Multi-Hop Question Answering Using Knowledge Graph

Frank Tianren Wang

Supervised by Diana Inkpen

6040795

CSI4900

April 20, 2020

Abstract

The main aim of this project is to explore the ability of a graphical knowledge representation in supplementing deep learning models to learn multi-hop reasoning, which is a crucial aspect of reasoning over language. Modern deep learning has largely forsaken this approach to problem solving in favor of purely dense neural networks due to their ease of use and state-of-the-art performance, so exploration of a graphical approach could open new avenues of research that has more future potential. The project investigated building a graphical knowledge representation using a classical transformer encoder-decoder architecture and an information extraction system called OpenIE. This report shows that the transformer approach is inadequate for building a knowledge graph because of contamination of the transformer output by positional encoding. The graph neural network model achieved underperforming accuracy on a multi-hop reasoning dataset, but it performed at baseline level without proper embedding of knowledge graph. Ablation experiments showed that the model may not be processing the graph in a discrete reasoning-like manner, and such reasoning is difficult to simulate in a purely deep learning manner.

Table of Contents

1.0 Introduction 1

1.1 Language Reasoning in Deep Learning 1

1.2 Graphical Approach to Deep Learning 1

1.3 Project Goal 2

2 Methods 2

2.1 Dataset 2

2.2 Graph Embedding 3

2.3 Graph Neural Network 4

3 Result 5

3.1 Transformer as a graph embedder 5

3.3 Performance on questions 7

4.0 Discussion 8

5.0 Conclusion 9

6.0 Bibliography 9

1.0 Introduction

1.1 Language Reasoning in Deep Learning

Forming relationships between ideas and entities and applying them in novel situation is the hallmark of reasoning. For example, reading scientific papers or textbooks reveal new ideas that the readers relate to their existing understandings of a topic, and they can apply their new understanding to form novel conclusions and hypothesis. This is a fairly complex process, as the extraction of key ideas and their relationships from new information has to be accurate, and there is a countless number of ways to manipulate and interpret them.

Since language is the primary medium for transmitting information, there has been hope amongst the NLP community that deep learning language models would exhibit characteristics of reasoning[1]. One such example is GPT-2 developed by OpenAI, where they showed that their language model can perform a variety of NLP tasks without being specifically trained on them just by language modelling millions of domain non-specific texts. Google's pretrained BERT modelled has also being used by a lot of people in downstream tasks of reasoning and inference and they are achieving state-of-the-art performance in those tasks[2].

While the performance achieved by these models on their respective datasets is respectable, they are nothing more than statistical pattern matchers. Theoretically, if the models are large enough and have sufficient training data, then they can mimic the desired result it is trained for without acquiring the understanding of the semantics of entities and relationships in language. For example, OpenAI's GPT-2 model produces sentences like "there is fire in the water". When asked "what is the largest state in United States?", GPT-2 would wrongly answer California. What likely caused these glitches is that the model is simply putting together words that are likely to occur together, not because the semantics of the produced output is true. Fire and water occur together frequently in texts due to the nature of their interaction. Although California is not the biggest state, it is the most "popular" state because of its association with North American pop culture. A model that truly understands the real world relationships between entities in language will likely need explicit encoding of such information, but this is not strongly focused in modern deep learning research.

1.2 Graphical Approach to Deep Learning

Recently there has been a surge in using graphical methods in deep learning due to the increased abundance of network data, such as connections in social media or links between websites. Just as convolutional and recurrent neural networks are adept at processing image and sequential inputs, respectively, graph neural networks (GNN) are designed to process inputs where there are relationships between entities[3]. Like its predecessors, GNN can perform inference on the network, like classification of whether a network of individuals are part of a university club or company. GNN can also solve some classical computer science problems with high accuracy, like finding the shortest path between two points in a graph or sorting an array (provided that the array is expressed in a form that can be understood as a graph).

There are several potential advantages to solving NLP reasoning problems in graph form. Firstly, graphs are well-suited for solving multi-hop problems. For example, to answer the question "does tiger have blood?", if given the statements that "animals have blood" and "tiger is an animal", then the answer would be "Yes". This question can be solved by converting the statements into conceptual entities of blood, animal, tiger and their relationships, tiger would be connected to animal, and animal would be connected to blood. In this structured representation, it is easy to tell that there is an indirect connection between tiger and blood that suggests tiger most likely has blood. The second advantage with graphs is that it explicitly encodes unique relationships between entities. Using an earlier example, letting the relationship of concept of water with concept of fire be represented by an antagonistic vector could reduce the probability that water and fire occur in a neutral or friendly context. Lastly, the number of connections in a graphical knowledge representation can be kept sparse to minimize spurious interactions between concepts. This is in contrast to an NLP architecture like Transformer, where it allows every token in a sequence to interact with every other token, turning the sequence into an entangled representation[4]. This is less likely to happen in a sparse graph network, since information is only passed between relevant concepts.

1.3 Project Goal

While there are formal and informal publications on the topic of solving reasoning problems using GNN, no prior work has attempted to specifically solve multi-hop reasoning problems using aggregated knowledge representations, thus this is the goal of this project. While the result is still a work in progress or possibly might be a dead end, it reveals a number of weaknesses in the approach taken that were never discussed in the research community, and could pave the way for future work in this line of research.

2 Methods

2.1 Dataset

This project attempted to solve the task of OpenBook Question and Answer dataset, which contains around 6000 pre-high school level multiple choice science questions associated with 1500 science facts that are directly necessary for answering the questions and an additional 5000 crowd sourced science facts that supplements the core facts. This dataset was chosen because this was the only dataset to my knowledge that contained both the questions and the facts necessary to answer them, and solving the questions require multi-hop reasoning. For example, one question in the validation set is “There is most likely going to be fog around a) a marsh b) a tundra c) the plains d) a desert. (correct answer is A)”. The facts necessary for answering this question are present, and they are “fog is formed by water vapor condensing in the air” and “Swamps are uncultivated bodies of water also known as marshes.”

As described above, each fact in this dataset is a single sentence, and it typically describes a simple relationship between two concepts. Some facts in the dataset describe physical phenomenon in great detail like “a force acting on an object in the opposite direction that the object is moving can cause that object 's speed to decrease in a forward motion.”. With the methodology of this project, facts like this are extremely difficult to conceptualize so they may not be properly integrated into a knowledge graph.

2.2 Graph Embedding

The first component of this project is the embedding of facts into a knowledge graph, since knowledge must be in a graph form in order to perform graph operations on NLP tasks. There are pre-existing methods where one can obtain knowledge graph from information extraction of knowledge texts[5,6]. One could also get them from databases like DBpedia and Wordnet that has knowledge graph data on common topics, but the issue with them is that there does not exist a reasoning-heavy NLP task that is well-suited for any specific knowledge in these databases. Another more unconventional method is to use sequence-to-sequence techniques to convert pieces of texts into representations that could be treated as graph nodes.

Aside from the method of acquiring the knowledge graph, another important consideration is whether to aggregate all the knowledge known in the corpus into a single graph or to generate knowledge graph on a sample-by-sample basis. Most literature published on this topic use the latter approach, but an aggregated knowledge graph has the significant advantage of allowing the model to perform multi-hop reasoning with information that are not present in the sample context. This project will embed the knowledge graph by aggregation.

The first embedding method studied in this project uses a unidirectional transformer autoencoder[4] to learn the structure of the science facts and extract the facts’ intermediate representations to be used as graph nnodes (figure 1). The original motivation for using transformer was that it had slightly better performance on language translation tasks than an RNN and the design of transformer overcomes RNN’s major weakness in not being able to simultaneously capture the context of the entire input text. The other motivation to use transformer was that it could compress a sequence into a smaller sequence. This theoretically helps to reduce the amount of space each sentence can take so the graph does not waste graph space embedding spurious words like “the” and “of”. For example, a sentence like “An example of reproduction is laying

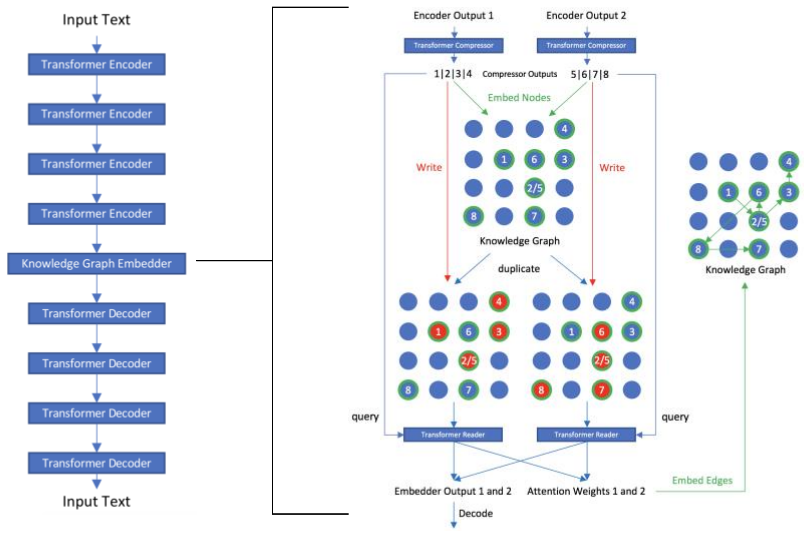


Figure 1. Transformer autoencoder architecture for knowledge graph embedding. Left, the classical transformer encoder-decoder architecture as described in Vasvani and et al 2017 supplemented with an intermediate knowledge graph embedder. Right, the knowledge graph embedder. It takes as input the current batch of outputs from the last encoder layer, and compresses them into constant size sequence (in this diagram 4).

eggs." could hopefully be compressed into a sequence of three tokens that represent "reproduction' 'is' 'laying eggs'. The nodes from a previously randomly initialized graph were then matched with these tokens based on Euclidean similarity, and then the nodes were updated using exponentially smoothed average.

The second method used an information extraction system called OpenIE to convert fact sentences into relationship triples. Each element of the triple is made up of words, so the vector representation of each element is derived by taking their average GloVe word embedding[7]. In the relationship triple, there are two arguments (which are basically the subject and the object in a sentence) and their relationship, and they were treated as the nodes and the edge, respectively, during graph embedding. The concepts obtained from the OpenIE method were aggregated via K-means clustering to 512 clusters. Concepts that were connected in the relationship triple would also have an edge between them in their clustered nodes in the knowledge graph.

2.3 Graph Neural Network

As described earlier, a GNN will process the scientific questions. A typical GNN consists of several functions to perform its graph computations. It first passes the hidden state of every node in the graph through a custom set of layers (MLP in this case), and then pass the output to every node that it is connected to. A node can receive signal from multiple nodes, so these signals are either averaged or summed (averaged in this case). If the edges of a graph also encodes information important for a graph, then it can be part of the computation in sending the signal between nodes (all edges are featureless in this report). In addition, the network keeps track of a global state that sums or averages the hidden states of all nodes and passes the aggregated result through its own custom set of layers. These computation essentially allows the nodes in the graph to interact with one another, and let them reach a state that is optimal for performing inference.

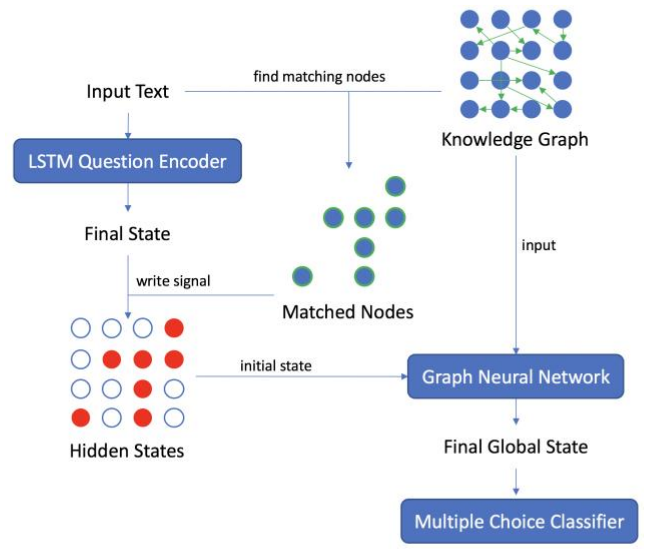


Figure 2. GNN model for answering multi-hop questions. The model first passes the input question concatenated to one of its choices through an LSTM to generate a question context vector (the final state of the LSTM). The model then identifies the nodes in the original graph that it would write the question context vector to based on similarity to the tokens in the input text. It then writes that signal into a zeroed graph hidden state, and used it is used as an input along with the original knowledge graph for the graph neural network. The node features and the connections in the knowledge graph tell the graph neural network how to process the hidden states in each recurrent run. After 5 recurrent runs, the GNN outputs the final global state for that question choice. The multiple choice classifier aggregates the four input choice variations to generate a probability distribution for the correct answer.

For this particular task, each sample is made of four inputs that represent the four choices in the question, each of which was concatenated with the question stem. They are passed through an LSTM to get the context vector for the question, and the context vector was written into the hidden states of nodes chosen by finding the closest matching node of each word in the input by Euclidean similarity. The hidden states of the nodes were all zero to prevent interaction between nodes that were not written. After five rounds of GNN recurrence, the model uses the global state of each graph to output a probability distribution over the different question choices and classify for the correct answer.

3 Result

3.1 Transformer as a graph embedder

In order for transformer to appropriately embed the knowledge graph, the output of the transformer encoder needs to compress the original fact into meaningful chunks. To determine whether the compressed sequence adequately represents the semantically meaningful concepts of the input sequence, the distribution of attention weights in the compressed sequence of some samples were visualized in figure 2. The issue with

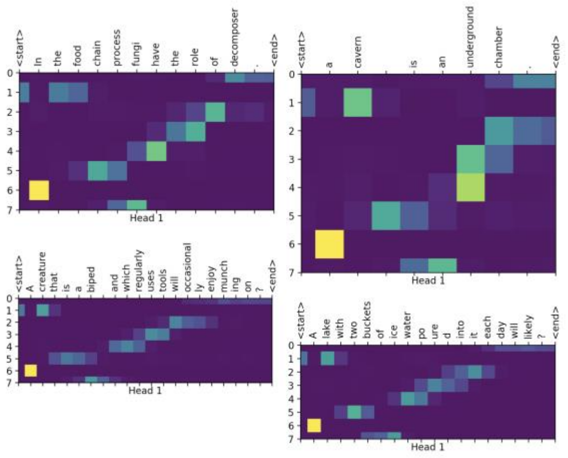
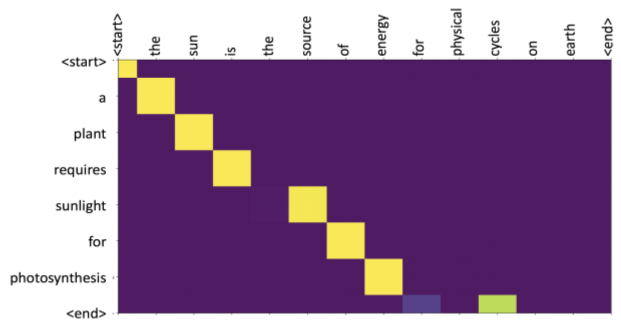


Figure 3. Attention weight distribution of sequence compression by transformer. The Y-axis represent the position index of the compressed sequence. The X-axis represent the tokens inputted into the transformer. The four example are four different sentences inputted into the transformer. Yellow represents maximum attention (100%), dark being minimum attention (0%).



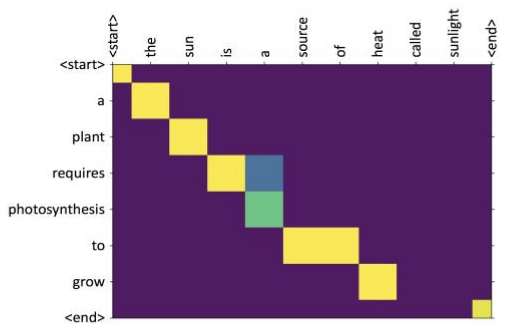


Figure 4. Vector similarities between words of related sentences after transformer encoding. Yellow represents maximum similarity by dot product, dark being minimum similarity.

this attention distribution is that they actually all follow the same pattern. Position 6 of the compressed sequences always paid all of their attentions on the second token of the inputs, position 2 always paid attention on the start and the third and fourth tokens, the rest of the positions' attention weights diagonally goes from bottom left to top right, and the attention gets stretched depending on the size of the input sentence. This is problematic for the graph embedding because the compressed values for the input sentences are completely position-dependent and semantic meaning of the sentence has no affect on the compression.

To determine whether encoded representations of related sentences have matching concepts to facilitate multi-hop reasoning, the Euclidean similarities between the tokens of two related facts were visualized. For example, encountering the sentences "Tree is a living thing" and "Poison kills living things", the model would have to encode similar signatures to the "living thing(s)" in the sentences so that they can be connected by vector similarity. As shown in figure 3, this is far from the case. Each token of the sentence is the most similar to the token of another sentence that shares the same relative distance within the sentence, which, once again, makes the encoding process completely position-dependent. This is likely caused by the addition of positional encoding into the input sentence to let the model know the position of each token in the sequence (because without the encoding, a transformer is unable to discriminate the position of the tokens since they are all treated the same[4]). For the reasons above, the method of using transformer to embed the knowledge graph was not further pursued.

3.2 OpenIE as a graph embedder

OpenIE was used to embed the knowledge graph since the transformer method of embedding graph does not encode meaningful representations of facts. OpenIE does have a few bugs, however, as it sometimes ignores parts of a sentence for unknown reasons, ignores an entire sentence, or not segregate a long segment of a sentence into smaller chunks (which dilutes the precise meaning of the segment after averaging all the word vectors). Luckily, OpenIE also provides multiple extractions for a sentence, and each extraction often

Concept 1	Relationship1	Concept 2	Relationship 2	Concept 3	Original Fact
bird	is	pollinating animal			bird is pollinating animal
chemical change	is	acid	breaking down	substances	an example of a chemical change is acid breaking down substances
fog	is formed	water vapor	condensing	in air	fog is formed by water vapor condensing in the air
car engine	converts	gasoline	converts	into motion	a car engine usually converts gasoline into motion and heat through combustion.
radiator	is	radiator	is source of	heat	a radiator is a source of heat.

Table 1. OpenIE relationship extraction samples after cleaning. The last two samples are slightly buggy results.

times compensates for the lack of information of the others. These extractions were programmatically combined in a way that minimizes the word count of each element of the triple while minimizing loss of information, as show in figure X. Not all facts could be converted into concept relationships due to errors during the extraction, which led to approximately 5% of the facts being ignored, with 13046 concepts remaining and with approximately 7440 relationships connecting them. To further minimize the number of words in a concept, repeating words or phrases like “an example of” , “the”, or “an” that do not significantly contribute to the meaning of a relationship were removed.

After the concepts were clustered using K-means to form the embedded knowledge graph, it had a final loss of 156870, with initial loss of 247042. This method clustered simple concepts accurately. For example, the two facts “Tree is a living thing” and “Poison kills living things” formed connections since “living thing” and “living things” are extremely similar concepts after taking their word embedding average, but the two facts “fog is formed by water vapor condensing in the air” and “swamps are uncultivated bodies of water also known as marshes.” did not form connections because vectors for “water vapor” and “uncultivated bodies of water” are too different. This method does not capture some important connections that are necessary for solving some of the questions in the dataset.

3.3 Performance on questions

The GNN model described in the method was trained for 50000 steps with batch size of 16. There were approximately 5000 training and 500 validation and testing samples. The main GNN model roughly achieved 47% accuracy on the testing dataset, which is far lower than the 83% achieved by the highest performing model in the OpenbookQA leaderboard. To determine whether the GNN improves the model’s performance, various parts of the GNN model were ablated (table 2). Just LSTM by itself can reach 47% accuracy, the same as a working GNN, but adding more layers does not

Model	Accuracy
1 LSTM	47%
2 LSTM	47%
LSTM + random graph	29%
LSTM + shuffled nodes	26%
LSTM + shuffled nodes + random write	25%
LSTM + random edges	44%
LSTM + random write + graph	49%
LSTM + graph	47%
OpenbookQA leader	83%

Table 2. Accuracy of tested models on OpenbookQA.

have a significant effect on accuracy. Interestingly, the accuracy of the model drops by around 20% when either the graph nodes' features are randomized (random graph and shuffled nodes). This suggests that a knowledge graph with properly embedded features is actually necessary for introducing meaningful biases to the inference process. Finally, randomly selecting the nodes to write the encoded question or having random connections between nodes did not significantly reduce the accuracy.

4.0 Discussion

This project attempted to build a deep learning workflow to specifically tackle a dataset that theoretically benefits from a graphical and multi-hop approach. Using transformer to embed the knowledge graph mostly ended up in failure, since the transformer-encoded outputs are heavily contaminated by the positional encoding. Also, output of transformer compression is not semantically meaningful, and it is unclear at this point whether there is an alternate approach for compression.

Intriguingly, experiment with GNN showed that it's performance significantly depended on the content of the embedded knowledge graph. While the current model is underperforming in efficiency and accuracy compared to simpler or more established models, there may actually be room for improvement if the knowledge graph is embedded using better methods. As mentioned earlier, OpenIE contains bugs that cannot accurately generate triples for complex and long sentences and it could not be fixed at the time of writing this report. Furthermore, averaging the word embedding vector values from the relationship triple may be a naïve approach, since it could dilute the meaning of each word, making the overall concept vector less meaningful. Lastly, it is worth noting that the model built for this project treated all edges as featureless for the sake of simplicity. Adding edge feature increases the expressiveness of the model which could improve accuracy (albeit at a cost to performance).

The GNN model was not without its flaws though. One issue was that it was unaffected by the connectivity between nodes or by the nodes selected to write the question context vector. This suggests that the model was learning to make use of all nodes of the knowledge graph rather than learning to use only a few nodes for specific questions like in discrete reasoning. This could be the case since the total loss after K-means clustering during graph embedding was only half of the starting loss. Another explanation is that the method of choosing the nodes to write was random-like, but it is unknown at the time of this writing whether better methods exists. Evidently, more work needs to go into better understanding the cause of this issue.

5.0 Conclusion

This report investigated a graphical approach to solving an NLP task that involved multi-hop reasoning called OpenbookQA. The building of knowledge graph using transformer was difficult because the encoded output was heavily contaminated by the addition of positional encoding. Because of this, the knowledge graph was built by extracting relationship triples from facts using OpenIE. The model using graph neural network with the knowledge graph reached 47% accuracy when it processed the multi-hop questions. This model heavily underperformed when compared to top performers on the investigated task, and was outperformed efficiency wise when compared to a basic LSTM benchmark model. However, the model's accuracy reduced drastically when the knowledge graph was replaced with a randomly initialized graph of the same size and structure, which showed that the embedding of knowledge was actually beneficial for the model performance. The model sometimes do not behave expectedly and there are still some issues involving the way it is calculating the loss, but once these issues are resolved it may be possible to improve the model's performance further.

6.0 Bibliography

1. Radford, A. *et al.* Language Models are Unsupervised Multitask Learners. (2018).
2. Kenton, M. C., Kristina, L. & Devlin, J. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (2018).
3. Hamrick, J. B. *et al.* Relational inductive biases, deep learning, and graph networks. 1–40
4. Vaswani, A. *et al.* Attention is all you need. in *Advances in Neural Information Processing Systems* **2017-December**, 5999–6009 (Neural information processing systems foundation, 2017).
5. Angeli, G., Premkumar, M. J. J. & Manning, C. D. Leveraging Linguistic Structure For Open Domain Information Extraction. *Assoc. Comput. Linguist.* **1**, 344–354 (2015).
6. Luan, Y., He, L., Ostendorf, M. & Hajishirzi, H. Multi-Task Identification of Entities, Relations, and Coreference for Scientific Knowledge Graph Construction. *Proc. 2018 Conf. Empir. Methods Nat. Lang. Process. EMNLP 2018* 3219–3232 (2018).
7. Pennington, J., Socher, R. & Manning, C. D. Glove: Global Vectors for Word Representation. *Assoc. Comput. Linguist.* 1532–1543 (2014). doi:10.3115/V1/D14-1162