Reaching out for the Answer: Relation Prediction

Khaoula Benmaarouf and Nadine Steinmetz

Technische Universität Ilmenau, Germany
firstname.lastname@tu-ilmenau.de

Abstract. This paper presents our contribution to the SMART challenge 2021 (SeMantic AnsweR Type Prediction Task), specifically the relation prediction task for both the DBpedia and Wikidata datasets. We introduce our approach to predict the ontology properties (relations) mentioned in a natural language question in order to be able to answer the question correctly. Our solution is based on a pre-trained BERT model using fastai and in combination with data augmentation (for DBpedia). The techniques separately are proven to be very effective for text classification problems and outperform other approaches. In this paper, we used a multi-label classification method built-in fastai library for the SMART task, which gives very good results. Achieving high performances for relation prediction is assured by using DBpedia (∼760 classes) and Wikidata class hierarchy (∼ 50K classes) for results of an experimental evaluation.

Keywords: Question Answering · Text Augmentation· Relation Prediction.

1 Introduction

This paper describes our approach on relation prediction for natural language questions within the context of the SMART (Semantic Answer Type and Relation Prediction Task) challenge 2021 [7]. This task is focused on one of the most popular tasks in Natural Language Processing (NLP) – Knowledge Base Question Answering (KBQA). The aim of KBQA is to transform a natural language question to a formal query – specifically SPARQL – to be able to answer the question. In order to achieve this, two main subtasks can be utilized within the process pipeline: answer type prediction, and relation prediction.

Question Answering (QA) systems are commonly used as interface between a large amount of (un)structured data and users who are enabled to request the data without knowledge of a formal query language. There are two types of QA: open and closed domain. Open domain QA systems do not solve specific topics, but are used to get the proper answers from various topics in shorter form. The downside of open domain QA is that it is difficult for the system to get answers for all possible questions facing various challenges, as e.g. ambiguity, or incomplete knowledge bases. On the other hand, closed domain QA systems are

focused on particular domains, where the QA application has been developed for a specific task, which helps the system to get the answers very fast, and (mostly) correct. For instance, QA systems in the medical field (Alzheimer's diseases) [2] or chatbots applied for specific customer service tasks. In both cases, the QA application can benefit from various subtasks within the QA pipeline. Relation prediction detects references within the natural language questions to assign the correct ontology properties which are necessary for the formal query (specifically SPARQL). This paper proposes a solution for solving the relation prediction task using Bidirectional Encoder Representations from Transformers (BERT), where the prediction task is considered a multi-label classification problem.

This paper is structured as following: Section 2 discusses some previous work that is related to our subtask. Section 3 gives an overview of the datasets and Section 4 depicts some results of the analysis of the both datasets (DBpedia and Wikidata). The preprocessing steps and the training steps are discussed in Sections 5 and 6. Evaluation results are described in Section 7. Finally, a conclusion for our results as well as an outlook is described in Section 8.

2 Related Work

The approach on relation and entity linking by Sakor et. al is based on a set of rules and a mapping of connected entities and relations [11]. Therefore, the approach is independent from the underlying knowledge and the it can be transferred to various knowledge graphs. The initial solution has been tested on Wikidata and achieved good results, but the publicly available API also provides links to the DBpedia knowledge graph.

Abolghasemi et. al proposed an instance-based method to detect the relation of a new question using similar paraphrases of questions in the training data [1]. This method uses two subnetworks : question-question network which uses semantic matching between input question and training questions to know the shortest distance between the input question and its corresponding question in the dataset. The second subnetwork is created from the question-answer relation, where the output question from question-question network is used to get the corresponding answer from the dataset. The approach benefits from the assumption that there are various lexical representations for each question about a relation. Based on these similarities, the authors claimed that the likeness of questions can be utilized to find out the relation hidden behind question phrases. The dataset SimpleQuestions were used for the training. The approach achieved an accuracy of 93.41% which is increased compared to the other state-of-the-art models.

Zhao et. al proposed a solution to solve the problem of incompleteness of KGQA, as the researches are focused on processing each problem independently, without taking the hidden relations inherit from the neighborhood in their consideration [14]. The authors used attention-based graph embedding to capture both entity and relation features between entities in the near neighborhood. The implemented KGQA has an increased F1 score for the relation prediction task

over the model that has no relation configuration for the datasets SimpleQuestions, WebQuestions, GQ and QALD-5.

The problem of QA has been investigated by Mohammed et. al, where the authors aimed to focus on accuracy-complexity tradeoff, as simple straightforward baselines CNN and GRUs were used plus a few heuristics on the SimpleQuestions dataset [9]. The results show that the basic deep learning approach achieves similar results as the state-of-the-art result. The authors performed several experiments utilizing birectional Gated Recurrent Units (BiGRU) and Convolutional Neural Networks (CNN), amongst others. The best approach has reached accuracies of 82.3%, 82.8% respectively in relation predictions.

Since its publication, BERT [3] has been widely used for tasks that require the transformation of language patterns. Transformation of language applies to text summarization, language translation, or question answering. Relation prediction can also be considered a transformation task: from a natural language question to a set of relation labels. Mihindukulasooriya et. al proposed their approach SLING using a BERT embedding based classifier and the AMR graph of the question [8]. After creating AMR triples from the AMR graph representation, the authors combine and rank the results of supervised and unsupervised classification tasks. Naseem et. al presented an approach utilizing a pre-trained BERT model and leveraging the AMR (Abstract Meaning Representation) of a question for the relation linking task [10]. The two-staged approach first identifies the number and position of potential relations in the sentence and the respective AMR graph. In the next step, the most relevant relation is predicted for each previously identified spot. With their approach, the authors outperform several other approaches ([11] and [8] amongst others) on the datasets QALD-9, LC-QuAD 1.0/2.0, and SimpleQuestions.

3 Datasets

Table 1: Datasets of DBpedia and Wikidata.

Datasets	**Train**	**Test**	**Total**
DBpedia	34.204	8.552	42.756
Wikidata	24.112	6.029	30.141

The SMART task provides datasets for the two KBs DBpedia and Wikidata. Some statistical details on the datasets provided for the challenge are shown in Table 1 and Section 4. Our approach considers the task as a relation prediction classification, where each question is assigned a relation category. While this task is considered a short-text classification, what makes the classification challenging is a few unique characteristics of the datasets which contribute to data sparsity. For the challange, for both ontology tasks the following datasets are provided:

relation vocabularies, train data, and test questions. To train a model on the data, it needs to be transformed into a feature-target form.

3.1 DBpedia

The DBpedia dataset consists of 42,756 samples, which is split into 80% as training data, and 20% as testing as shown in Table 1. The dataset is divided into three files (relation vocabulary, test questions, and train data). The relation vocabulary contains properties from the mapped ontology[1] and unmapped properties[2]. Moreover, the total number of relations in the vocabulary is 717. The train file contains the following four attributes: question, relations, number of relations and ID. The number of relations specifies how many different classes of relations are contained in the questions - which results in one list of relevant relations per class. The test questions file only contains the questions and the ID. A sample of the DBpedia train data is shown in Fig **??**.

	question	relations	num_of_rels	id
0	Who is credited as the cinematographer for ete...	[[dbo:cinematography]]	1	smart-2021-rl-dbpedia-0
1	Who was the developer behind demons winter?	[[dbo:developer]]	1	smart-2021-rl-dbpedia-1
2	what east asian country is yokohama landmark t...	[[dbo:location]]	1	smart-2021-rl-dbpedia-2
3	what locations are within united states	[[dbo:locatedInArea, dbo:city, dbo:country, db...	1	smart-2021-rl-dbpedia-3
4	what record is produced by ronnie mcdowell	[[dbo:recordLabel]]	1	smart-2021-rl-dbpedia-4

Fig. 1: Five train samples of the DBpedia dataset.

3.2 Wikidata

Wikidata consist of 24,112 as training data, and 6,029 as testing data as shown in Table 1. The train data contains five attributes: questions, relations, relation_labels, num_of_rels, id. For Wikidata, ontology properties have a unique identifier[3] which is not human-readable (attribute `relations`) and additionally human-readable labels (attribute `relation_labels`). The length of the vocabulary is 3,639. Figure 2 shows five sample records from the training dataset.

4 Data Analysis

4.1 DBpedia

We analyzed the training datasets for the frequencies of the occurring properties to be able to assess the distribution of properties and the sparsity of data for

[1] having `http://dbpedia.org/ontology/` as prefix
[2] having `http://dbpedia.org/property/` as prefix
[3] usually starting with a P, such as P2397 for the property with the human-readable label "YouTube channel ID"

	question	relations	relation_labels	num_of_rels	id
0	What is monomer of propylene ?	[P4599]	[monomer of]	1	smart-2021-rl-wikidata-train-0
1	On what military branch Sigmund Jähn served un...	[P241, P582]	[military branch, end time]	2	smart-2021-rl-wikidata-train-1
2	What movie did Katina Paxinou win an Academy A...	[P166, P805]	[award received, statement is subject of]	2	smart-2021-rl-wikidata-train-2
3	When converted to SI unit, is the darcy equal ...	[P2370]	[conversion to SI unit]	1	smart-2021-rl-wikidata-train-3
4	At what rate was inflation in Venezuela in the...	[P1279, P585]	[inflation rate, point in time]	2	smart-2021-rl-wikidata-train-4

Fig. 2: Five training samples of the Wikidata dataset.

specific classes (class as in classification of relation sets). As shown in Figure 3 the distributions of properties is long-tailed. The most frequent properties are `dbo:genre` and `dbp:birthPlace` respectively. Out of 338 unique mapped properties, 42 only occur once. For the unmapped properties these numbers are 357 and 58 respectively.

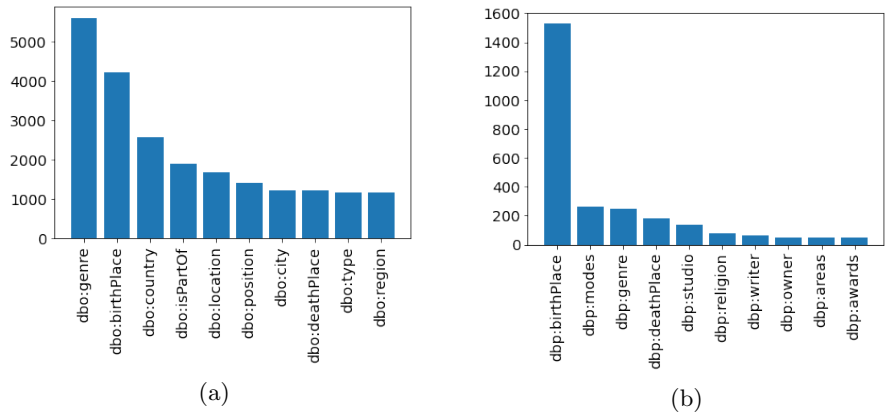

Fig. 3: Top 10 (a) mapped and (b) unmapped ontology properties within the DBpedia train dataset

4.2 Wikidata

Figure 4 shows the most frequently used labels in the training dataset, where *instance of* is the most frequent relation with 6,418 occurrences which is 4 times higher than the second most frequent relation (*point of time* with 1,314 occurrences). Out of 3,171 unique relations in the dataset, a large amount of 1,889 only occurs once.

4.3 Handling of Imbalanced Data

As shown in the previous sections, the training datasets are very imbalanced regarding the distribution and frequency of relations throughout the dataset.

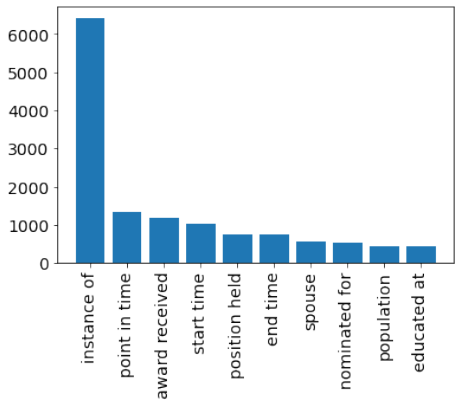

Fig. 4: Top 10 relations within the Wikidata train dataset

This results in low accuracies for questions referring to relations in the long tail of the distribution. Although not considered for the approach presented in this paper, we examined strategies to compensate such imbalances. Unfortunately, simple data augmentation strategies, as discussed in Section 5.2 do not suffice. The training dataset requires to be enriched with questions that contain the long tail relations, but with different contexts and wording than contained in the dataset. We consider this future work to further improve the results of our approach.

5 Preprocessing

Our analytical approach includes several processing steps. For the different classification processes for DBpedia and Wikidata, we utilize the same preprocessing pipeline except in the first step. For DBpedia, we remove the prefix from the relation labels, then lower casing the letters, and resolve the camel case format of the labels. For example, `dbo:RecordLabel` is transformed to *record label*. After that, the pipelines for Wikidata and DBpedia are the same. In the next step, the pipeline takes the questions, relations, and blocks throw data parsing, data augmentation, remove duplicates inside the block, create an indexation between the index of the label inside vocabulary and their block index. Labels that exist in the vocabulary and do not exist in the training set are added. A binary matrix is created and then the labels are merged inside the sentence randomly. The training dataset is split into train and validation sets and finally, the text is tokenized using the BERT tokenizer.

5.1 Parsing Data

Questions, relations, and block parsing is done on training data in the way that the questions syntactic order is shuffled in three different orders and then

combined all the three different types of questions. Indexation is used between vocabulary and their related blocks to get the index values to speed up the process. We utlize lists consisting of: question, length, block, relations.Training data Q is appended into these lists separately and then the question list is split into Q1 and Q2. The subsequent data is appended randomly into splited question lists. We remove duplicates that cause redundancy, as this will lead to easier computations for the model to find patterns from unique blocks, without being biased to one block instead of the other because it is redundant. The matrix is transformed to binary format to be fitted for the training.

5.2 Data Augmentation:

To extend the training data, several augmentation strategies can be applied [4]. The method used for augmentation in our approach was the *Copy-Paste* method. The Copy-Paste technique is a method which duplicates existing data to increase the sample size and add slightly modified or synthetic data. Increasing the number of data helps the model to "see" the specific pattern more often, which is useful when the data is relatively small to be feed to the neural network model. In our case, the data has been copied three times. Thus, the sample size of DBpedia increased to 102,612 records and to 72,336 records for Wikidata.

5.3 BERT Tokenization

For the tokenization of the input data, we utilized the FastAIBertTokenizer from the fastai library [6]. The BERT tokenizer takes the text input and maps it to its integer representation in the BERT word embeddings dictionary and adds some special tokens as `[CLS]` at the beginning of the input text, and `[SEP]` at the end of each input text, `[PAD]` for padding to have all the input texts at the same assigned maximum length, `[UNK]` is given for the tokens that do not exist inside the vocabulary of BERT dictionary. The input text that exceeds the given maximum length is truncated automatically to make the input matrix all the input matrices with the same size. More details on tokenization with BERT are described in [3].

6 Training of the Model

6.1 Language Model and Prerequisites

Most of the modern NLP systems utilized gated recurrent neural networks (RNNs), such as long short-term memory (LSTMs) and Gated Recurrent Units (GRUs), with additional attention mechanisms before the release of transformers [13]. Recurrent neural networks were the state-of-the-art in sequence models especially in NLP problems such as machine translation, text summarization as they can memorize sequence dependencies using the help of some gates [12]. Since RNNs are taking the input tokens one by one according to their position

in the sequence, this increases sequential computation and training time, especially at longer sequence lengths [5]. In addition, RNNs suffer from challenges in handling long-term dependencies as by increasing the number of sequence data, this will be harder for the model to memorize all the past dependencies [13]. Vanishing and exploding gradients are also the reasons that prevent the RNNs to capture the long-term dependencies [12].

In 2017, an encoder-decoder model called transformer was introduced to solve the problems that facing RNNs. Transformers can be used in classification problems that are considered supervised learning, as in our case. For our approach the BERT model is used which is a model from the transformers family. We utilized `BERT_BASE_UNCASED` which is not case-sensitive and has a much lower number of layers compared to the `BERT_LARGE` model, as base models have only 12 layers in Encode, and Decoder, with a total number of parameters 110M [3]. The input text is tokenized using the BERT tokenizer, then the transformed input is given to the BERT model to classify the input text to one of the given classes.

6.2 Hyperparameters

The loss function algorithm is used as binary cross entropy with logistic losses which applies a sigmoid activation layer to the output of the binary cross entropy layer to be mapped to 0 or 1. The binary cross entropy is preferred over multilabel entropy because of higher accuracies.

The evaluation metric used to evaluate the model is F1-score. The simple accuracy metrics cannot be used, as it is not taking into consideration the imbalance of the dataset, while F1-score uses precision, and recall getting a score out of 100% to know how good the model could predict each label. The learning rates are set to 6 different rates.

The maximum sequence length was set to 256, with a batch size of 32, and the model is trained for 20 iterations.

In terms of training and validation, we utilized two different validation methods: static 80/20 split and k fold cross validation with 3 folds. As shown in our results in the next section, we achieved better results with the cross validation method for the Wikidata dataset, but not for DBpedia.

7 Evaluation

The results from different combinations of strategies are shown in Table 2. We utilized the basic approach with trained model and in addition data augmentation for the increase of training data and three different validation split methods: 80/20 split, 99/1 split and cross validation as a flexible version for the validation step. Obviously, the quality of the results did not increase for both datasets using all additional strategies. The cross validation strategy did not achieve better results for DBpedia, but for Wikidata. For DBpedia, the 80/20 split achieved better results than 99/1 split. Whereas the data augmentation step was only

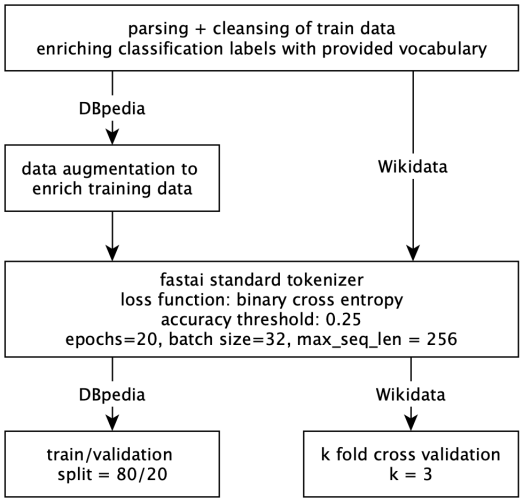

Fig. 5: The pipelines of the best achieving combinations of strategies for both datasets – Wikidata and DBpedia.

Table 2: Results for DBpedia and Wikipedia using different combinations of strategies: cv – cross validation (3 folds), da – data augmentation

Approach	Wikidata		
	Precision	Recall	F1
base + 80/20	0.72152	0.74473	0.70697
base + 99/1	0.79481	0.73229	0.74884
base + cv	**0.75094**	**0.8163**	**0.76018**
base + 80/20 + da	0.79532	0.20055	0.29985
competitor SMART 2021	0.6163	0.61105	0.60701
Approach	DBpedia		
	Precision	Recall	F1
base + 80/20	0.84094	0.84204	0.83586
base + 80/20 + da	0.86135	0.87602	**0.86232**
base + 99/1 + da	**0.86475**	0.87129	0.86194
base + cv + da	0.82497	**0.9204**	0.85404
competitor SMART 2021	0.83682	0.82958	0.83151

successful for DBpedia. For Wikidata, the recall decreases significantly using the augmented training dataset. While we were not able to identify the exact reason for that behavior, we noticed a significantly increased amount of predicted relations when utilizing the augmented training dataset – an average of 9 relations are predicted compared to the results with the best F1 score having only 2 relations predicted at average.

The pipelines of the best achieving combinations are depicted in Figure 5. Overall, our best strategy combinations could outperform the other competitor of the SMART 2021 challenge, as shown in Table 2.

8 Conclusion

In this paper, we presented our approach for the SMART Task challenge of ISWC 2021 for the Relation Prediction Task. The goal was to predict a set of relations relevant to create the formal SPARQL query to be able to answer the question. We created a classification pipeline and additionally implemented data augmentation and cross-validation methods. The results of our experiments show different combinations of the strategies for both datasets – Wikidata and DBpedia. The combination of different strategies achieved very good results compared to the other participant of the relation prediction task. We consider the problem as a set of sequence classification tasks, each one making use of a fine-tuned BERT classifier. For the more fine-grained (and more challenging) problem of Relation Prediction (since the classes can be hundreds or thousands), we have proposed the enrichment of the BERT trained model with additional strategies. For future work, we consider a more adaptive strategy to deal with the imbalanced datasets and utilize data augmentation only for the long tail of the properties in terms of the frequency distribution. Also, for Wikidata, the relation labels should be considered instead of the IDs - although they are the ones required for the classification task.

9 Acknowledgements

This work was partially funded by the German Research Foundation (DFG) under grant no. SA 782/30-1 and STE 3033/1-1.

References

1. Abolghasemi, A., Momtazi, S.: Neural relation prediction for simple question answering over knowledge graph. CoRR **abs/2002.07715** (2020), `https://arxiv.org/abs/2002.07715`
2. Buzaaba, H., Amagasa, T.: Question answering over knowledge base: A scheme for integrating subject and the identified relation to answer simple questions. SN Comput. Sci. **2**(1), 25 (2021). https://doi.org/10.1007/s42979-020-00421-7, `https://doi.org/10.1007/s42979-020-00421-7`

3. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. CoRR **abs/1810.04805** (2018), http://arxiv.org/abs/1810.04805
4. Feng, S.Y., Gangal, V., Wei, J., Chandar, S., Vosoughi, S., Mitamura, T., Hovy, E.: A survey of data augmentation approaches for nlp (2021)
5. Ghojogh, B., Ghodsi, A.: Attention mechanism, transformers, bert, and gpt: Tutorial and survey. http://dx.doi.org/10.31219/osf.io/m6gcn (2020)
6. Howard, J., Gugger, S.: fastai: A layered API for deep learning. CoRR **abs/2002.04688** (2020), https://arxiv.org/abs/2002.04688
7. Mihindukulasooriya, N., Dubey, M., Gliozzo, A., Lehmann, J., Ngonga Ngomo, A.C., Usbeck, R., Rossiello, G., Kumar, U.: Semantic answer type and relation prediction task (smart 2021). arXiv (2022)
8. Mihindukulasooriya, N., Rossiello, G., Kapanipathi, P., Abdelaziz, I., Ravishankar, S., Yu, M., Gliozzo, A., Roukos, S., Gray, A.: Leveraging semantic parsing for relation linking over knowledge bases. The Semantic Web – ISWC 2020 p. 402–419 (2020)
9. Mohammed, S., Shi, P., Lin, J.: Strong baselines for simple question answering over knowledge graphs with and without neural networks (2018)
10. Naseem, T., Ravishankar, S., Mihindukulasooriya, N., Abdelaziz, I., Lee, Y., Kapanipathi, P., Roukos, S., Gliozzo, A., Gray, A.G.: A semantics-aware transformer model of relation linking for knowledge base question answering. In: Zong, C., Xia, F., Li, W., Navigli, R. (eds.) Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 2: Short Papers), Virtual Event, August 1-6, 2021. pp. 256–262. Association for Computational Linguistics (2021). https://doi.org/10.18653/v1/2021.acl-short.34, https://doi.org/10.18653/v1/2021.acl-short.34
11. Sakor, A., Singh, K., Patel, A., Vidal, M.E.: Falcon 2.0: An entity and relation linking tool over wikidata. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management. p. 3141–3148. CIKM '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3340531.3412777, https://doi.org/10.1145/3340531.3412777
12. Siami-Namini, S., Tavakoli, N., Namin, A.S.: A comparative analysis of forecasting financial time series using arima, lstm, and bilstm. CoRR **abs/1911.09512** (2019), http://arxiv.org/abs/1911.09512
13. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA. pp. 5998–6008 (2017), https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
14. Zhao, F., Hou, J., Li, Y., Bai, L.: Relation prediction for answering natural language questions over knowledge graphs. In: 2021 International Joint Conference on Neural Networks (IJCNN). pp. 1–8 (2021). https://doi.org/10.1109/IJCNN52387.2021.9534205