# Improving Embeddings by Augmenting DBPedia with Results of SPARQL Queries

Tim de Boer* VU Amsterdam

Michael Cochez (supervisor) † VU Amsterdam

**ABSTRACT**

Data represented in Knowledge Graphs (KGs) is graph-based. However, machine learning (ML) algorithms expect data to be in propositional form. Therefore, converting nodes in KGs to low-dimensional feature vector representations is needed. This is done by embedding methods such as RDF2Vec. Unfortunately, information is lost by embedding the data. In this study, a basic method is investigated to add additional information by augmenting the graph with triples resulting from SPARQL queries requested by users. The idea is that important nodes would be requested more often in SPARQL queries, and therefore would be augmented more often, which in turn will add information about importance to the graph. The method was tested for a subset of the DBPedia 2016-10 KG by measuring performance of embeddings generated by RDF2Vec for the downstream ML tasks classification and regression, as evaluated by an extensive framework. Results show no improvement in both ML tasks. A more advanced augmentation method is proposed for future research to investigate if a more advanced method would give better performance for downstream ML tasks when compared to the basic method.

## 1 INTRODUCTION

Knowledge graphs (KGs) provide a structured representation of multi-relational data with labeled edges (relations) between nodes (entities). Large-scale, cross-domain KGs such as DBPedia [1], have been used to augment datasets with features taken from KGs, which in many cases have been shown to improve results of data mining problems [2, 3].

Data represented in KGs is graph-based. Graph data consists of triples *(h, r, t)*, denoting that there exists a relationship *r* between entities head *h* and tail *t*. However, most data mining and machine learning algorithms expect data to be in propositional form, where each instance is represented as a vector of binary, numerical or nominal features $< f_1, f_2, ..., f_n >$. Thus, for accessing the information stored in KGs for data mining problems, transformations have to be performed to create propositional feature vectors from graph data.

Converting nodes in KGs to low-dimensional feature vector representations have become popular with various embedding methods such as TransE [4], ComplEx [5] and RDF2Vec [6]. Whereas these methods have been shown to be reasonably successful in preserving some of the rich information of the graph as low dimensional vector space embeddings, not all information is preserved.

In this paper, we propose the idea of adding additional information about the importance of certain nodes and edges to KGs, as means of preserving the more important information in embeddings. We will explore a basic method of adding this importance information by investigating result sets of the KG from user requested SPARQL querries. The idea is that important nodes would be requested more often in SPARQL queries, and therefore would be augmented more often, which in turn will add information about importance to the graph. To our knowledge, this idea has not been implemented in earlier work. The idea is tested with the RDF2Vec embedding method by measuring performance on the machine learning (ML) tasks classification and regression.

The remainder of this paper is structured as follows. We first give an overview of related work. In section 3, we give an overview of the data used for this project. In section 4, we present our approach for extracting, simplifying and transforming complex SPARQL queries to queries with only Basic Graph Patterns (BGPs), so these queries were useable for further steps. In section 5, we explain how we have used of the simplified queries by instantiation of the result sets to create new triples on top of original triples to add to the original KG, with as purpose to add information about importance of original nodes and edges. In section 6, we present the approach of the RDF2Vec embedding method. This is followed by the results of an extensive evaluation on downstream ML tasks in section 7 for the original and augmented graph. We end with a discussion of the results, a summary and an outlook on future work.

## 2 RELATED WORK

In recent years, various methods have been brought up to improve embedding of graph data, such as TransE [4], ComplEx [5] and RDF2Vec [6]. For RDF2Vec, extensive research has been done in designing heuristics for generating the walks in order to improve embeddings [6]. An analytical study for SPARQL queries has been done before [7], but this has not been used for further implementations.

Up to date, the authors are not aware of research done for augmenting the graph with triples extracted from the result sets of user requested SPARQL queries in order to improve the embeddings.

## 3 DATASET

The dataset used for this project was downloaded from the SPARQL endpoint of the 2016 dataset of DBPedia [1].

Initial experiments demanded an amount of memory which exceeded our available capacity. As in later steps, the embeddings would be evaluated using a ML evaluation framework build for a subset of DBPedia 2016 [8], we made the choice to scale down our dataset to the following subset as also used by the ML evaluation framework: *article_categories_en.ttl; instance_types_en.ttl; instance_types_transitive_en.ttl; mapping-based_objects_en.ttl; skos_categories_en.ttl*

Still, computational demands exceeded our capacity. Therefore, we scaled down even further to a subset with only the exact entities used in the ML evaluation framework, and their two-hop neighbourhood (one hop being one triple starting from the observed entity). This subgraph consists of around 9 million statements.

This subgraph is used for further steps. Although creating this subgraph at this stage will result in less results for the SPARQL queries, we chose this as it does limit the size of the augmented graph, making it manageable for further steps.

*e-mail: wbr950@student.vu.nl

†e-mail: m.cochez@vu.nl

[1] Downloaded from http://downloads.dbpedia.org/2016-10/

## 4 PREPROCESSING SPARQL QUERIES

OpenLink Software hosts the SPARQL platform[2] and has provided us with query logs to analyse. The SPARQL language is a broad language, with various constructs. The basic form of a query is the Basic Graph Pattern (BGP), which are sets of triple patterns. A BGP query with two triple patterns has the following structure:

```
SELECT ?person
WHERE {
?person dct:subject dbc:Presidents_of_the_United_States.
?person dbo:birthPlace dbr:Hawaii
}
```

The result set of this query would consist of all persons which were president of the United States, and have Hawaii as birthplace. This particular query has a instantiation of the result set of one instance: Barack Obama.

However, SPARQL queries can become very complex, consisting of a lot more constructs, like filters for restriction on solutions, optional values, subtracting results sets from one query to another, negations, and unions of queries. This complexity makes extracting the BGPs from the query for our use difficult, as arbitrary choices have to be made in choosing the result set of a query. As per example, for a union of 10 queries, with optional negations, what would we eventually use as result set and why? Therefore, we have simplified the queries to only BGPs (e.g., constructing a set of BGP queries from separate blocks of a complex query).

In this way, we are able to create simple queries which are manageable for further steps. We acknowledge that this preprocessing does tear down the original intent of these queries, but we argue that enough structure is left for use in further steps. This preprocessing resulted in 36 million querries for further use.

## 5 AUGMENTING THE GRAPH WITH IMPORTANCE DATA

From each of the 36 million querries, we retrieved the triples in the original KG from the instantiation of the result sets for the original KG. For each triple we retrieved, we add three new triples in order to increase the connection between the nodes in the specific triple. Methods like RDF2Vec use the neighbourhood nodes by generating walks, which eventually is used to create embeddings for a node. By adding more triples for a specific connection, this might result in more walks generated over this connection, which in turn might add information about importance of the connection.

These three triples are defined as presented in figure 1. Here, the sequence of triples from the instantiation was $< E1 > < relation\_1\_2 > < E2 >, < E2 > < relation\_2\_3 > < E3 >$. Therefore, these triples are both separately augmented. For each triple, a unique ID is created, so a triple of the original KG could be augmented more than once. All triples containing a literal value (e.g., string, data, time) were not added, as embbeding methods do not work properly with literals.

Since we have 36 million queries, augmenting the graph with results for all these queries would still give a considerably big graph. To limit the size of the augmented graph, we decided to omit queries whose results are too general. This was done in two ways. We first decided to omit all queries which had only 1 triple pattern in their structure, as result sets of this query would be very general and thus do not add much specific importance information for certain nodes or edges in the graph. Also, if the instantiation of the result set of a query gave more than a 1000 triples, the query would be too general as well, and therefore these queries were also omitted.

After executing all queries, the resulting augmented graph consisted of 49 million statements.

[2]Available at http://dbpedia.org/sparql

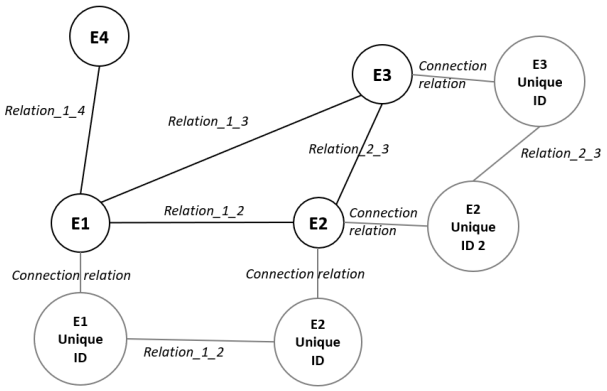


Figure 1: The original graph is presented in black colors. The basic method for augmenting the graph is presented in grey colors. If from the instantiation of a result set of a query, the sequence of triples $< E1 > < relation\_1\_2 > < E2 >, < E2 > < relation\_2\_3 > < E3 >$ have been retrieved, these triples are augmented separately with a unique ID.

Due to the large-scale nature of this step, and the RDF2Vec embedding next, we used the Dutch medium-scale distributed system DAS5 to execute the programs with higher computing resources [9].

## 6 RDF2VEC EMBEDDING

To create embeddings, we chose to use the RDF2Vec embedding method [6]. We used the jRDF2Vec[3] implementation of RDF2Vec [10].

Default parameters in jRDF2Vec were used: at most 100 walks were generated with a depth of 4 hops. The model was trained for 5 epochs, and the dimension of the resulting embedding vectors was set at 200. The random walk method without duplicates was used.

Although 5 epoch is on the low side, training the model took 80 hours already. With more time or compute, a higher amount of epochs could be chosen.

With these parameters, the inner workings of RDF2Vec are as follows: for each entity in the graph, a random walk is generated. For a depth of 4, this means 4 hops, where the current entity of interest would be in the middle:

```
<e1 rel1 e2>, <e2 rel2 entityOfInterest>,
<entityOfInterest rel3 e3>, <e3 rel4 e4>
```

For each entity, at most 100 unique walks were generated. As the depth for jRDF2Vec was 4, the generated sequence could still capture the original entities when walks were generated for the augmented nodes. Then, these walks are treated as a sequence of nodes and edges for which we can use the word2vec neural language model [11].

## 7 EVALUATION

We evaluate the resulting embeddings on the downstream ML tasks classification and regression on five different datasets from different domains using an framework for evaluating and comparing graph embedding techniques for ML tasks as described in [8].

The datasets used are *Cities*, *Metacritic Movies*, *Metacritic Albums*, *AAUP* and *Forbes*. We use Naive Bayes (NB), k-Nearest Neighbors (kNN), C4.5, and Support Vector Machine (SVM) for classification, and Linear Regression (LR), M5Rules (M5), and KNN for regression. We report the average accuracy and root mean

[3]github.com/dwslab/jRDF2Vec

squared error (RMSE) over a stratified 10-fold cross validation for classification and regression, respectively.

With this framework, only the embeddings of a subset of entities are evaluated for these tasks. Some of the entities used for evaluation were not found in our graph. However, this was a negligible amount and thus we think that this will not influence our results much.

The results for classification are presented in table 1, and results for regression in table 2. We can observe that for almost all datasets and algorithms, the original graph embeddings outperform the augmented graph embeddings, which is contrary to our hypothesis.

| Dataset | Algo | Original KG | Augmented KG |
|---|---|---|---|
| Cities | NB | 75.53 | **76.34** |
| | KNN | 65.90 | **73.00** |
| | SVM | **79.85** | 78.88 |
| | C4.5 | 51.19 | **53.78** |
| Metacritic Movies | NB | **68.39** | 54.06 |
| | KNN | **64.25** | 60.77 |
| | SVM | **72.40** | 66.00 |
| | C4.5 | **58.07** | 55.60 |
| Metacritic Albums | NB | **68.18** | 64.81 |
| | KNN | **70.60** | 63.07 |
| | SVM | **75.41** | 71.97 |
| | C4.5 | **61.23** | 56.77 |
| AAUP | NB | **56.84** | 49.70 |
| | KNN | **55.09** | 52.20 |
| | SVM | **68.92** | 64.01 |
| | C4.5 | **50.95** | 50.60 |
| Forbes | NB | **48.19** | 44.93 |
| | KNN | 53.10 | **54.13** |
| | SVM | **60.82** | 60.81 |
| | C4.5 | **50.95** | 49.72 |

Table 1: Results for the average classification accuracy over a 10-fold cross validation for embeddings from RDF2Vec for the original graph and the augmented graph. The best results for each graph, for each dataset, are marked in bold.

| Dataset | Algo | Original KG | Augmented KG |
|---|---|---|---|
| Cities | LR | **30.94** | 51.12 |
| | KNN | 16.31 | **14.28** |
| | M5 | **24.21** | 25.61 |
| Metacritic Movies | LR | **20.67** | 22.30 |
| | KNN | **22.91** | 23.68 |
| | M5 | **29.69** | 30.90 |
| Metacritic Albums | LR | **12.20** | 12.92 |
| | KNN | **12.71** | 14.01 |
| | M5 | **17.45** | 18.53 |
| AAUP | LR | **67.14** | 71.15 |
| | KNN | **86.41** | 87.65 |
| | M5 | **97.80** | 102.42 |
| Forbes | LR | **36.45** | 36.90 |
| | KNN | **37.68** | 37.74 |
| | M5 | **48.64** | 50.43 |

Table 2: Results for the average for regression Root Mean Squared Error (RMSE) over a 10-fold cross validation for embeddings from RDF2Vec for the original graph and the augmented graph. The best results for each graph, for each dataset, are marked in bold.

## 8 Discussion and Conclusion

In this study, original RDF2Vec embeddings for downstream classification and regression ML tasks for a subset of DBPedia 2016 were compared with embeddings of an augmented graph from the DBPedia subset. The augmented graph was augmented with triples retrieved from the instantiation of results sets of SPARQL queries requested by users. The idea was that triples which were selected in queries often, would also contain important nodes and edges in the graph. Adding the triples could add information about this importance.

Contrary to our hypothesis, the embeddings for the augmented graph did not show better results for the ML tasks classification and regression, when compared to the original graph.

For this study, the most basic method for retrieving triples from the SPARQL query results was used. By adding a complete set of 3 triples for each individual triple found in the instantiation of the result set of a query, the context of the sequence triples in the results is lost. When we have a query result containing a sequence of two or more triples, a more advanced method may be to add the full sequence and not separately as done in the basic method. This can be done by re-using the unique object id of the first triple, which results in the augmentation presented in figure 2. With this method, 1 triple less has to added when compared to the basic method, which will save storage. Future research could investigate if using a more advanced method like the one above will lead to improvement of downstream ML tasks when compared to the original graph.

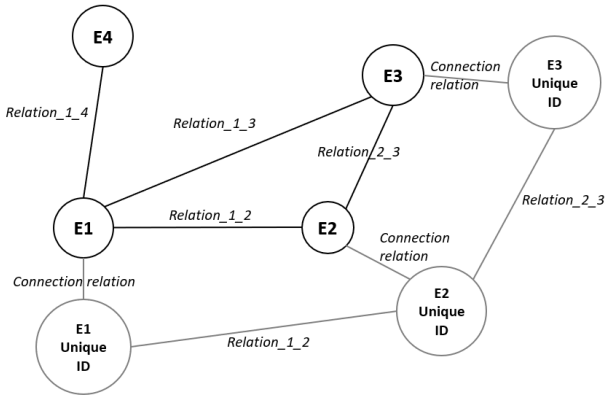


Figure 2: The original graph is presented in black colors. A more advanced method for augmenting the graph by preserving the sequence of triples is presented in grey colors.. If from the instantiation of a result set of a query, the sequence of triple $< E1 > < relation\_1\_2 > < E2 >$, $< E2 > < relation\_2\_3 > < E3 >$ have been retrieved, these triples are augmented as a sequence with a unique ID.

A limitation of this study is the limit of computational resources used. Therefore, we did not use the full DBPedia 2016 dataset but rather a subset containing all the entities and neighbours of entities used in the downstream ML tasks. Initially, the full dataset was used, but as the resulting augmented graph had around 1.5 billion statements we weren't able to run RDF2Vec on our machines. Even using a subset of the dataset, running jRDF2Vec for the augmented graph still took more than 7 days to run.

In future research, an improvement of the RDF2Vec implementation could be investigated, as the jRDF2Vec implementation resulting in issues with memory. With this or with more compute, the experiment could be repeated using the full DBPedia 2016 dataset, which might give other results. Next, future research could test the generalizability of the proposed idea for more embedding methods, such as KGlove [12], as well as trying the proposed more advanced method for augmenting the graph.

## 9 Acknowledgements

We would like to thank OpenLink Software for hosting the http://dbpedia.org/sparql platform and providing us with log files to analyse.

## REFERENCES


[1] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, and Christian Bizer. DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*, 6, 2014.

[2] Petar Ristoski and Heiko Paulheim. Semantic Web in data mining and knowledge discovery: A comprehensive survey. *Web Semantics: Science, Services and Agents on the World Wide Web*, 36, 2016.

[3] Heiko Paulheim. Exploiting Linked Open Data as Background Knowledge in Data Mining. In *Proceedings of the 2013 International Conference on Data Mining on Linked Data - Volume 1082*, DMoLD'13, page 1–10, Aachen, DEU, 2013. CEUR-WS.org.

[4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. In C J C Burges, L Bottou, M Welling, Z Ghahramani, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[5] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex Embeddings for Simple Link Prediction. *CoRR*, abs/1606.0, 2016.

[6] Michael Cochez, Petar Ristoski, Simone Paolo Ponzetto, and Heiko Paulheim. Biased Graph Walks for RDF Graph Embeddings. In *Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics*, WIMS '17, New York, NY, USA, 2017. Association for Computing Machinery.

[7] Angela Bonifati, Wim Martens, and Thomas Timm. An Analytical Study of Large SPARQL Query Logs. *CoRR*, abs/1708.0, 2017.

[8] Maria Angela Pellegrino, Michael Cochez, Martina Garofalo, and Petar Ristoski. A Configurable Evaluation Framework for Node Embedding Techniques. In *ESWC*, 2019.

[9] Henri Bal, Dick Epema, Cees de Laat, Rob van Nieuwpoort, John Romein, Frank Seinstra, Cees Snoek, and Harry Wijshoff. A Medium-Scale Distributed System for Computer Science Research: Infrastructure for the Long Term. *Computer*, 49(5):54–63, 2016.

[10] Jan Portisch, Michael Hladik, and Heiko Paulheim. RDF2Vec Light - A Lightweight Approach for Knowledge Graph Embeddings. *CoRR*, abs/2009.0, 2020.

[11] Tomas Mikolov, Ilya Sutskever, Kai Chen, G.s Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26, 10 2013.

[12] Michael Cochez, Petar Ristoski, Simone Paolo Ponzetto, and Heiko Paulheim. Global rdf vector space embeddings. In Philippe Cudre-Mauroux, Christoph Lange, Claudia d'Amato, Miriam Fernandez, Jeff Heflin, Freddy Lecue, Valentina Tamma, and Juan Sequeda, editors, *The Semantic Web – ISWC 2017 - 16th International Semantic Web Conference, Proceedings*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pages 190–207, Germany, January 2017. Springer Verlag. 16th International Semantic Web Conference, ISWC 2017 ; Conference date: 21-10-2017 Through 25-10-2017.