Reasoning Using Link Prediction

Bhavin G Chennur
bhavinchennur@gmail.com

Nishanth S Shastry
nssb107@gmail.com

1. ABSTRACT

The continuous development of autonomous machines increases the demand for reasoning systems in machines. One of the approaches to reasoning is to consider reasoning being equivalent to rule learning, hence a rule based reasoning method has been demonstrated here. Various techniques have been adopted to retain and extract the semantic meaning of the topics. The topics are then ranked using Term Frequency - Inverse document Frequency method. The text corpus is stored in a graph (Knowledge Base) and every topic is stored in one of the nodes of the graph. TF-IDF rule based link prediction is done among the different topics to connect similar topics and cosine similarity for assessing the similarity of the topics, which will then be used for reasoning.

2. KEYWORDS

TF-IDF, Link Prediction, Rule Based Reasoning, Cosine similarity

3. INTRODUCTION

Link prediction is the problem of predicting the existence of a link between two entities in a network. There are different ways to predict the links, for example, it could be framed as an inference problem in a probabilistic graphical model and could be resolved from a variational inference perspective [3]. Link Prediction can be used for recommendation systems, Social networks, information extraction/retrieval and many other fields. Suppose we are given a graph of the Social network at a moment and suppose there are two nodes vi and vj, link prediction is used to predict the probability of the link between the nodes vi and vj.It can be seen through the definition of link prediction that the link prediction task is divided into two categories:

1. The first category is to predict that the new link will appear in future time.

2. The second category is to forecast hidden unknown link in the space.

When we speak of Reasoning Based on Link Prediction, "Reasoning" here is rule based, out of the huge number of data nodes in the graph (also called a Knowledge Base). A link is predicted between the question node and a data node with the help of the TF-IDF concept.Every word will have a ranking and this ranking is limited to the document it is in. Of all the words in the question, the document which has the word with the highest ranking is returned. If not the entire document, the particular sentence / paragraph can be returned as the response.

In Natural Language Processing, during Information Retrieval, Document Similarity plays a very important role. Document Similarity is a concept which involves determination of how similar two or more documents are with respect to each other. It is not only used for searching but also for duplication detection. Key idea is to represent documents as vectors using TF-IDF. Once we have the vector representation, we can similarly find the similarity using any of the similarity metrics such as cosine similarity. Other similarity metrics include *Jaccard Distance and Euclidean Distance*, but we keep cosine similarity as our similarity measure in our project. With the combination of a feature extraction concept(*TF-IDF technique*) and a similarity metric(*cosine similarity*) we are able to distinguish and compare documents to each other based upon their similarities and overlap of subject matter.

(With the help of TF-IDF technique we are able to perform vectorization of documents with TF–IDF numerics. Further with the TF-IDF results for the document we calculate the similarity of the document results.)

4. LITERATURE SURVEY

Let us introduce ourselves to this field of study, knowledge graphs have the information of a diverse set of entities in this world. [1]. Many approaches have been used to avoid weak interaction between the entities of the graph. One such approach is neural tensor networks (NTN), they solve the problem of weak interaction by taking the products of entities and relations [21]. We will have to find ways to infer knowledge from these graphs. For instance, Hierarchial Reinforcement Learning framework learns chains of reasoning from a Knowledge graph automatically [19]. Here, the whole reasoning process is decomposed into a hierarchy of two-level Reinforcement Learning policies for encoding historical information and learning structured action space. Even with the presence of generative pre-trained language models to generate text, they still struggle to use the simple seman-

tic meaning to create new text. Structural and semantic information of the knowledge graphs could be exploited to facilitate commonsense aware text generation [17]. All the methods we have been seeing are the ones that find ways to perform reasoning in knowledge graphs. Generating multiple categories at once is also used instead of predicting links to explore the knowledge base [20].

Let's look at the recent progress in reasoning systems, the following papers were published after 2017. The entire job of using KGs for this purpose could be split into two separate tasks, the first task would be to use the KGs to perform reasoning and the second would be to filter the required information. Both the separate tasks communicate with each other to predict new links [16], this helps entities far apart could also be connected. The knowledge graphs could also be in three dimensional and it can still be worked on effectively [2]. Machines can prove neural theorems to discover meaningful rules and can trace through the neural network multiple times to generate meaningful explanations for their predictions [4]. Multi-hop reasoning is one of the most common techniques to understand new connections between entities [6][7]. Multi-hop reasoning could be learnt by reinforcement learning as well, it takes the best possible state to move forward through the continuous states [8]. In addition to this, two collaborative agents could be trained jointly [13]. Knowledge graphs could also be used for recommendations, for example, network routes could be generated by understanding the logic behind the connections of the entities, this way we reason why a particular customer bought an item [18].

Next, let us contrast the different approaches by looking into their disadvantages and advantages. To understand the challenges let us consider building a question answering system. The users could enter incorrect spellings for words in a query and a question answer system cannot understand them [9], this could be solved to an extent by multi-hop reasoning. KGs could be exploited by using rules to understand and then explain, could be exploited by representing groups of entities in different structures and by using weights and neuron layers [10]. In this paper we present a rule-based approach. Earlier we saw that multi-hop reasoning was a common technique to understand the connections between entities, a competitive approach for that is to use define specific policies and then use them to generate a path. It can integrate recommendations and understandability in a KG [11]. There could be cases where the model fails to predict an existing fact, this problem could be solved by using another supervised model to judge the influence from unobserved facts [12]. Also, Meta-learning can understand why a specific relation appears more number of times and it can use this learning to reduce the number of steps to predict such relations [14].

One big problem with majority of the approaches seen above is that they fail to predict long distance relations [15]. Instead, in our approach we rank all the words in the all the nodes (documents) and then compare them using cosine similarity. Our problem statement addresses this problem by taking all the words in the knowledge graph into consideration. Generally when neural networks are used the weights diminish with the increasing depth of the neural network because of the increasing size of the Knowledge Bases, i.e, vanishing and exploding gradient problems arise in neural networks. Our approach treats all the nodes in the graph equally, it can also be extended to parallel computation on a distributed platform for increased processing power, if needed.

We broadly saw the background literature to introduce ourselves to the field of study and then we saw the latest developments in this field. We also compared and contrasted few of the most common approaches for reasoning using knowledge graphs. Finally, we established our problem statement and showed how it could tackle many of the pitfalls of other studies in this field.

5. MOTIVATION OF THE WORK

KGs could be improved by reasoning as they complete the graphs, one can use embeddings or rules to do so. The method we have adopted in this paper is rule-based and it is generally more explainable and accurate [5]. We wanted to go beyond physical things and understand how thought, a mental phenomenon, arises from brain, a physical matter. This led us to exploring different modelling techniques to model the behaviour of the human brain. If we were to model a question, say, "how much have you improved in the last one year?", what would be the entities and relations to understand this question? We'll answer this in the later sections. We understood that no model is capable of performing all possible reasoning, we wanted to develop a model that was good in a small domain of reasoning. The domain we took up is, performing reasoning in large knowledge bases. Knowledge bases are best represented by Graph data structures and hence we are using graphs in our approach. And finally, our approach is to perform reasoning with the help of link prediction.

6. DATASET PREPARATION

Huge amounts of data are stored in Knowledge bases and these knowledge bases have very diverse set of topics, for example Google Search Engine fetches the results from Google's knowledge base. These knowledge bases are implemented using graphs wherein all the different topics are the nodes of the graph. The graph data structure is maintained using a two dimensional list. Every position in the two dimensional array is occupied by a document input to the knowledge base. Once all the documents are put to the graph the documents are pre-processed for analysis. A copy of the graph is created as the pre-processed data will not be readable and outputs are to be fetched in a readable format. Hence the pre-processing and the analysis are done on a copy of the graph and the outputs are fetched from the original graph. While preparing the graph pointers are maintained on the nodes of the graph to ensure that all the documents are put into different nodes of the graph.

Text pre-processing is done on all the documents in the graph, here we'll iterate through the important ones. These pre-processing are done to improve the efficiency of the lnk prediction between the different nodes of the graph. All the punctuation marks from all the documents are removed with the help of a list of all the punctuation marks. Later all the stop words are removed from all the documents. Stop words and punctuation marks would give a stiff competition to the other words to be ranked and there is a possibility that link prediction would happen on the basis of these stop words and punctuation marks, in which case the semantic meaning would be lost. Consider two sentences, say a question-

answer pair, they would be irrelevant to each other.

A few words can take up many forms, for example, consider the word 'consultant', this word could be in the form of 'consulting' or 'consulted'. In this case our predictor would treat all the three of them as different words and possible links would not be predicted. To improve the efficiency of the model, we perform *STEMMING* on all the words in all the documents (text corpus). Stemming converts all the above mentioned words to 'consult' and hence all the three words have a chance of being treated equally. There are two ways to do this, one is stemming and the other is *Lemmatization*. Stemming is more aggressive than Lemmatization i.e., Stemming performs its task in more cases than Lemmatization. After all this a bag of words is created. Bag of words is a collection of all the words in all the documents.

To tackle with large amounts of data and since we are considering the scenario of *"reasoning"*, we considered a FAQ data-set with predefined FAQ questions with respect to *'Aadhar FAQ', 'Amazon sagemaker FAQ', 'HDFC FAQ'*, and many more(provoded in *https://www.kaggle.com/*).

7. PROPOSED METHODOLOGY

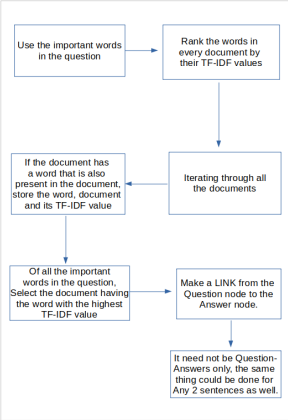

Figure 1: Flow Diagram

TF-IDF stands for "*Term Frequency, Inverse Document Frequency.*" It's a way to score the importance of words (or "terms") in a document based on how frequently they appear across multiple documents. *(standard definition)*

If any word appears in more number of documents the weightage for the word will decrease. The idea behind this is that common english words appear in all the documents and they are not the ones having the content of the topic. For example, 'The' appears in almost all the documents and it does not contain the content of the topic of the document. This is a very common example but every word will have a certain degree of being common, and hence the different weightages ranks the words accordingly.

(Lets look at some of the standard definitions of 'TF' and 'IDF', and their respective formulae)

Term Frequency (tf): Gives us the frequency of the word in each document in the corpus. It is the ratio of number of times the word appears in a document compared to the total number of words in that document. It increases as the number of occurrences of that word within the document increases. Each document has its own tf. (*Term Frequency also weighs words, if a word appears more number of times in a document then the term is given a higher weightage.*)

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}}$$

Figure 2: Term Frequency(tf)

tf i,j = number of occurrences of i in j

Inverse Data Frequency (idf): Used to calculate the weight of rare words across all documents in the corpus. The words that occur rarely in the corpus have a high IDF score. It is given by the equation below.

$$idf(w) = log(\frac{N}{df_t})$$

Figure 3: Inverse Data Frequency(idf)

df t = number of documents containing t

Inverse document frequency for a word is the same for the entire corpus whereas the term frequency for a word is different in every document.

TF-IDF is a ranking method for all the words in any document in the corpus. The below formula is applied for every word in a document to get the TF-IDF weightages for the words and later all these weights are sorted to get the rankings for the words. (*Combining the two we come up with the TF-IDF score (w) for a word in a document in the corpus. It is the product of tf and idf*)

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

Figure 4: TF-IDF formula

tf i,j = number of occurrences of i in j
df i,j = number of documents containing i
N = total number of documents

When a link is to be predicted between two topics one of the topics is given and the other is searched for. Firstly, pre-processing is done for the entire corpus and then we create a bag of words. Then the inverse document frequency is calculated for all the words in the corpus and then we start to consider every document. For every document term frequencies and then the TF-IDF weights are calculated for all the words in the document.

We have already selected or given one topic, for every word in the given topic we'll iterate through all the documents and check if the document has the the word. So, in every document we'll check if the document has any of the words of topic 1, if so we'll add the TF-IDF value and the word to a list. After this process is done, we'll select the word with the highest TF-IDF value from the list. At last we'll make a link from topic 1 to topic 2.

Lets consider an example where a database of documents is to be processed such that each term is assigned a dimen-

sion and associated vector corresponding to the frequency of that term in the document. There is a method which can be used called *cosine similarity.* We use this cosine similarity as a measurement to differentiate, also compare documents with one another with respect to their similarities and see if they are substantially identical. For a large amount of data, we can create a ML model which in turn uses cosine similarity to perform the same.

Lets consider the standard definition of *'Cosine Similarity'*. It measures the cosine of the angle between two non-zero vectors of an inner product space. This similarity measurement is particularly concerned with orientation, rather than magnitude. Lets look at the given definition, mathematically. We begin the measurement of cosine similarity by finding the cosine of the two non-zero vectors. We can further derive by using the Euclidean dot product formula which is generally written as shown in '**Figure 5**'.

$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \, \|\mathbf{B}\| \cos\theta$$

Figure 5: dot product formula

Further, considering the two vectors and the dot product, we can define cosine similarity as shown in '**Figure 6**'.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Figure 6: cosine similarity formula

The output will produce a value ranging from -1 to 1, indicating similarity where -1 is non-similar, 0 is orthogonal (perpendicular), and 1 represents total similarity. *(Here, Ai and Bi are components of vector A and B respectively.)*

As we will be working on this concept, it would be nice to brief its usage in our work. *(Cosine similarity is a measure of similarity between two nonzero vectors of an inner product space that measures the cosine of the angle between them.)* If we consider the visual view with respect to the similarity relation, the cosine of the angle between two vectors is a measure of how similar they are,

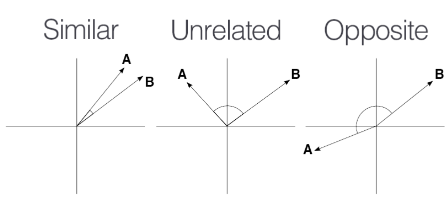

Figure 7: Similarity Relations

We apply TF-IDF as a transformation to texts which gives out two real-valued vectors. Consider an angle θ as the angle between the vectors which can be seen in the above figure. Further we can then obtain the cosine similarity value of any pair of vectors by taking their dot product and dividing that by the product of their norms, hence resulting in the cosine of the angle (θ) between the vectors.

8. RESULTS

Upon successfully applying the concepts of TF-IDF and cosine Similarity, we could see the application of Link prediction at first hand and sought out to procure the results to perfection. *Firstly*, we implemented the concepts described with plain/straightforward python implementation. With numerous testing, we could finally perform link prediction and end up with exact responses for a small dataset. This gave us the complete internal working of link prediction when implemented with the concepts briefed earlier. Once implemented with a large dataset, the model built with features of python such as classes, functions and the libraries provided, showed impressive accuracy with 98.43 % being the lowest and highest being 98.84%.

Firstly, let us look at a toy example to understand how link prediction works:

Here, the graph is implemented as a two dimensional list. To explain the concept we have taken a graph with 100 nodes, this is scalable to larger numbers. Each element of the two dimensional list is a node of the graph and every node holds one text document of any size.

The following is an instance of the graph:

[['the bottle is red in colour', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '], [' ', 'bs ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '], [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '], [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '], [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '], [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '], [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '], [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '], [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '], [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']]

'bs' is another extra word added to show the impact of other documents on the rankings. Now, let us create a copy of the graph and remove the punctuation marks and the stop words from the document and let us also perform stemming. The graph will now look like:

[['bottl red colour ', '', '', '', '', '', '', '', '', ''], ['', 'bs ', '', '', '', '', '', '', '', ''], ['', '', '', '', '', '', '', '', '', ''], ['', '', '', '', '', '', '', '', '', ''], ['', '', '', '', '', '', '', '', '', ''], ['', '', '', '', '', '', '', '', '', ''], ['', '', '', '', '', '', '', '', '', ''], ['', '', '', '', '', '', '', '', '', ''], ['', '', '', '', '', '', '', '', '', ''], ['', '', '', '', '', '', '', '', '', '']]

You can see that 'bottle' was stemmed to 'bottl', this is done so that the different forms of the words are recognizable for link prediction. Consider 'bottling' and 'bottle', the common part to both the variants of the same word is 'bottl'. Hence the above graph has the words in this form.

A bag of words is a unique collection of all the words in the entire corpus, the bag of words for this toy example is:

'bottl': 1, 'red': 1, 'colour': 1, 'bs': 1

The inverse document frequencies of all the words in the graph is:

'bottl': -0.6931471805599453, 'red': -0.6931471805599453, 'colour': -0.6931471805599453, 'bs': -0.6931471805599453

TF-IDF rankings are unique for every documents, rankings for words in all the documents of this graph is: [['bottl': -0.23104906018664842, 'red': -0.23104906018664842, 'colour': -0.23104906018664842, , , , , , , , ,], [, 'bs': -0.6931471805599453, , , , , , , ,], [, , , , , , , , ,], [, , , , , , , , ,], [, , , , , , , , ,], [, , , , , , , , ,], [, , , , , , , , ,], [, , , , , , , , ,], [, , , , , , , , ,], [, , , , , , , , ,]]

If the query is, 'which colour is the bottle ?'

Using the explained methodology the answer would be: 'the bottle is red in colour'

Question will be in one node and a link is predicted from the question node to the document containing the answer.

Here, the graph holds only one document in the first node and this document has a simple text 'the bottle is red in colour'.

The below table shows a few successful runs of the program, *(A total of **2415** Q/A count was taken, **2415** is the no. of rows(Q/A data) in dataset)* The references for the dataset have been mentioned.

Test Runs	Passed Test Cases	Failed Test Cases	Accuracy in %
1	2384	31	98.72
2	2377	38	98.43
3	2383	32	98.67
4	2377	38	98.43
5	2381	34	98.59
6	2387	28	98.84
7	2383	32	98.67

9. CONCLUSION

Our approach of Link Prediction could be applied for various real life applications. For instance, could be used to recommend text articles for a user based on his preferences. Here, the nodes will have the text documents and a link would be predicted between the nodes of the graph. Another application would be to improve the search results, if a user's search query doesn't yield good results are approach of link prediction could be applied to find topics similar to the user's query and enhance the user's search query to obtain better results. Our current work also has high potential to be converted to a simple *FAQ chatbot*, but with conversations limited to the specified dataset provided to the program. With the fundamentals and the different uses of link prediction, there are a few **standard limitations** of link prediction which we considered:

1. Link prediction proves to be slow for large vocabularies when it computes document similarity directly in the word-count space.
2. Link Prediction considers counts of different words as a proof for independent similarity measurements.
3. Semantic similarities between words, are not used in Link Prediction.

10. REFERENCES

1) Kazemi, Seyed Mehran and Poole, David, Advances in Neural Information Processing Systems,S. Bengio and H. Wallach and H. Larochelle and K. Grauman and N. Cesa-Bianchi and R. Garnett,4284–4295,Curran Associates, Inc., SimplE Embedding for Link Prediction in Knowledge Graphs, volume 31,2018.

2) M. Nayyeri, M. M. Alam, J. Lehmann and S. Vahdati, "3D Learning and Reasoning in Link Prediction Over Knowledge Graphs," in IEEE Access, vol. 8, pp. 196459-196471, 2020, doi: 10.1109/ACCESS.2020.3034183.

3) Wenhu Chen and Wenhan Xiong and Xifeng Yan and William Yang Wang, Variational Knowledge Graph Reasoning, CoRR, volume abs/1803.06581,2018.

4) Minervini, P., Riedel, S., Stenetorp, P., Grefenstette, E. Rocktäschel, T.. (2020). Learning Reasoning Strategies in End-to-End Differentiable Proving. Proceedings of the 37th International Conference on Machine Learning, in PMLR 119:6938-6949.

5) Wen Zhang, Bibek Paudel, Liang Wang, Jiaoyan Chen, Hai Zhu, Wei Zhang, Abraham Bernstein, and Huajun Chen. 2019. Iteratively Learning Embeddings and Rules for Knowledge Graph Reasoning. In The World Wide Web Conference (WWW '19). Association for Computing Machinery, New York, NY, USA, 2366–2377.

6) Z. Wang, L. Li, D. D. Zeng and Y. Chen, "Attention-based Multi-hop Reasoning for Knowledge Graph," 2018 IEEE International Conference on Intelligence and Security Informatics (ISI), Miami, FL, 2018, pp. 211-213,

7) Batselem Jagvaral, Wan-Kon Lee, Jae-Seung Roh, Min-Sung Kim, Young-Tack Park, Path-based reasoning approach for knowledge graph completion using CNN-BiLSTM with attention mechanism, Expert Systems with Applications, Volume 142, 2020, 112960, ISSN 0957-4174,

8) Wenhan Xiong and Thien Hoang and William Yang Wang, DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning, journal CoRR, 2017.

9) Yuyu Zhang and Hanjun Dai and Zornitsa Kozareva and Alexander J. Smola and Le Song, Variational Reasoning for Question Answering with Knowledge Graph, journal CoRR, volume abs/1709.04071,2017.

10) Xiaojun Chen, Shengbin Jia, Yang Xiang, A review: Knowledge reasoning over knowledge graph, Expert Systems with Applications, Volume 141, 2020, 112948, ISSN 0957-4174.

11) Yikun Xian, Zuohui Fu, S. Muthukrishnan, Gerard de Melo, and Yongfeng Zhang. 2019. Reinforcement Knowledge Graph Reasoning for Explainable Recommendation. In Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19). Association for Computing Machinery, New York, NY, USA, 285–294.

12) Xi Victoria Lin and Richard Socher and Caiming Xiong, Multi-Hop Knowledge Graph Reasoning with Reward Shaping, journal CoRR, volume abs/1808.10568,2018.

13) Cong Fu and Tong Chen and Meng Qu and Woojeong Jin and Xiang Ren, Collaborative Policy Learning for Open Knowledge Graph Reasoning, journal CoRR, volume abs/1909.00230,2019.

14) Liang, Xiaodan and Hu, Zhiting and Zhang, Hao and Lin, Liang and Xing, Eric P, Advances in Neural Information Processing Systems, pages 1853–1863,Curran Associates, Inc.,Symbolic Graph Reasoning Meets Convolutions, volume = 31,2018.

15) Xin Lv and Yuxian Gu and Xu Han and Lei Hou and Juanzi Li and Zhiyuan Liu, Adapting Meta Knowledge Graph Information for Multi-Hop Reasoning over Few-Shot Relations,CoRR, volume abs/1908.11513, 2019.

16) Yunan Zhang and Xiang Cheng and Heting Gao and Chengxiang Zhai, Cooperative Reasoning on Knowledge Graph and Corpus: A Multi-agentReinforcement Learning Approach, CoRR, volume abs/1912.02206,2019.

17) Haozhe Ji and Pei Ke and Shaohan Huang and Furu Wei and Xiaoyan Zhu and Minlie Huang, Language Generation with Multi-Hop Reasoning on Commonsense Knowledge Graph, 2020.

18) Wang, X., Wang, D., Xu, C., He, X., Cao, Y., Chua, T.-S. (2019). Explainable Reasoning over Knowledge Graphs for Recommendation. Proceedings of the AAAI Conference on Artificial Intelligence, 33(01), 5329-5336.

19) Guojia Wan, Shirui Pan, Chen Gong, Chuan Zhou, Gholamreza Haffari. Reasoning Like Human: Hierarchi-

cal Reinforcement Learning for KnowledgeGraph Reasoning. Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20), 2020.

20) Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu and Jun Zhao. Knowledge Graph Embedding via Dynamic Mapping Matrix. d the 7th International Joint Conference on Natural Language Processing, pages 687–696,Beijing, China, July 26-31, 2015.

21) Socher, Richard and Chen, Danqi and Manning, Christopher D and Ng, Andrew, Advances in Neural Information Processing Systems, pages 926–934, 2013.

Dataset References

1. https://www.kaggle.com/narendrageek/mental-health-faq-for-chatbot

2. https://www.kaggle.com/abbbhishekkk/faq-datasets-for-chatbot-training