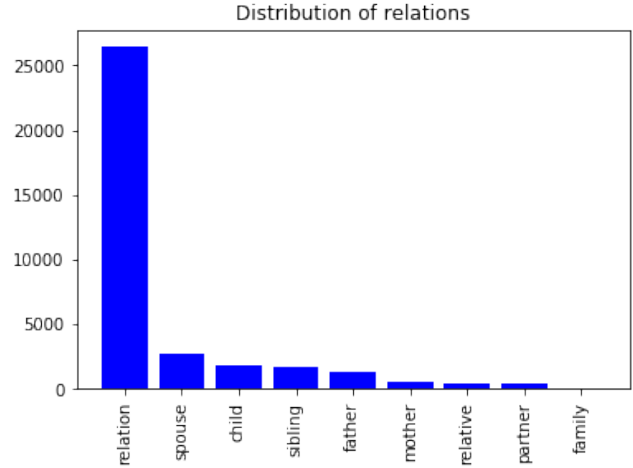Figure 8: Distribution of node attributes

Figure 9: Distribution of relation types

Derive Data - Derived data is essentially data that has been processed by the algorithm. The data derivation process includes four main events: 1. Raw data is parsed into segment specific unload files 2. Comparison strings are built from standardized data 3. Members are assigned bucket hashes 4. Binary files are created for faster computation

Generate weights - The weight generation process is an integrated utility that goes through multiple steps to measure the frequency of individual values in the database, then assigns weights to those values the most common weighing less and the most rare weighing more. The weight generation process creates unload files and loads them into the database.

Bulk Cross Match - The bulk cross match (BXM) is a process that allows to compare and link thousands of records per second. The BXM process is made up of two primary jobs; Compare Members in Bulk (mpxcomp) and Link Entities (mpxlink). After running the compare and link, the data will need to be loaded into the database

Applying the above PME algorithm on our example data, we get the score shown in Figure 6.

We'll now analyze the PEP Links dataset that we have created in the previous section to see if the desired characteristics mentioned in Section 4 are satisfied.

4.3 Diversity of the dataset

As shown in Figure 8 and Figure 9, we have generated a medium sized graph, which can be used to conduct experiments on a single GPU. It is fairly simple to expand this dataset both by adding more seed nodes (PEPs) and by adding links from more hops.

	PEP Links Dataset
Persons	20912
Links	67564
Entities	1681747
Attributes	70
Relations	9

Table 1: Statistics on personal data annotations

Fleischmann is *married to* Brenda M. Fleischmann . Max Fleischmann <spouse> Brenda M. Fleischmann

Walden and *his wife* , Mylene , live in Hood River with their son Anthony . Paul E. Walden <spouse> Mylene

Pugh and *her husband* John live on their farm at Turiwhate , near Kumara . Pugh <spouse> John

O'Neil lives with *her partner* Brendan , an anaesthetist , and their young son . O'Neil <partner> Brendan

Figure 10: Text to verify the links predicted by our model

As shown in Figure 4, the merged dataset obtained from three sources is much richer than the one extracted from just the Wikipedia text.

5. PEP LINKS PREDICTION

5.1 Verifiability

One of the desirable properties of the dataset that we set out to create is the ability to verify the links present in the dataset.

[11] introduced the FEVER dataset for fact verification. We follow a similar method in our work to generate verifiable text that a human annotator to decide if the predicted link is valid or otherwise.

While in this dataset, we use text from Wikipedia pages, we can also include links from other sources namely organization charts, logs, emails and other enterprise documents to generate such verification text. The verifiable text for our example is as shown in Figure 10.

5.2 'False' false positives

When we have an incomplete graph, and the link prediction model predicts a new link not present in the original graph, it is not obvious if the predicted link is a false positive or new link that we wish to predict. Graphs created from Wikipedia and DBPedia especially suffer from this problem, because the Wikipedia pages and relations may not have been created by volunteers. Even in enterprise graphs, the new links predicted may have to be verified by some process, to be not considered as false positives.
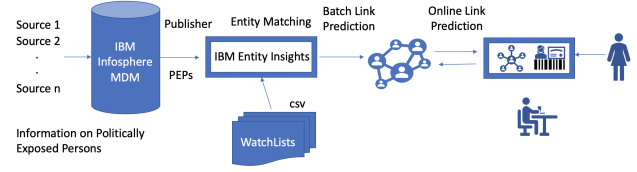

Figure 11: Predicting Links to PEPs

To explain this 'false' False Positives problem, we present the results of our Link Prediction experiments on the UDBMS dataset. The results are shown in Table 3.

```
Number of Nodes: 510733
Number of Edges: 28945
Node: max 335, min 11, mean 23.79
Edge: max 888, min 20, mean 62.74
Anchor num 64, Batch size 8
Number of Subgraphs = 194 (ignored less than 10 nodes)
```

As can be seen in Table 3, the AUC and Accuracy scores are do not account for the incomplete graph. They assume the negative samples to be not linked, because those nodes are not linked to each other in the dataset, but this could be just because of the incompleteness of the crowd sourced DBPedia content. As model predicts links on these negative samples, the AUC* and Accuracy* will suffer. One way to handle this is to just report the positive samples accuracy - the ability of the model to predict the links that were removed by us for testing. We can also report the positive predictions on the negative samples. In this UDBMS dataset, there seems to be substantial amount of new links predicted which were not present in the original dataset. This could either mean, the model has low precision or it could mean the underlying graph is incomplete. Considering there are only around 28k edges for 510k nodes, there may as well be a significant number of edges missing in this dataset. This is one of the prime motivations to create a new dataset for our experiments.

5.3 Training on similar subgraphs

Typical Link Prediction experiments tend to split the training data into train, test and validation sets by removing some existing links. Training models by removing some existing links, could potentially affect the performance of the models.

Like in the PPI dataset, we have generated multiple similar subgraphs using the elected representatives of a country (and EU) as the starting set of nodes. From these nodes (our PEPs), we follow the links in a breadth first fashion to add nodes linked to the PEPs. There are two advantages because of this dataset arrangement. Like in the original PPI graph, we can average the performance across the different subgraphs. Further, this presents us the option to experiment training on the other subgraphs, and testing on a hold out subgraph. This can be for k-fold validation on the subgraphs or for inferring links on a new subgraph with similar characteristics.

5.4 Watch List Application

The PEP Links Prediction is a typical usecase in Master Data Management as shown in Figure 11. When the master

	Large Subgraphs	Large Subgraphs	All	All
	No Features	38 features	No Features	38 Features
AUC*	0.5578	0.5615	0.3801	0.3888
Accuracy*	0.42714	0.4286	0.3599	0.4039
Positive Samples Accuracy	0.6684	0.7360	0.4996	0.4460
Positive Predictions on Negative Samples	0.8409	0.8709	0.6894	0.5300

Table 2: Link Prediction on UDBMS Dataset

data is initially created or updated in batch, there is a need for batch mode link prediction. Later when the model is deployed, and a new node or an update to nodes and edges is performed, there is a need for online link prediction. Considering an enterprise property graph is likely to be updated quite frequently, the model needs to be frequently re-trained which is expensive or we have to accept the drift in model performance. The inductive approach to learning the node representations in GNNs helps to make the models more robust to unseen data, but the inductive approach will not be sufficient if the distribution of the unseen nodes is different from the graph created in batch mode.

Dataset	Model	ROC AUC	Std. Dev.
UDBMS	**GCN**	0.4689	0.0280
	P-GNN	0.6456	0.0185
PEP Links	**GCN**	0.4047	0.09184
	P-GNN	0.6473	0.02116

Table 3: Comparison of Link Prediction on the UDBMS and PEP Links datasets

6. CONCLUSION

We introduced a new dataset for the Link Prediction task on Property Graphs and described our method to create the dataset. We presented a number of motivating experiments and use-cases, which benefit from such a dataset. We then presented two improvements to the state of the art in Link Prediction using Graph Neural Networks, namely providing a text for verifying the predicted links and training models on similar subgraphs without dropping existing links that shows a marked improvement in performance.

Acknowledgements

7. REFERENCES

[1] Ukraine pep dataset. https://pep.org.ua/en/, 2019.
[2] R. K. Bellamy, K. Dey, M. Hind, S. C. Hoffman, S. Houde, K. Kannan, P. Lohia, J. Martino, S. Mehta, A. Mojsilovic, et al. Ai fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias. *arXiv preprint arXiv:1810.01943*, 2018.
[3] S. S. Dasgupta, S. N. Ray, and P. Talukdar. Hyte: Hyperplane-based temporally aware knowledge graph embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2001–2011, 2018.
[4] T. S. Greenberg, L. Gray, D. Schantz, C. Gardner, and M. Latham. *Politically exposed persons: A guide on preventive measures for the banking sector*. The World Bank, 2010.
[5] S. Guan, X. Jin, Y. Wang, and X. Cheng. Link prediction on n-ary relational data. In *The World Wide Web Conference*, pages 583–593, 2019.
[6] F. Guidance. Politically exposed persons (recommendations 12 and 22). *Financial Action Task Force Guidance*, 2013.
[7] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.
[8] J. Lu, J. Chen, and C. Zhang. Open graph benchmark. https://ogb.stanford.edu/index.html, 2019.
[9] 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 *Proceedings of the 2018 International Conference on Management of Data*, pages 19–34, 2018.
[10] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.
[11] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal. Fever: a large-scale dataset for fact extraction and verification. *arXiv preprint arXiv:1803.05355*, 2018.
[12] L. S. Vannur, L. Nagalapatti, B. Ganesan, and H. Patel. Data augmentation for personal knowledge graph population, 2020.
[13] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. Gnn explainer: A tool for post-hoc explanation of graph neural networks. *arXiv preprint arXiv:1903.03894*, 2019.
[14] J. You, R. Ying, and J. Leskovec. Position-aware graph neural networks. *arXiv preprint arXiv:1906.04817*, 2019.
[15] M. Zitnik and J. Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):190–198, 2017.