Knowledge Graph and Fusion Based Transformer Approach for Multi-Hop Question Answering

Akash Gujju **Arun Baalaaji** **Mukkesh Ganesh** **Trisha Mandal** **Balaji Chidambaram**

1 Introduction

One of the fundamental downstream task in NLP is Question Answering(QA). The QA problem provides a measurable and objective way to test the reasoning capacity of any intelligent systems. With the advancements in machine reasoning, a good natural language system should have the ability to perform multi-hop reasoning, where the system has to reason with information taken from more than one document to arrive at the answer. Most real-world questions require multi-hop reasoning to arrive at an answer. For multi-hop reasoning, a machine must understand the question, identify supporting facts from multiple knowledge sources and use reasoning to generate an answer. In this project, we want to focus on exploring various fusion techniques and experimenting with knowledge-based information retrieval systems.

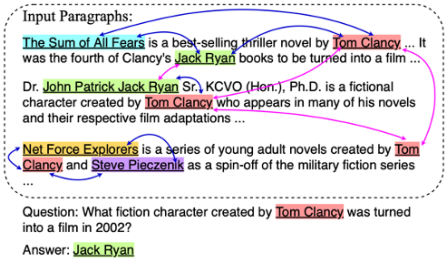

Figure 1: An example of a multi-hop question, where "The Sum of All Fears" is the missing entity. Note from the search results that it cannot be easily retrieved based on merely the question.

2 Related Work

2.1 GoldEn Retriever

The GoldEn Retriever model(Qi et al., 2019) is a retrieve-and-read system used to answer complex questions that involve multiple steps of reasoning in an open-context open-domain setting. The model answers these questions by iterating between understanding the context and generating natural language queries to search for supporting facts to read.

2.2 Hotpot baseline

The hotpot baseline model(Yang et al., 2018) released alongside the HotpotQA dataset performed relatively similar on the SQuAD dataset(Rajpurkar et al., 2018). The model incorporates shared normalization along with a three way classifier output. The classifier output either denote a text span or a yes/no answer. The model also uses bi-attention when fusing multiple context embeddings. The model's paragraph selector makes use of TF-IDF to evaluate paragraph importance given an input question. The proposed fusion experiments where implemented on this baseline architecture.

2.3 Dynamically Fused Graph Network (DFGN)

Answering complex questions involve understanding multiple paragraphs simultaneously. The DFGN model(Xiao et al., 2019) creates a dynamic entity graph for every question and performs iterative query update to arrive at an correct answer from the multiple input documents. This iterative query update mechanism is inspired by the human's step-by-step reasoning behavior. We study the importance of KG based information retrieval using this model.

3 Dataset

3.1 HotpotQA

HotpotQA(Yang et al., 2018) dataset is a challenging dataset created for the Multi-Hop Reasoning problem with Wikipedia QA pairs. The training, validation, and test data for the HotpotQA dataset are presented in native JSON format. Each training record has a question, an answer, facts to back up

the answer, and paragraph contexts. As an example of the whole pipeline of experiments being done on the HotpotQA dataset, including data download, data preprocessing, training, and assessment, a baseline model is given along with the dataset. The distractor setting and the full wiki setting are the two separate settings for the Hotpot model. Our model will be assessed in a complete wiki environment. The dataset contains over 90K training records and 22K dev/test records. The complexity of the questions found in the dataset can be seen in the table 2.

Name	Desc.	Usage	# Examples
train-easy	single-hop	training	18,089
train-medium	multi-hop	training	56,814
train-hard	hard multi-hop	training	15,661
dev	hard multi-hop	dev	7,405
test-distractor	hard multi-hop	test	7,405
test-fullwiki	hard multi-hop	test	7,405
Total			112,779

Figure 2: The table contains categories of train and test data

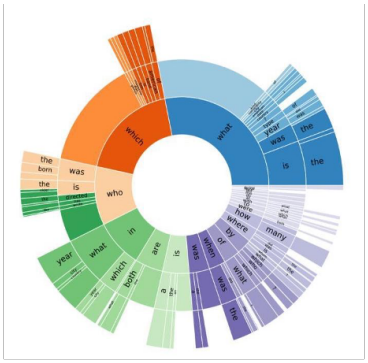

Figure 3: The figure contains the categories of questions

4 Methodology

4.1 Dynamically Fused Graph Network (DFGN)

We describe Dynamically Fused Graph Network in this section. We follow the normal human reasoning QA process in this system. We start with the entity of interest and then focus on the surrounding words near the entity, then we try to connect it with some related entity and repeat the same process to form a reasoning chain. This model has four key components namely, paragraph selector, entity graph construction, query-context encoder and DFGN fusion block.

Paragraph Selector uses a sub-network to select relevant paragraphs since not all paragraphs will be relevant to the question. The sub-network is designed with a pre-trained BERT (Devlin et al., 2018) followed by a sentence classification layer with sigmoid activation. The paragraph selector network takes the input query and a candidate document and outputs a relevance score between 0 and 1. If the paragraph has atleast one supporting sentence for each QA pair, a training label of 1 is assigned. Paragraphs with scores than η are selected and concatenated together as the context C.

Entity Graph is created using the filtered context from the paragraph selector which is dynamically generated for each question. We use the knowledge base with the named entities extracted from the context C. Named Entity Recognition(NER) is run on the input context to identify the various entities and categorize them. These entities and their classification type help in generating the entity graph.

Query-context encoding, a BERT encoder is used to produce a representation from the input query and context paragraph using a Bi-Attention layer. The bi-attention layer is used to enable interactions between the query and the context. In experiments, we find out that inserting a bi-attention layer achieves better performance than the BERT encoding.

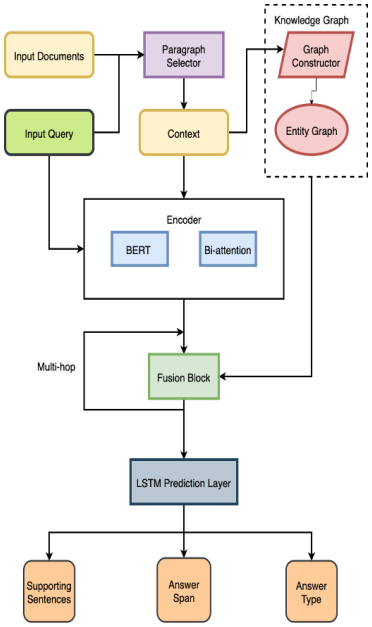

Figure 4: DFGN Fusion Block

DFGN Fusion Block takes in the tokens from the input context and calculates the entity embeddings. It is denoted as Doc2Graph in the figure 5. Graph Neural Network(GNN) with message passing algorithm is used to propagate information to the neighboring nodes. More importance is given to nodes which are closely related to the input query. In the Graph2Doc block, information is passed back from the entity nodes to the context tokens.

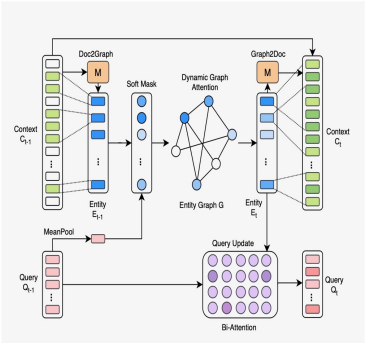

Figure 5: Reasoning with Fusion block in DFGN

5 Fusion Methods

Almost all implementations of the baseline models for HotpotQA make use of some attention-based skip-connection based blocks. In these blocks, they usually just concatenate the output of two blocks to be fed into the newer blocks. This proves as an ideal area to experiment with different fusion techniques, for potentially better performance. We are replacing the vanilla linear sum concatenations with these fusion blocks. These are some of the fusion techniques we have primarily coded up as alternatives to concatenation.

5.1 MCB

MCB is also known as Multimodal Compact Bilinear Pooling (Fukui et al., 2016). This is predominantly used in Visual Question Answering, where the Vision Embeddings and Text Embeddings are joined together for downstream tasks. The two embeddings are randomly projected to a higher dimensional space. By using an element-wise Fast Fourier Transform, we then convolve both vectors. We make use of count sketch projection to project the embeddings to the higher dimensional space.

5.2 MUTAN

MUTAN (Ben-Younes et al., 2017) is also known as Multimodal Tucker Fusion. This is once again,

```
def forward(ctx, h1, s1, h2, s2, output_size, x, y, force_cpu_scatter_add=False):
    ctx.save_for_backward(h1,s1,h2,s2,x,y)
    ctx.x_size = tuple(x.size())
    ctx.y_size = tuple(y.size())
    ctx.force_cpu_scatter_add = force_cpu_scatter_add
    ctx.output_size = output_size

    # Compute the count sketch of each input
    px = CountSketchFn_forward(h1, s1, output_size, x, force_cpu_scatter_add)
    fx = torch.rfft(px,1)
    re_fx = fx.select(-1, 0)
    im_fx = fx.select(-1, 1)
    del px
    py = CountSketchFn_forward(h2, s2, output_size, y, force_cpu_scatter_add)
    fy = torch.rfft(py,1)
    re_fy = fy.select(-1,0)
    im_fy = fy.select(-1,1)
    del py

    # Convolution of the two sketch using an FFT.
    # Compute the FFT of each sketch


    # Complex multiplication
    re_prod, im_prod = ComplexMultiply_forward(re_fx,im_fx,re_fy,im_fy)

    # Back to real domain
    # The imaginary part should be zero's
    re = torch.irfft(torch.stack((re_prod, im_prod), re_prod.dim()), 1, signal_sizes=(output_size,))

    return re
```

Figure 6: Code Snippet from MCB Block

used extensively in the VQA domain, where the fusion of embeddings are more important. This method is a tensor-based Tucker decomposition method to efficiently parameterize bilinear interactions between 2 embeddings. The authors of this paper have also designed a low-rank matrix-based decomposition to constrain the interaction rank.

```
def forward(self, x):
    x0 = self.linear0(x[0])
    x1 = self.linear1(x[1])

    if self.dropout_input > 0:
        x0 = F.dropout(x0, p=self.dropout_input, training=self.training)
        x1 = F.dropout(x1, p=self.dropout_input, training=self.training)

    m0 = self.merge_linear0(x0)
    m1 = self.merge_linear1(x1)
    m = m0 * m1
    m = m.view(-1, self.rank, self.mm_dim)
    z = torch.sum(m, 1)
    if self.normalize:
        z = torch.sqrt(F.relu(z)) - torch.sqrt(F.relu(-z))
        z = F.normalize(z, p=2)

    if self.dropout_pre_lin > 0:
        z = F.dropout(z, p=self.dropout_pre_lin, training=self.training)

    z = self.linear_out(z)

    if self.dropout_output > 0:
        z = F.dropout(z, p=self.dropout_output, training=self.training)
    return z
```

Figure 7: Code Snippet from MUTAN Block

6 Experiments

Fusion experiments

In our model, we are fusing the Question Attention embeddings with the Context Attention embeddings. Complex fusion techniques has been shown to improve the convergence rate of the models and slightly increase the performance. We studied two fusion techniques:

- Multi Modal Tucker Fusion
- Multi Modal Factorized Bi-Linear pooling(MFB)

The hotpot-baseline model passes both the context paragraphs and the questions through a character RNN and a Word embeddings. These embeddings are passed through a bi-attention layer and pass through multiple RNN layers. Each of these RNN blocks have skip connections that just use linear sum to concatenate the outputs from the previous layers. Here we replace the linear sum components with the above mentioned fusion techniques for various experiments. We retain the output dimensions after fusion to ensure model compatibility.

Fine-tuning BERT with Causal Language Modeling(CLM)
Since our QA dataset was primarily built on paragraphs from Wikipedia, fine-tuning the BERT model on wikitext would aid the model in generating better language understanding. WikiText-2 corpus is mainly used for language modeling tasks and it contains a larger vocabulary focused on quality wikipedia articles(Merity et al., 2016). We make use of hugging face trainers to fine tune our BERT(bert-base-uncased) model with the WikiText-2 corpus.

7 Results

We set 0.50 as our target F1 for our fusion experiments. The baseline Hotpot model with linear sum achieved a score of 0.50 after 18 epochs. Whereas, MFB took 12 epochs, Tucker Fusion took 14 epochs and the combination of MFB + Tucker Fusion took 15 epochs to surpass the given target F1.
All the fusion models where evaluated after 18 epochs as show in Table 1. The MFB + Tucker Fusion model was the best performing fusion model while achieving a F1 score of 0.5246. Moreover, all the fusion models consistently outperformed the linear-sum baseline.
The baseline model only used off-the-shelf information retrieval system to select the context paragraphs. Whereas the DFGN model incorporated the context paragraphs into a dynamically generated KG which vastly increased the F1 score of the model from 0.50 to 0.65.

Further the F1 score increased from 0.60 to 0.65 after the DFGN baseline BERT model was replaced with WikiText2 Fine-Tuned BERT model. This shows that higher embedding quality can be attained by fine-tuning existing model with newer and relvant corpuses.

Model	F1 Score	EM	Precision	Recall
Fusion Baseline	0.5104	0.3762	0.5300	0.5297
MFB	0.5186	0.3831	0.5393	0.5362
MFB + Tucker Fusion	0.5246	0.3941	0.5480	0.5442
Tucker Fusion	0.5155	0.3808	0.5363	0.5335
DFGN baseline	0.6030	0.4708	0.6033	0.6011
Fine-tuned DFGN	0.6500	0.5008	0.6737	0.6672

Table 1: Models Performance

8 Conclusion

Based on the results of our fusion experiments, we can conclude that faster convergence can achieved when fusion techniques are employed. Additionally, we inferred that complex fusion techniques consistently outperform the linear-sum baseline model. Use of KG based retrieval systems greatly increases the model performance when compared with Non KG based approaches. KG based models tend to closely model the relationships needed to answer multi-hop questions. Fine-tuning the BERT model with latest and relevant corpuses increase the embedding quality which in-turn led to superior performance.

9 Future Work

- We plan on utilizing multi-GPU training to overcome the memory and compute requirements.
- We plan to incorporate fusion techniques on dynamic graph fused network.
- We plan to experiment with BeerQA dataset.
- We plan to fine-tune the BERT model on DBPedia and ConceptNet.
- We also can try scaling the dataset by only using 75% of the training set.
- We can run our experiments on the distractor setting since they require only 10 paragraphs.

10 Division of Labour

Dataset preparation and environment setup: Akash Gujju, Trisha Mandal, Balaji Chidambaram
Baseline Model: Balaji Chidambaram, Akash Gujju, Trisha Mandal
DFGN baseline and BERT fine-tuning: Arun Baalaaji

Fusion and language models: Mukkesh Ganesh
Documentation: All

11 Code Repo

https://github.com/arunbaalaajis/nlp-project-39

References

Hédi Ben-Younes, Rémi Cadène, Matthieu Cord, and Nicolas Thome. 2017. MUTAN: multimodal tucker fusion for visual question answering. *CoRR*, abs/1705.06676.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. *CoRR*, abs/1606.01847.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *CoRR*, abs/1609.07843.

Peng Qi, Xiaowen Lin, Leo Mehr, Zijian Wang, and Christopher D. Manning. 2019. Answering complex open-domain questions through iterative query generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2590–2602, Hong Kong, China. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822.

Yunxuan Xiao, Yanru Qu, Lin Qiu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. 2019. Dynamically fused graph network for multi-hop reasoning. *CoRR*, abs/1905.06933.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.