Fuzzy Search of Knowledge Graph with Link Prediction

Takanori Ugai
ugai@fujitsu.com
Fujitsu Limited
Kawasaki, Japan

ABSTRACT

In SPARQL, only actual connected nodes and properties are retrieved as fact. However, many knowledge graphs expressed in RDF are incomplete, with knowledge graph embedding and graph neural networks used to compensate for the incompleteness.

In this short research paper, we propose a query language, TranSPARQL, a SPARQL extension, enables RDF fuzzy searching that uses these neural network-based link predictions. A variable can describe one predictor in the pattern of subject, property and object in TranSPARQL. Additionally, we describe two implementations; a monotonic implementation with TransE for link predictions and a hybrid implementation using TransE and graph neural networks.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Information systems** → **Query languages**.

KEYWORDS

Fuzzy Search; SPARQL; Link Prediction; Graph Neural Network; Knowledge Graph Embedding

ACM Reference Format:
Takanori Ugai. 2021. Fuzzy Search of Knowledge Graph with Link Prediction. In *The 10th International Joint Conference on Knowledge Graphs (IJCKG'21), December 6–8, 2021, Virtual Event, Thailand.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3502223.3502238

1 INTRODUCTION

In the world of knowledge graphs (RDF), what is written is assumed to be true. However, many knowledge graphs represented in RDF, such as Dbpedia[4], are incomplete. Therefore, when new knowledge is obtained from the knowledge graph, its incompleteness is often compensated for with rules, other means, or by connecting it to other knowledge graphs or databases[7, 20]. Knowledge graph embedding is a technique that gives a vector representation of the relationships between the subjects, properties and objects in a knowledge graph so that they can be approximated by inner product similarity and so on[11, 25]. Knowledge graph embedding is expected to be one of the techniques to compensate for the lack of knowledge graphs by prediction, and many pieces of research have been carried out. Graph Neural Networks[17] can also be used for link prediction. In this paper, we propose a retrieval language, TranSPARQL, an extension of SPARQL, to enable ambiguous retrieval of RDF using link prediction based on these neural networks. The extensions for SPARQL are

- At most, one predictor can be placed for each subject, property and object variable.
- Predictor variables can be assigned a score corresponding to the confidence of the prediction.

It is assumed that the triples that match the conditions indicated by the scores are obtained as triples that match the pattern.

Section 2 describes related works on fuzzy logic based fuzzy search and graph embedding techniques for predicting the pattern. In Section 3, we describe TranSPARQL, an extension of SPARQL that uses link prediction. Section 4 describes two implementations; one is a monotonic implementation using TransE for link prediction and the other a hybrid implementation using both TransE and graph neural networks. Section 5 presents the performance evaluation.

Finally, in Section 6, we give a summary and talk about future work.

2 RELATED WORKS

There have been proposals for extensions that allow fuzzy logic to be applied to search criteria to enable fuzzy searching for objects [1, 2, 13, 14, 21]. In these studies, the objects of three sets of literals in RDF are used as predicates of fuzzy logic, and the search condition can describe the formula of fuzzy logic related to the object. Grammatically, it is an extension of SPARQL and is a user-defined function of the FILTER part without extending SPARQL. Also, only conditions on literal objects can be described using fuzzy logic, not conditions on subjects or properties. The fuzzy search proposed in this paper is based on link prediction utilising RDF structure with knowledge graph embedding techniques and graph neural network techniques. It is not logically rigorous, but it can perform a fuzzy search for any of the three sets of subjects, properties and objects. Nayyeri al el[16] propose the Weighted Triple Loss , a new loss function for KGE models that takes full advantage of the additional numerical weights on facts and it is even tolerant to incorrect weights. they also extend the Rule Loss , a loss function that is able to exploit a set of logical rules, in order to work with weighted triples. There have been quite a lot of work on Description Logics (DLs) based fuzzy ontology languages[22] as theoretical studies. [19] reports on scalable implementation in the fuzzy DL-Lite query engine.

TransE[6], a typical knowledge graph embedding method, learns that if the subject, property, and object representation vectors of a knowledge graph are v_s, v_p, and v_o, respectively, then in a positive

IJCKG'21, December 6–8, 2021, Virtual Event, Thailand

ACM ISBN 978-1-4503-9565-6/21/12…$15.00
https://doi.org/10.1145/3502223.3502238

example triplet, the relationship $v_s + v_p = v_o$ holds. Various other methods exist, such as models that restrict the space according to the relationship (TransH)[24], models based on matrix transformations (TransR) [12], and models that convert to a complex space (ComplEx) [23]. Hamilton al el.[9] proposed a framework to efficiently make predictions about conjunctive logical queries—a flexible but tractable subset of first-order logic—on incomplete knowledge graphs. In our approach, we embed graph nodes in a low-dimensional space and represent logical operators as learned geometric operations (e.g., translation, rotation) in this embedding space. The proposed approach in [3] needs less training on a large and diverse set of complex queries.

Andriy al el.[18] proposed an approach to enable combined usage of the original RDF graph data, implicit relations encoded by word and graph embedding models, and additional knowledge extracted with the help of embeddings by transparent querying using federated SPARQL queries. They use federated hybrid SPARQL queries to combine explicit information stated in the graph, implicit information from the associated embedding models, and information extracted using vector embeddings transparently for the end-user.

3 TRANSPARQL

This section describes TranSPARQL, an ambiguous search language that extends SPARQL with link prediction based on neural networks. The grammatical extensions of TranSPARQL to SPARQL are the following two points. Compared with the proposal in [18] using federated SPARQL queries, our syntax is more natural in standard SPARQL.

- At most, one predictor can be placed for each triple of subject, property and object variable.
- Predictors can have a score corresponding to the confidence of the prediction.

It is assumed that a triple matching the condition indicated by the score can be obtained as a triple matching the pattern.

The grammar changes two things about the existing SPARQL.

- Predictors are now double question marks in a row.
- A score should follow predictions.

For example, if the object is a predictor, it would look like this.

$$subject\ property\ ??object(0.1).$$

The score should be in the range of 0 to 1.0, with a vector tolerance with the sigmoid function.

Extending the semantics[10] of SPARQL, the semantics of the triple pattern including a predictor in TranSPARQL is defined as follows.

$$sub\ prop\ ??obj(score)$$

and

BGP(sub prop ??obj(score)) = U(BGP(sub prop ?obj))

where

$distance((sub)_v + (prop)_v, (obj)_v) < score, obj \in E$

$(E)_v$ is a vector representation of the entity E.

The function distance(x, y, z) is the value of the loss function for x, y, z. BGP is Basic Graph Pattern defined in the semantics of SPARQL.

Calculating the loss function for the given subject and property, all resulting entity triples and the triples containing entities whose value is less than the score are the triples that match the pattern.

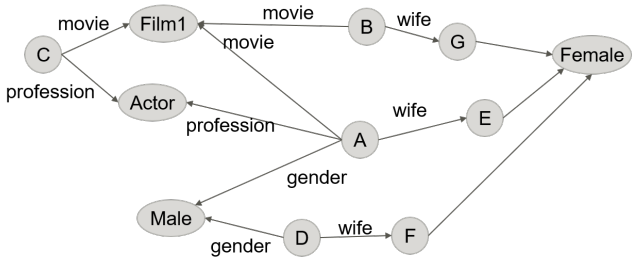

Figure 1: Example of data

For example, we use the knowledge graph of Figure 1. A is a Male Actor who appears in Film1 and has a wife called E. B is an Actor in Film1 and has a wife called G. For this data, a SPARQL query asking who the Male Actor is can be written as follows.

```
Select ?person where {
   ?person a ex:Person .
   ?person exp:Profession  exp:Actor .
   ?person exp:Gender exp:Male .
}
```

For this query, we get the result.

$$?person = A$$

A similar query, written in TranSPARQL, would look like this.

```
Select ?person where {
   ?person a ex:Person .
   ??person(0.1) exp:Profession exp:Actor .
   ??person(0.1) exp:Gender exp:Male .
}
```

The query predicts occupation and gender using link prediction.

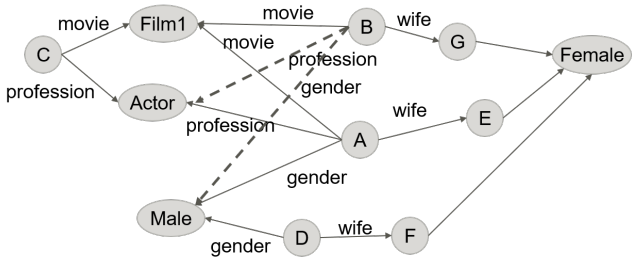

Figure 2: Example of link prediction

From Fig. 1, it is intuitively predicted that the person in Film1 is likely to be an Actor, and the person with a wife is likely to be a Male. From this, we can see that B is likely to be the case, as shown in Fig. 2; and that C and D are also likely to be the case, but less likely than B. In this query, we have specified a small score of 0.1, so we expect to get the most likely answer, B.

In our implementation, including A, we obtained the following results.

Table 1: Example of prediction results

?Person	score
A	0.0001
B	0.01

Since A is actually included in the data, the value of the loss function is smaller.

4 IMPLEMENTATION OF TRANSPARQL

4.1 Server implementation

The server was implemented by modifying Jena[15]. We added a way to write patterns to be predicted to the grammar interpretation part. We made it possible to select the model to be predicted when interpreting the triplet patterns so that multiple prediction models can be used selectively. The RDF model is stored in memory as a server, and then the model is created by merging the prediction results. SPARQL is created by modifying the pattern to be predicted against the model for a regular pattern, then the SPARQL is executed, and finally, the results are returned. If multiple patterns need to be predicted, as in the example shown in the previous section, the predictions are made in parallel using threads. The results are extended to return the final SPARQL execution and the node and score pairs obtained by link prediction, as shown below.

```
{
  SPARQL : JSON Format of SPARQL Result
  Prediction : {
     variable_name : {
        node : score,
     },
  }
}
```

Fig. 3 shows an example screen of the prototype server. It consists of three parts: the top part is where the user enters the TranSPARQL query. The middle part is where the final results are displayed, and the bottom part is where the predictions are displayed with the scores.

4.2 Prediction with TransE

In this implementation, TransE[6] is used for link prediction and the link prediction is based on the model $S_v + P_v = O_v$ used by TransE. The distance between the two is used as the score. For example, *sub prop* ??$obj(score)$ calculates $abs(S_v + P_v - O_v)$ for all nodes in order O, and if it is less than the score, it outputs the node and its score.

4.3 Prediction with combined methods

In this implementation, only for type prediction, the vectors obtained by TransE for each node are used as features and trained by a neural network consisting of all coupled layers for the nodes

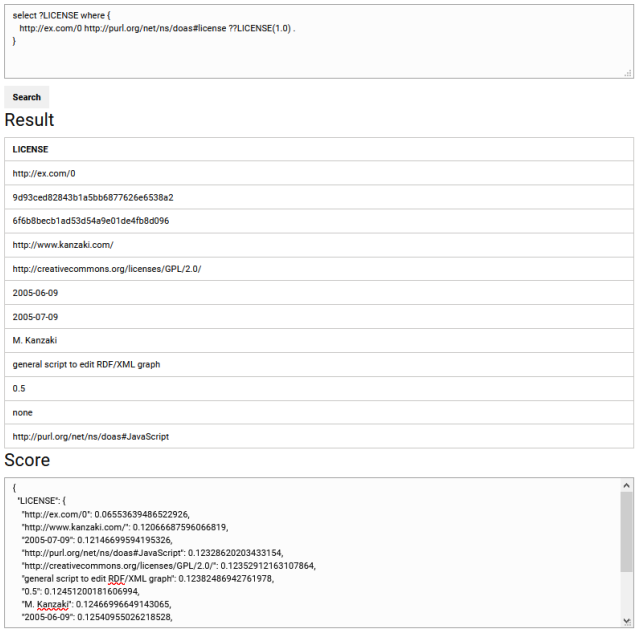

Figure 3: Screenshot of the user interface of the server

with rdfs:type. For methods other than type prediction, TransE is applied, and its distance is scored based on $S_v + P_v = O_v$.

The server uses *sub prop* ??$obj(score)$ for pattern retrieval, and uses link prediction by neural network for other patterns, simply applying TransE.

5 EXPERIMENTS

In this section, we present the results of the performance evaluation of the implementation with the prediction by TransE, TransR and ComplEx and the hybrid implementation of GCN and TransE. The computer used for the evaluation was a virtual PC with a Xeon E5 CPU running at 3.1GHz with 16 cores and 64GB of memory.

5.1 Implementation with TransE, TransR, ComplEx

For the evaluation, The AIFB dataset describes the AIFB research institute in terms of its staff, research groups, and publications. In [5], the dataset was first used to predict the affiliation (i.e., research group) for people in the dataset. The dataset contains 178 members of five research groups. However, the smallest group contains only four people, which is removed from the dataset, leaving four classes. The query used for the evaluation is as follows.

```
Select ?TYPE where {
   ex:0 a ??TYPE(0.5) .
}
```

That is, it finds the distance between $V(ex:0) + V(a)$ and the other nodes, and outputs the one that is smaller than the input(0.5). The processing time of the API is shown in Table 2.

The time taken without the initial cache is the time to find and reorder the distance between $V(ex:0) + V(a)$ and the other nodes.

Table 2: Processing Time of Monotonic Implementation

with TransE	First(without cache)	251ms
	Second or later	22ms
with TransR	First(without cache)	260ms
	Second or later	22ms
with ComplEx	First(without cache)	309ms
	Second or later	22ms
without Prediction		15ms

Table 3: Processing Time of Hybrid Implementation

with Prediction	First (without Cache)	543ms
	Second or later	20ms
without Prediction		15ms

Therefore this time increases with the number of nodes, i.e. with the size of the knowledge graph. Since the predictions are made in parallel, the number of predictions in a query is not very sensitive to the number of nodes, as long as they can be made in parallel. The time taken by the second and subsequent rounds with predictions, compared to no predictions, is the time taken to select those that meet the score criteria from the predictions replicate the original RDF model and merge them with the predictions.

5.2 Hybrid Implementation

In this section, we evaluated the performance of the prediction part using a neural network whose features are vectors obtained by the TransE embedding method. Since loss function f is normalized from 0 to 1 by (sigmoid(f)+1)/2, the same score values as those used for the link prediction by TransE can be used.

The data used for the evaluation was the same as used in the previous section. The processing time of the API is shown in Table 3.

6 SUMMARY AND FUTURE WORK

In the world of knowledge graphs (RDF), it is assumed that what is written is true. However, RDF knowledge graphs are often incomplete, and knowledge graph embedding and graph neural networks are used for link prediction to compensate for this. In this paper, we propose a retrieval language, TranSPARQL, which is an extension of SPARQL, to enable fuzzy retrieval of RDF using link prediction based on neural networks. Two different implementations, one using TransE for link prediction and the other a hybrid implementation, using both TransE and graph neural networks have been implemented and evaluated. The current implementation requires a long time for the first search because it calculates the value of the loss function for every node. The challenge is to optimize the given search formula and reduce the number of possible nodes in advance. We have implemented the ability to use multiple prediction methods, but it is not possible to specify the prediction method from within TranSPARQL. It would be desirable to allow the user to specify the prediction method from within TranSPARQL, and to allow the system to choose the best prediction method if the user has not specified one. We believe that the implementation method of AUTOML[8] can be used as a reference for this purpose. In the current implementation, the RDF model is kept in memory, and the RDF model extended by prediction is a copy made of the original model in memory for retrieval. This requires twice as much memory as the original model, making it difficult to process large data sets. We think it is necessary to implement a system where only the prediction part is stored in memory. In addition, since prediction in parallel using threads also requires memory, it is necessary to implement distributed execution in the future. Hamilton's embedding[9] requires only small dimensions. Using this embedding method helps to achieve high performance.

REFERENCES

[1] Jesus Almendros-Jimenez, Antonio Becerra-Teron, and G. Moreno. 2018. Fuzzy queries of social networks with FSA-SPARQL. *Expert Systems with Applications* 113 (06 2018). https://doi.org/10.1016/j.eswa.2018.06.051

[2] J. M. Almendros-Jimenez, A. Becerra-Teron, G. Moreno, and J. A. Riaza. 2019. Tuning Fuzzy SPARQL Queries in a Fuzzy Logic Programming Environment. In *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. 1–7. https://doi.org/10.1109/FUZZ-IEEE.2019.8858958

[3] Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. 2021. Complex Query Answering with Neural Link Predictors. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Mos9F9kDwkz

[4] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A Nucleus for a Web of Open Data. In *The Semantic Web*. Springer Berlin Heidelberg, Berlin, Heidelberg, 722–735.

[5] Stephan Bloehdorn and York Sure. 2007. Kernel Methods for Mining Instance Data in Ontologies. In *The Semantic Web*, Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 58–71.

[6] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 2787–2795. http://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data.pdf

[7] AnHai Doan, Alon Halevy, and Zachary Ives. 2012. *Principles of Data Integration* (1st ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[8] P. Gijsbers, E. LeDell, S. Poirier, J. Thomas, B. Bischl, and J. Vanschoren. 2019. An Open Source AutoML Benchmark. *arXiv preprint arXiv:1907.00909 [cs.LG]* (2019). https://arxiv.org/abs/1907.00909 Accepted at AutoML Workshop at ICML 2019.

[9] William L. Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. 2018. Embedding Logical Queries on Knowledge Graphs. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montréal, Canada) *(NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 2030–2041.

[10] Steve Harris and Andy Seaborne. [n.d.]. SPARQL 1.1 Query Language. https://www.w3.org/TR/sparql11-query/.

[11] Thomas N. Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *CoRR* abs/1609.02907 (2016). arXiv:1609.02907 http://arxiv.org/abs/1609.02907

[12] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (Austin, Texas) *(AAAI'15)*. AAAI Press, 2181–2187.

[13] Y. Lv, A. Li, and X. Wei. 2014. Fuzzy SPARQL query based on fuzzy RDF model. *ICIC Express Letters* 8 (12 2014), 3145–3150.

[14] Ruizhe Ma, Xiangyue Jia, Jingwei Cheng, and Rafal Angryk. 2015. SPARQL queries on RDF with fuzzy constraints and preferences. *Journal of Intelligent & Fuzzy Systems* 30 (09 2015), 183–195. https://doi.org/10.3233/IFS-151745

[15] Brian McBride. 2002. Jena: A Semantic Web Toolkit. *IEEE Internet Comput.* 6, 6 (2002), 55–59. https://doi.org/10.1109/MIC.2002.1067737

[16] Mojtaba Nayyeri, Gökce Müge Cil, Sahar Vahdati, Francesco Osborne, Andrey Kravchenko, Simone Angioni, Angelo Salatino, Diego Reforgiato Recupero, Enrico Motta, and Jens Lehmann. 2021. Link Prediction of Weighted Triples for Knowledge Graph Completion Within the Scholarly Domain. *IEEE Access* 9 (2021), 116002–116014. https://doi.org/10.1109/ACCESS.2021.3105183

[17] Dat Quoc Nguyen. 2017. An overview of embedding models of entities and relationships for knowledge base completion. *CoRR* abs/1703.08098 (2017).

arXiv:1703.08098 http://arxiv.org/abs/1703.08098

[18] Andriy Nikolov, Peter Haase, Daniel M. Herzig, Johannes Trame, and Artem Kozlov. 2018. Combining RDF Graph Data and Embedding Models for an Augmented Knowledge Graph. In *Companion Proceedings of the The Web Conference 2018* (Lyon, France) *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 977–980. https://doi.org/10.1145/3184558.3191527

[19] Jeff Z. Pan, Giorgos Stamou, Giorgos Stoilos, Stuart Taylor, and Edward Thomas. 2008. Scalable Querying Services over Fuzzy Ontologies. In *Proceedings of the 17th International Conference on World Wide Web* (Beijing, China) *(WWW '08)*. Association for Computing Machinery, New York, NY, USA, 575–584. https://doi.org/10.1145/1367497.1367575

[20] Heiko Paulheim. 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web* 8, 3 (2017), 489–508.

[21] Olivier Pivert, Olfa Slama, Grégory Smits, and Virginie Thion. 2016. A Fuzzy Extension of SPARQL for Querying Gradual RDF Data. In *IEEE International Conference on Research Challenges in Information Science (RCIS'16) Posters*. Grenoble, France. https://hal.inria.fr/hal-01297190

[22] Umberto Straccia. 2001. Reasoning within Fuzzy Description Logics. *J. Artif. Int. Res.* 14, 1 (apr 2001), 137–166.

[23] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. arXiv:1606.06357 [cs.AI]

[24] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes.. In *AAAI*. 1112–1119.

[25] Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2018. Deep Learning on Graphs: A Survey. *CoRR* abs/1812.04202 (2018). arXiv:1812.04202 http://arxiv.org/abs/1812.04202