２０２２年度 修士論文

Answer Prediction for Commonsense Multiple-Choice Questions by Utilizing Pretrained Language Model, Knowledge Graph and Additional Information

指導教員 岩井原 瑞穂 教授

早稲田大学大学院情報生産システム研究科

情報生産システム工学専攻 データ工学研究

44211051-1 LI Jiaxin

Abstract

Our basic understanding of human beings and the world could be reflected by the basic knowledge used in commonsense reasoning. Commonsense question answering contains different types of commonsense knowledge to predict the final answers from five choices. The purpose of the commonsense question answering is to test whether the model could answer the commonsense questions that everyone knows. In our model, we use pretrained language model (PLM), knowledge graph (KG) and additional information to predict the final answer for commonsense multiple-choice questions from the five choices. First, we need to encode the question and choices as QA context. Then, we will retrieve subgraphs of the knowledge graph from the large knowledge graph ConceptNet. We also calculate the relevance score between QA context and KG nodes. For better reasoning or pruning paths in the working graph, we also use a self-attention graph pooling (SAGPooling) to do message passing with multiple rounds. Moreover, we merge the information of the knowledge graph to the information of the pretrained language model. Only questions and choices are not enough for predicting the correct answer. In this case, we also have additional information, such as statements, answer choice meanings and triples. In the end, we concatenate the QA context representation, the PLM representation, and the pooled working graph representation to predict the final answer with the probability score. In our task, we test our model on both RoBERTa-base and RoBERTa-large pretrained language models. Several attempts have improved some have not. Overall though, the performance of our model improves compared to the baseline model so far.

Keywords: Question Answering, ConceptNet Knowledge Graph, Pretrained Language Model, CommonsenseQA, Wikitionary, Triples.

Content

Abstract…… 2

Content…… 3

List of Figures…… 5

List of Tables……..…….... 6

1. Introduction…… 7

2. Related Work…… 9

2.1 RoBERTa Model…… 9

2.2 Encoder-BiLSTM…… 9

2.3 Modified MLP…….. 9

2.4 Self-Attention Graph Pooling…… 10

2.5 Wikitionary & Triple…… 10

3. Problem Definition…… 11

4. Methodology…… 13

4.1 Baseline Model…… 13

4.2 Model Structure…… 14

4.2.1 Proposed Method 1 (BiLSTM)…… 14

4.2.2 Proposed Method 2 (Modified MLP)…….. 14

4.2.3 Proposed Method 3 (SAGPooling)…….. 15

4.2.4 Proposed Method 4 (KGPLM)…… 16

4.2.5 Proposed Method 5 (Statement & Wikitionary & Triple)…… 19

4.2.6 Overall Model Structure…… 21

5. Experiments…… 23

5.1 Datasets……... 23

5.1.1 CommonsenseQA Dataset…… 23

5.1.2 ConceptNet Knowledge Graph…… 24

5.2 Experiment Settings…….. 25

5.3 Experiment Results…… 26

5.4 Results Analysis…….. 31

6. Conclusion & Future Work…… 33

Acknowledgement…… 34
References…… 35

List of Figures

Figure 1 An example of commonsense multiple-choice question

Figure 2 The architecture of the baseline model QAGNN

Figure 3 The architecture of the modified MLP

Figure 4 The architecture of the KGPLM

Figure 5 Previous input structure

Figure 6 Updated input structure

Figure 7 The architecture of the overall model

Figure 8 An example of the CommonsenseQA

Figure 9 Updated each QA context

Figure 10 A sample of commonsense knowledge in ConceptNet

Figure 11 The flowchart of testing methods with different token sets

List of Tables

Table 1 CommonsenseQA data statistics

Table 2 Experiment Parameters

Table 3 Results on RoBERTa-base model

Table 4 Results on RoBERTa-large model

Table 5 Comparation of Softmax and Minmax based on Scaled Dot Prodcut

Table 6 The results of testing merging functions

Table 7 The results for different input sets

1. Introduction

In everyone's student days, we all have multiple-choice questions when taking exams and doing assignments. The fields they cover may be in mathematics, medicine, English reading comprehension and so on. In order to choose the final answer from the five options, we need to use what we have learned in school to find the correct answer. Today, we already have technology that can help us to find out the answer through model training and learning. Here, we will introduce the question answering model.

Question answering model is a deep learning model that could answer questions in a given context. Also, question answering is a computer science discipline within the fields of information retrieval and natural language processing (NLP), which is concerned with building systems that automatically answer questions posed by humans in a natural language. Question answering in natural language processing is a very popular task in recent years. It could be used for the Google searches, conversational questions, and multiple-choice questions we are going to introduce about today.

Two major paradigms have been used in question answering systems. One is knowledge-based question answering (KBQA) [8] and another one is information-retrieval-based question answering (IRQA) [8]. For knowledge-based question answering (KBQA), it is an important task in natural language processing. The purpose of using the knowledge-based question answering is to answer a question over a knowledge base. For information-retrieval-based question answering (IRQA), the question answering model could extract answer phrases from paragraphs or paraphrase the answer generally. The models need to have a semantic understanding of the question and the context, understand the structure of the language, could locate and find the position of an answer phrase or span, and so on.

For common sense reasoning on CommonsenseQA [21], in some early question answering models, they mainly use the language model as the main model, which could be too simple and not effective enough. Also, a knowledge graph [23] is a knowledge base which adopts a graph-structured data model to represent data for knowledge reasoning. Knowledge graphs are widely used in many applications, such as biomedicine, pharmaceuticals and network security. Therefore, in our model, we use both pretrained language model and knowledge graph. We use RoBERTa [13] as our pretrained language model and ConceptNet [20] as our knowledge graph.

In our task, we mainly use the knowledge-based question answering (KBQA) [8]. Our model needs to make judgments and predict the final answer based on commonsense knowledge that everyone knows. For example, Figure 1 shows an example of commonsense multiple-choice question. In this case, the blue words are topic entities of the question, the red words are answer choices, and the final answer is “bank” for this question. The ultimate goal of our model is to predict the correct answer based on the commonsense domain.

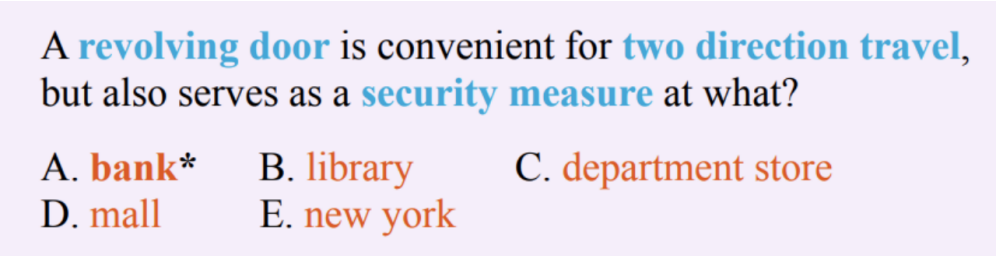

Figure 1 [26]: An example of commonsense multiple-choice question.

2. Related Work

2.1 RoBERTa Model

Our baseline model is from QAGNN [26]. The QAGNN model uses a pretrained language model and a knowledge graph to predict an answer from the commonsense question answering dataset. They use the RoBERTa-large model as their encoder. RoBERTa [13] is a robustly optimized BERT pretrained language model. Their best model achieves state-of-the-art results on many datasets, such as RACE, GLUE and SQuAD. Therefore, in our model, we use the pretrained language models, RoBERTa-base and RoBERTa-large, to encode QA contexts and obtain language model representation. We also use these two Roberta models to calculate the relevance score of each node in subgraphs of a knowledge graph. In this case, the RoBERTa model also has great performance on CommonsenseQA dataset which is mainly used in our question answering task.

2.2 Encoder-BiLSTM

In order to increase the accuracy of our model, we add the bidirectional Long Short-Term Memory (BiLSTM) [19] after retrieving and obtaining representations from the RoBERTa encoder. Pearce, K. et al. [17] propose an ensemble model architecture using BERT and BiLSTM, and evaluate its performance against standard pretrained models on extractive question answering. The performance of BERT-BiLSTM is better than the performance of only using one BERT encoder. In this case, we add a BiLSTM after the RoBERTa encoder in our model for better encoding.

2.3 Modified MLP

In order to increase the accuracy of our model, we also use the modified multilayer perceptron (MLP) [7] to concatenate the representations and calculate the probability score of being the final answer. The modified MLP mentioned in this thesis is mainly applied to classification, and it is effectively used in the classification task compared to the traditional MLP. Thus, in our question answering task, we use this kind of MLP for improving the accuracy of our model. We try two different activation functions, Leaky Rectified Linear Unit (LeakyReLU) [7]

and Mish [15], because they could have better performance, lower computational cost and lower standard deviations than Rectified Linear Unit (ReLU)'s for some models, like YOLOv4 [5] model.

2.4 Self-Attention Graph Pooling

From the baseline model QAGNN [26], we need to encode the input through a language model, and retrieve a subgraph from the large knowledge graph ConceptNet [20]. ConceptNet [20] could connect words and phrases of natural language with labeled edges. Its knowledge is collected from many sources which were created by experts, crowd-sourcing and so on. It is designed to represent the general knowledge involved in understanding language, improving natural language applications by allowing the application to better understand the meanings behind the words people use. Now, we can observe that each step is done separately, and then enter representation of each step to the MLP finally to predict the correct answer.

Therefore, we apply a self-attention graph pooling (SAGPooling) [11] to the knowledge graph conditioned on the information of the pretrained language model. The SAGPooling [11] proposes a graph pooling method based on self-attention. We use it in our model to better help models to reason or prune irrelevant paths based on the information of the pretrained language model.

2.5 Wikitionary and Triples

In our baseline model QAGNN, there are only two inputs, a question and five answer choices. For better reasoning with more information, we try to add additional information, such as statement, Wikitionary [24] and triples, to our dataset. Xu, Y. et al. [25] use Wikitionary [24] and triples as the input, and use the Albert model to produce the prediction for commonsense reasoning. Overall, it performs better than most models. In this thesis, we try to add these additional information as our new input, and let our model train from these information.

3. Problem Definition

Given a question q, an answer choice a, a triple t, a statement s, and an answer choice meaning acm, we concatenate them into a *QA context*:

$$[q, a, t, s, acm]$$

Here, we denote the QA context as the input x, and we first use the pretrained language model to encode the QA context. The encoder function $f_{enc}(x)$ maps an input to a vector representation. We define the knowledge graph as a multi-relational graph, $G = (V, E)$, where V is the set of entity nodes in the knowledge graph, and E is the set of edges. E is a subset of $V - R - V$, where R is a set of relation types that connects nodes in V.

We connect a new QA context node z which represents the QA context to each topic entity in $V_{q,a}$ on the KG subgraph G_{sub}, where $V_{q,a} = V_q \cup V_a$. We retrieve the knowledge subgraph from the ConceptNet knowledge graph G which consists of all the nodes on the k-hop paths between nodes in $V_{q,a}$.

$$G_{sub}^{q,a} = (V_{sub}^{q,a}, E_{sub}^{q,a})$$

In the graph, each node is associated with one of the four types:

$$T = \{Q, A, Z, O\},$$

where Q means nodes in V_q, A means nodes in V_a, Z means the context node z, and O means other nodes. To calculate the relevance score of each node in the knowledge graph, we use the function below:

$$\rho_v = f_{rs}(f_{enc}([text(z); text(v)])), \text{ [26]}$$

where f_{rs} uses the representation of encoder f_{enc} to do the score calculating task. We need to concatenate the entity of $text(z)$ in the QA context with each node $text(v)$ in the KG, and then calculate the relevance score of each knowledge graph node using function ρ_v.

We then have a 2-layer MLP to calculate the probability of being the final answer with the function below:

$$p(a|q) \propto \exp\left(MLP(z^{LM}, z^{GNN}, g)\right). \text{ [26]}$$

That is, we could concatenate the language model representation, the QA context representation and the pooled graph representation to the MLP for calculating the probability score of being the correct answer.

Our aim is to predict the final answer for commonsense question answering with five choices by utilizing pretrained language model (PLM), knowledge graph (KG) and additional information.

4. Methodology

4.1 Baseline model

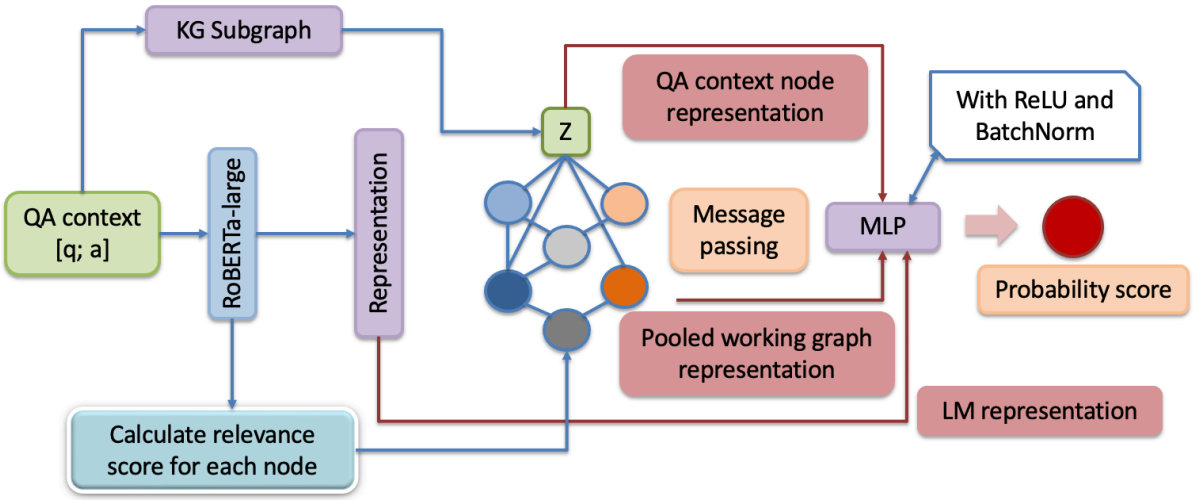

Figure 2: The architecture of the baseline model QAGNN.

As shown in Figure 2, given a question and choices, we first need to encode them. For this encoding part, we can denote the given question and choices as a QA context, and set this QA context as the input. That is, the input should be a QA context contains a question and five choices. We then use a pretrained language model to encode the given QA context. The original paper [26] uses the pretrained RoBERTa-large model. After we obtain the language model representation, we then need to focus on the knowledge graph part. We first retrieve subgraphs from the ConceptNet [20], the commonsense knowledge graph, conditioned on the entities from the QA context.

There are still a large number of irrelevant paths in the retrieved subgraph. In order to prune and reason effectively, we need to calculate the relevance score of each node in the subgraph with the entities in the QA context by using the pretrained language model. For reasoning on the knowledge graph, our GNN module adopts the graph attention framework (GAT) [22], which induces node representations via iterative message passing between neighbors on the graph. It will jointly leverage and update the representation of the QA context and the knowledge graph as our GNN message passing operates on the graph. We keep top-200 nodes when we prune and reason with multiple rounds of message passing according to the relevance scores.

Following the above steps, we have obtained the QA context representation, pretrained language model representation and pooled working graph representation. Then, we can move to the final step. In this step, we concatenate these representations first. Then, we use the multi-layer perceptron (MLP) to calculate the probability score and find the final answer of question. We optimize our model using the cross-entropy loss.

4.2 Model Structure

4.2.1 Proposed Method 1 (BiLSTM)

Our main goal for this question answering task is to improve the accuracy of predicting the final answer of the commonsense question from the five choices. Therefore, we first modify two parts in order to optimize the model and check to see whether its performance has improved. In Sections 4.2.1 and 4.2.2, we will introduce two parts that we modified.

In the baseline method QAGNN, their encoding part uses the RoBERTa-large model. In our model, we use both RoBERTa-base and RoBERTa-large to do encoding task in order to check the performance of our proposed methods on different models. Compared to the baseline model QAGNN, we do not only use the pretrained language model to encode the given QA context, but also add one BiLSTM after the encoder. BiLSTM [19] is bidirectional Long Short-Term Memory which is a recurrent neural network used in NLP and could make any neural network have the sequence information in both backward and forward directions. Pearce, K. et al. [17] apply the (BiLSTM) layer after the BERT model to obtain representation for information extraction task. The result of using Encoder-BiLSTM is better than the result of using only one encoder model. Therefore, in our model, we first add a BiLSTM layer after retrieving the RoBERTa encoding to obtain more contextual representation from the inputs.

4.2.2 Proposed Method 2 (Modified MLP)

For this proposed method, we try to focus on the MLP part in order to improve the accuracy of calculating the probability score of being the final answer. The original MLP uses the ReLU activation function and batch normalization. Fiedler, et al. [7] mention that they create a modified MLP in their classification task, and the accuracy improves compared to normal MLP which uses ReLU as the activation function and batch normalization. Therefore, we intend to modify the MLP

in our question answering task. Figure 3 shows the architecture of our modified MLP. We try to use the two different activation functions, LeakyReLU [7] and Mish [15], instead of the ReLU function. LeakyReLU is a type of activation function based on a ReLU, but it has a small slope for negative values instead of a flat slope. Mish function is a novel self-regularized non-monotonic activation function. In addition, we use the Ghost batch normalization [7], a modified version of batch normalization that normalizes the mean and variance for disjoint sub-batches of the full batch, instead of the batch normalization. Thus, we expect this proposed method to work well in our model.

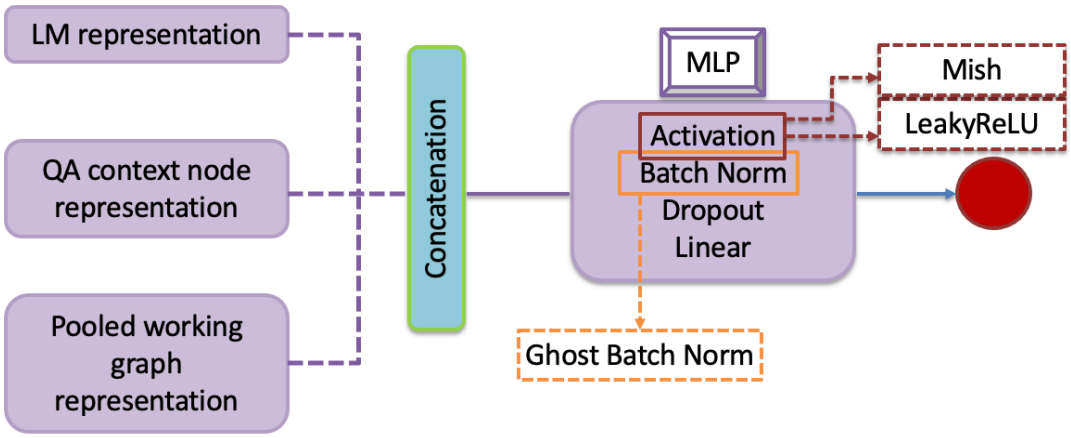

Figure 3: The architecture of the modified MLP.

4.2.3 Proposed Method 3 (SAGPooling)

In our model, we use the pretrained language model to calculate the relevance score of each node between graph nodes and entities from the QA context, which is the same as the baseline model QAGNN. Doing this step is to prune the irrelevant nodes and reason the possible answer. In this case, the head nodes of the subgraph are topic entities of the question from the QA context. There are many relation edges with different types between each node. We can do message passing with multiple rounds to do reasoning and pruning.

However, in the baseline model QAGNN, each task is done separately and then concatenate them to the final MLP to predict the correct answer. That is, the encoding task could be done by

using a pretrained language model. The pruning or reasoning part could be done by message passing in the working graph conditioned on the relevance score. They only use the relevance scores between nodes of the subgraph and entities of the QA context when they reason or prune, and there is no relational information from the encoder.

Thus, we try to add a self-attention graph pooling layer with the information of encoder representation to the knowledge graph in order to better save the related paths or prune the irrelevant paths. In other words, the self-attention graph pooling layer also needs to prune and reason based on the information of the encoder.

4.2.4 Proposed Method 4 (KGPLM---Merge KG Information with PLM Information)

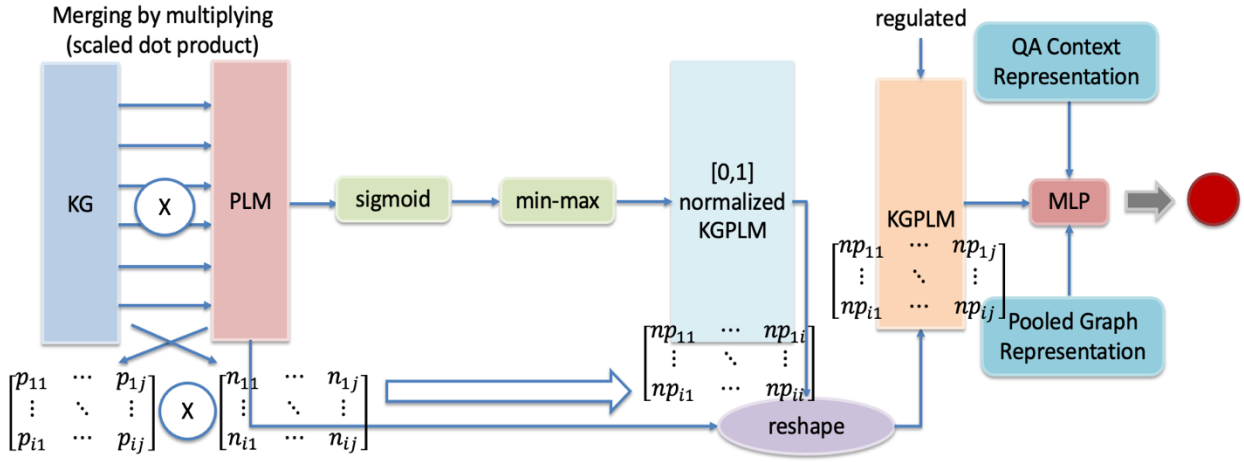

Figure 4: The architecture of KGPLM.

In the original work [26] of the question answering model, the outputs from the pretrained language model RoBERTa and knowledge graph are injected individually to the final MLP. However, inputting such information individually may not help the model clearly learn the relation between such information. In this situation, we consider trying to merge the information from the knowledge graph (KG) with the information from the pretrained language model (PLM) before concatenating them to the MLP. In our model, we mainly use the RoBERTa pretrained language model and the ConceptNet as our knowledge graph. We believe by doing so, the information in

both knowledge graph and pretrained language model would be pre-interacted with each other, and could become easier to be trained or learned by the model.

To merge the information of the KG and the PLM, we need to build a merge function with appropriate methods to help mix the information up, thus, the scaling mechanism would be discussed here. According to Figure 4, when we merge the KG information to the PLM information, we assume that a KG subgraph is extracted and represented as an $i \times j$ matrix with n nodes, and the output from the PLM is represented as an $i \times j$ matrix with p nodes, where i refers to the index of each choice for the question, j refers to the index of the possibilities of input choice i. Using matrix multiplication (dot function) would be one way to fold the corresponding elements together by using the following formula. The matrices of merging are also shown below. In this step, we reference two product models which are the scaled dot product and double linear product.

$$Merge = D_{PLM} \cdot D_{KG}^{T}$$

$$\begin{bmatrix} p_{11} & \cdots & p_{1j} \\ \vdots & \ddots & \vdots \\ p_{i1} & \cdots & p_{ij} \end{bmatrix} \cdot \begin{bmatrix} n_{11} & \cdots & n_{1j} \\ \vdots & \ddots & \vdots \\ n_{i1} & \cdots & n_{ij} \end{bmatrix}^{T} = \begin{bmatrix} np_{11} & \cdots & np_{1i} \\ \vdots & \ddots & \vdots \\ np_{i1} & \cdots & np_{ii} \end{bmatrix}$$

Scaled Dot Product model [18] would help us combine the information with less chance of out of range. If we use the dot product directly, the result may have larger value, which may lead the training loss to become Nah value. As shown in the formula below, the scaled dot product model would zoom the range of the result by dividing by a given number, so that the chance of value exceeding would be lower. Additionally, our research is solving problems with high-dimensions, the variance of using dot product would be higher, which may lead low-convergency to the training model. However, using Scaled Dot Product would help with this problem as well.

$$Merge = \frac{D_{KG}^{T} \cdot D_{PLM}}{n}$$

Double Linear Product [18] model would be an alternative method for Scaled Dot Product model. Through the formular below, both side of the information would be linearized by a matrix W. By doing so, the product result would retain the distribution of origins with lower in changes of variance.

$$Merge = D_{KG}^{T} \cdot W \cdot D_{PLM}$$

However, when the measurement from the two sources is different, the result of merging might be meaningless and hard to be understood. Such a mechanism would help us merge the information from two sources and output the result with a new measurement. Since the outputs from the KG and the PLM can be regarded as possibilities of choosing corresponding input entities, we need to set a mechanism to convert the merging result back to the status of possibilities. In this case, the scaling range should give back results reflecting by real numbers in range [0,1]. In this case, we consider to use the softmax function and the minmax function to support such scaling mechanism in our research.

Softmax [16] is a useful function in logistic regression problems. This function is usually seen as activation function of an attention layer. Softmax converts the given value to the number in range [0,1] and ensures that the sum of all the converted values of dimensions in each input equals 1.

$$f(x) = \frac{e^{x}}{\sum_{i=1}^{n} e^{x_i}}$$

Minmax Standardization [1] would be another method for scaling. For the formal Minmax function showing below, it would compute the range of input by looking for minimum value and maximum values, and then gives the position of each element in that range as the output. The closer to the minimum value, the converted value would be more approaching to 0, vice versa.

$$f(x) = \frac{x - min}{max - min}$$

The problem of using it is clear. That is, there would be chance of getting 0 values when x equals the minimum value and the chance of getting 1 values when x is the maximum. Both of the cases would lead to loss explosion for the loss function of multi-class classification problems. To resolve the situations, we add two more parameters to this function. For instance, α means a scaling factor that used to zoom the model from 1 to a float number between 0 and 1. ε would be an extremely small float number used to keep the result of the minmax function greater than 0.

$$f(x) = \alpha \cdot \left(\frac{x - min}{max - min}\right) + \varepsilon \begin{cases} 0 < \alpha < 1 \\ \varepsilon = 0.000001 \end{cases}$$

Moreover, when the distribution of the inputs is extremely unbalanced, for example, only having large numbers approaching to 1 and small numbers nearing 0. This would lead to the result of using the minmax function being expressed in a small range. That would be harder for the training model to catch the difference between each input. To fix it, the sigmoid function can be utilized before using the minmax method. By using the formula below, the sigmoid function would help us enforce the difference between inputs and convert to values in range [0,1] for the first time. Then we should use Minmax method for adjustment.

$$f(x) = \frac{1}{1 + e^{-x}}$$

Now, we obtain a [0,1] normalized KGPLM. We then reshape it as the form of the original input (from an $i \times i$ matrix to $i \times j$ matrix), and obtain a regulated KGPLM:

$$\begin{bmatrix} np_{11} & \cdots & np_{1i} \\ \vdots & \ddots & \vdots \\ np_{i1} & \cdots & np_{ii} \end{bmatrix} \rightarrow \begin{bmatrix} np_{11} & \cdots & np_{1j} \\ \vdots & \ddots & \vdots \\ np_{i1} & \cdots & np_{ij} \end{bmatrix}$$

and then enter it with other two representations, the QA context representation and the pooled graph representation, to the MLP for predicting the probability of being the final answer.

4.2.5 Proposed Method 5 (Statement & Wikitionary & Triples)

In the baseline model QAGNN, the QA context consists of two types of information, only question and answer choices, as the input. Therefore, in this case, we wonder whether the performance of the model can be improved by incorporating more additional information. We refer the work by Xu, Y., et. al [25] which input the Wikitionary information and triple information to a language model. The Wikitionary [24] is a web-based multilingual project aimed at creating a free content dictionary of terms in all natural languages and some artificial languages. In their model [25], they use a dump of Wiktionary includes definitions of about 1,000,000 concepts. They find the closest match in Wiktionary for every question or choice concept. They also employ the

KCR method (Lin, 2020) [12] to select relation triples. The performance of their QA model is better than most of models, such as UnifiedQA [10] and T5 [14]. Thus, we intend to add meaning of the answer choices extracted from the Wikitionary and the relation triples based on commonsense knowledge to our dataset. This is expected to make our pretrained language model learn better based on the detailed additional information.

After incorporating the additional information, our input format of QA context is updated below:

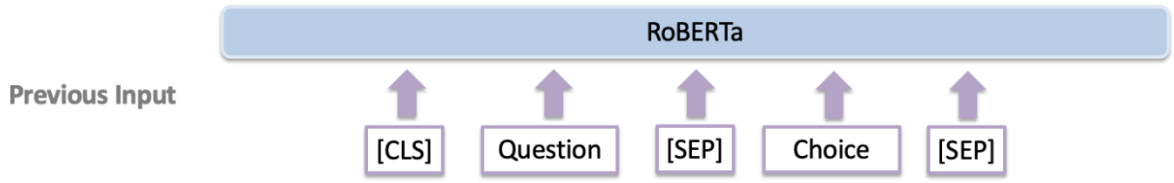

Figure 5: Previous input structure.

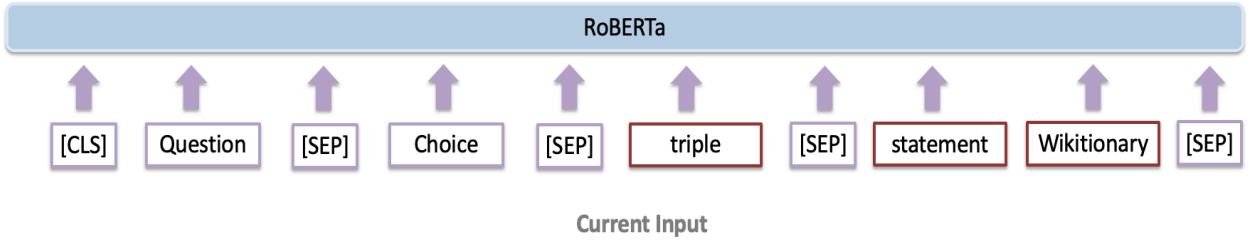

Figure 6: Updated input structure.

For previous format in Figure 5, given the question and the answer choices, we concatenate them as the QA context $[q, a]$, then enter such a QA context to the RoBERTa model. The maximum sequence length is 128. For the updated input format in Figure 6, the QA context is $[q, a, t, s, acm]$, consisting of a question, answer choices, relation triples, statement, and the meaning of answer choices (we call ACM, retrieved from Wikitionary). The maximum sequence length is 256 here.

The dataset format is updated as follows:

From:

Question_ID	Question_Entity	Answer_Choice	Labeling	Statement

To:

Question_ID	Question_Entity	Answer_Choice	Triple	Statement+ACM	ACM	Labeling

We can see that the statement information is included in the previous dataset format. However, the baseline model, QAGNN, does not apply such information to the input. In this case, we apply the statement information also as our additional information to our model, and check its performance. Here is an example of additional information:

Question: Which is the fruit?

Choice A: Apple

Statement: Apple is the fruit.

ACM (Meaning of answer choices): Apple is a kind of … (from the Wikitionary)

For the question "Which is the fruit?" and the choice A "Apple", the statement seems like a blank fill thing that questionary sentence can be a statement sentence with answer choice filled in. The ACM expresses the meaning of each answer choice retrieved from Wikitionary, giving a simple definition for the answer choice. The triple relation is having the format of [subject-predicate-object], and for instance, ["library", "PartOf", "house"] is a triple, retrieved based on the commonsense knowledge. The triple represents the relationship between the head and tail entities, meaning that "library is a part of house" in this example.

4.2.6 Overall Model Structure

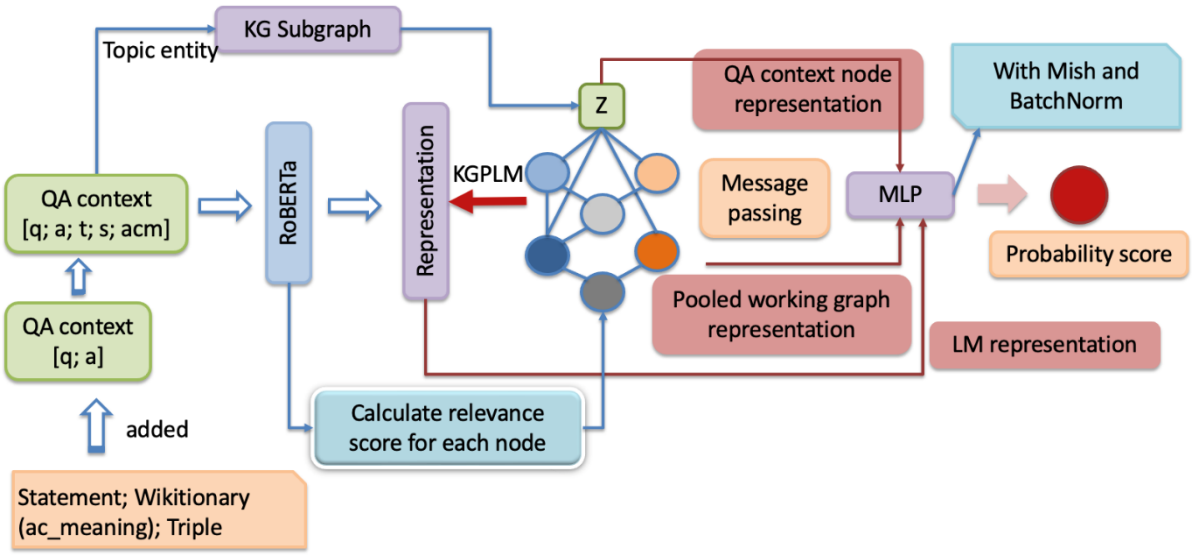

Figure 7: The architecture of the overall model.

As mentioned above, we try a number of methods to improve our model. Several attempts have improved, while some have not. The details will be discussed in the next chapter. Figure 7 is our final best model structure which has the highest accuracy compared to all the proposed methods we tried.

For our final model, the new QA contexts are constructed, which contains question, answer choice, triples, statement and answer choice meaning. The knowledge subgraph is retrieved from the ConceptNet based on the topic entities from the QA context. To encode the QA context, the result of RoBERTa-BiLSTM model is not good as we expected. The performance of RoBERTa is better than RoBERTa-BiLSTM's after we have more additional information. Thus, we use only RoBERTa model in our final model. In addition, we do not apply the SAGPooling layer and Ghost Batch Normalization to our final model since the performance of adding them has not improved, it's gotten worse. Our final model aims to predict the final answer with the high accuracy.

5. Experiments

5.1 Datasets

5.1.1 CommonsenseQA

In our task, we use the Commonsense Question Answering Dataset which is a 5-way multiple choice QA task that requires reasoning with commonsense knowledge, containing 12,102 questions. This dataset was generated by Amazon Mechanical Turk workers. Table 1 shows the data statistics of commonsense QA dataset. Figure 8 [21] is an example of CommonsenseQA. One question has five choices with one correct answer.

Training Set	Dev Set	Test Set
9741	1221	1140

Table 1: CommonsenseQA data statistics.

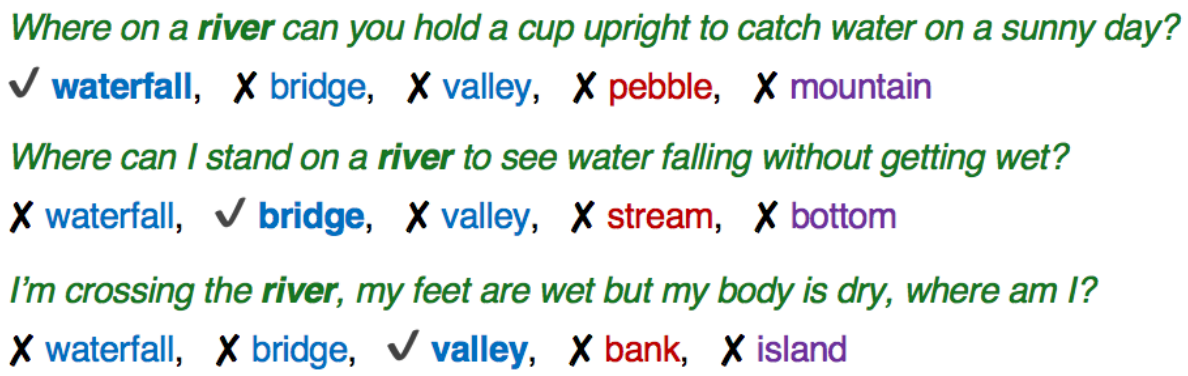

Figure 8 [21]: An example of the CommonsenseQA.

Since we have augmented each sample with additional information, our QA context is resampled, for instance, Figure 9:

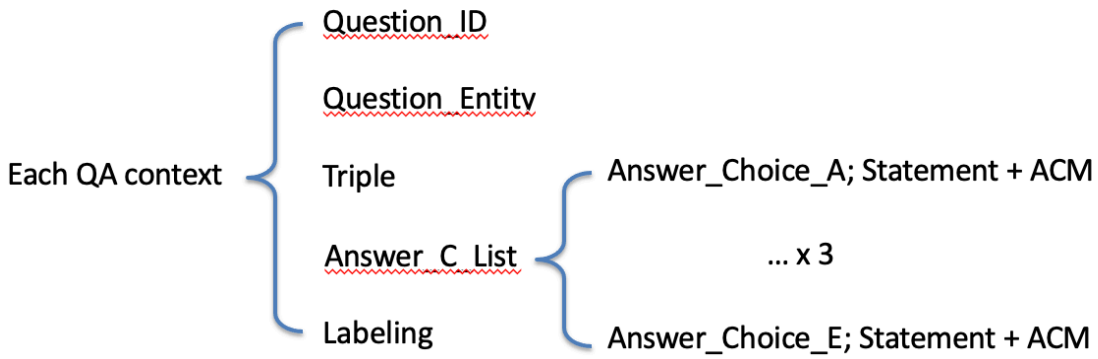

Figure 9: Updated each QA context.

Each QA context consists of a question ID, question entity, triple relation, answer choice list which contains the answer choices, a statement and the answer choice meaning for each choice, and the label.

5.1.2 ConceptNet

For the knowledge graph part, we use ConceptNet, a general-domain knowledge graph, as our structured knowledge source. ConceptNet has 799,273 nodes and 2,487,810 edges in total. Node embeddings are initialized using the entity embeddings which are obtained by applying pretrained language models to all triples in ConceptNet and then obtains a pooled representation for each entity. Figure 10 [9] is a sample of commonsense knowledge in ConceptNet.

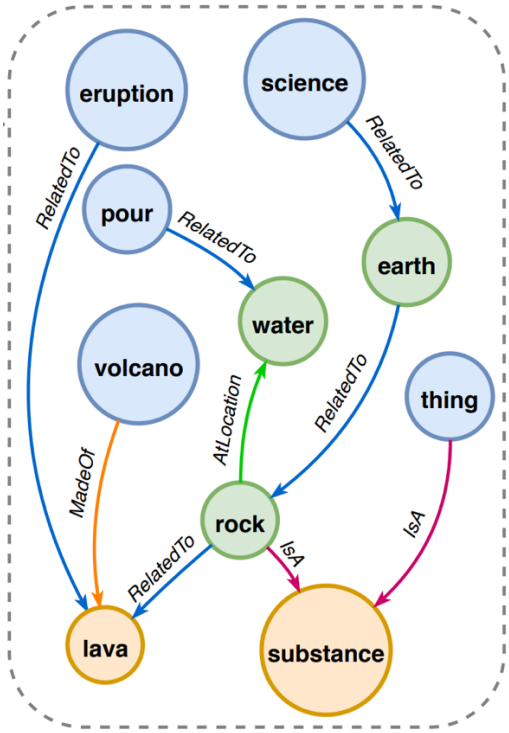

Figure 10 [9]: A sample of commonsense knowledge in ConceptNet.

5.2 Experiment Settings

In our experiments, we use both RoBERTa-base and RoBERTa-large pretrained language models. The RoBERTa-base is performed on NVIDIA GeForce RTX 2080 Ti, 11GB. The RoBERTa-large is performed on NVIDIA RTX A6000, 48GB. We also set the batch size is 64, the number of epochs is 20 and dropout rate is 0.2 when we train our model. The max sequence length is updated to 256 due to additional information. The learning rates for the encoder and decoder are 0.00001 and 0.001 respectively. Also, we use cross entropy loss as the loss function to optimize our model, and use RAdam as the optimizer. For the knowledge graph, we let the maximum node number of the working graph be 200 when we perform reasoning. The dimension of GNN is 200, and there are 5 layers in the knowledge graph. We retrieve 5-hop knowledge subgraphs. Also, the number of relation edges can be up to 38. Table 2 shows the experiment parameters.

Parameters	Values
Batch Size	64
Number of Epoch	20
Learning Rate for Encoder	0.00001
Learning Rate for Decoder	0.001
Loss	Cross Entropy Loss
Optim	RAdam
Maximum Node Number	200
Maximum Sequence Length	256
Dropout	0.2

Table 2: Experiment Parameters.

5.3 Experiment Results

In our model, we use two pretrained language model to test our methods in order to check our results with different language models, therefore, we have two result tables here for RoBERTa-base and RoBERTA-large respectively.

We first add a BiLSTM [17] after receiving the encoder representation of the RoBERTa pretrained language model in order to get more contextual representation. Then, we modify our MLP [7] for getting higher accuracy of predicting the final answer. Next, we apply a SAGPooling [11] to our ConceptNet knowledge graph for better pruning the irrelevant paths or nodes. Then, we merge the information from KG with the information of PLM so that the PLM could have information of the KG before it goes into the MLP. Finally, we add the triples, statement and Wikitionary as our additional information or features [25]. These attempted methods are grouped as (1), (2), (3), (4) and (5) shown on the following tables. For example, (1) indicates our proposed method 1, and (2) indicates our proposed method 2 which contains several attempts with different activation functions and the ghost batch normalization.

Methods	Epochs	Dev-accuracy	Test-accuracy	Rate of Change(%)
RoBERTa+GNN(Baseline)	14	0.6740	0.6374	--
RoBERTa+BiLSTM+GNN (1)	14	0.6773	0.6392	+0.18
RoBERTa+BiLSTM+GNN w/ LeakyReLU (2)	14	0.6855	0.6430	+0.56
RoBERTa+BiLSTM+GNN w/ Mish (2)	14	0.6896	0.6438	+0.64
RoBERTa+BiLSTM+GNN w/ Mish+GBN (2)	14	0.6175	0.5802	-5.72
RoBERTa+BiLSTM+GNN w/ LeakyReLU+GBN (2)	13	0.6601	0.6068	-3.06
RoBERTa+BiLSTM+GNN w/ Mish+SAGPool (3)	13	0.6626	0.6277	-0.97
RoBERTa+GNN w/ Mish+KGPLM (4)	12	0.6846	0.6559	+1.85
RoBERTa+GNN w/ Mish+KGPLM+Triples+Statement+ACM (5)	18	0.6699	0.6608	+2.34

Table 3: Results on RoBERTa-base model.

Methods	Epochs	Dev-accuracy	Test-accuracy	Rate of Change(%)
RoBERTa+GNN(Baseline)	13	0.7420	0.7284	--
RoBERTa+BiLSTM+GNN (1)	12	0.7559	0.7333	+0.49
RoBERTa+BiLSTM+GNN w/ LeakyReLU (2)	11	0.7584	0.7389	+1.05
RoBERTa+BiLSTM+GNN w/ Mish (2)	19	0.7723	0.7397	+1.13
RoBERTa+BiLSTM+GNN w/ Mish+GBN (2)	18	0.7296	0.7188	-0.96
RoBERTa+BiLSTM+GNN w/ LeakyReLU+GBN (2)	20	0.7138	0.7212	-0.72

RoBERTa+BiLSTM+GNN w/ Mish+SAGPool (3)	14	0.7392	0.7202	-0.82
RoBERTa+GNN w/ Mish+KGPLM (4)	14	0.7805	0.7389	+1.05
RoBERTa+GNN w/ Mish+KGPLM+Triples+Statement+ACM (5)	16	0.7674	0.7389	+1.05

Table 4: Results on RoBERTa-large model.

According to our results, we can see that the accuracies on both pretrained language models are slightly raised after adding BiLSTM. After modifying the MLP, the results are increased when we use the LeakyReLU and Mish activation functions instead of ReLU. Nevertheless, the accuracies go down on both models after we use Ghost Batch Normalization. Therefore, in this case, we decide to use the model, RoBERTa+BiLSTM+GNN w/ Mish. We then do the rest of experiments based on this model.

Then, we apply a SAGPooling in our new model, however, the results have gone down rather than up on both RoBERTa-base and RoBERTa-large. Thus, we cannot use this method. For the KGPLM method, the result has improved on RoBERTa-base. The performance is better on RoBERTa+GNN w/ Mish model, but it is not good on RoBERTa+BiLSTM+GNN w/ Mish model. In this case, we decide to use RoBERTa+GNN w/ Mish+KGPLM model without the BiLSTM. The accuracy does not seem to have changed with the KGPLM method on RoBERTa-large. In addition, after adding additional information, such as triples, statement, answer choice meaning, the performance of our model has improved on RoBERTa-base, but it also does not seem to have changed on RoBERTa-large. However, compared to the baseline model QAGNN, the results of using the KGPLM method and the method by adding additional information have improved on both RoBERTa-base and RoBERTa-large. The experiment details of these two main methods will be shown below.

In order to compare the benefits, we need to evaluate effectiveness of the methodologies we designed in this research. We have done several controlled experiments with the same model structures but different merging techniques and input tokens. The merging techniques have been implemented in the experiments are the Double Linear Product and Scaled Dot Product as we mentioned in Section 4.2.4. The input tokens are the combination in the following sets:

Set_1: {[Question Tokens] + [Answer Tokens] + [Wikitionary Tokens]};
Set 2: {[Question Tokens] + [Answer Tokens] + [Statement Tokens] + [Wikitionary Tokens]};
Set 3: {[Statement Tokens] + [Answer Tokens]};
Set 4: {[Question Tokens] + [Triple Relation Tokens] + [Statement Tokens] + [Wikitionary Tokens]}.

Part I. Revising Merging Functions

First to build a merging function with higher performance, we need to test the performance of different revising functions. In this case, this part of experiment is focusing on helping to choose revising methods for our merging function. As described in Chapter 3, the potential revising methods would be implemented in this experiment are Softmax and Sigmoid+Minmax. We test the model with the original input by using merging functions implemented with these two methods. The results are shown in Table 5 (based on merging function: Scaled Dot Pruduct, and on RoBERTa-base):

Methods	Dev_accuracy	Test_accuracy
Softmax	0.6689	0.6492
Sigmoid+Minmax	0.6731	0.6559

Table 5: Comparation of Softmax and Minmax based on Scaled Dot Product.

As Table 5 shows, the optimum revising method appears to be the Sigmoid+Minmax method, with improvement of 0.67% compared with the Softmax function. Based on the result, the Sigmoid+Minmax function is chosen for the main revising method for our merging function. Then, we can move on to the next part.

Part II. Final Accuracy Experiment

To improve the research efficiency, we would firstly test and pick the merging techniques from the two designed methods, then testing the performance by using different tokens to RoBERTa. The flowchart is shown in Figure 8:

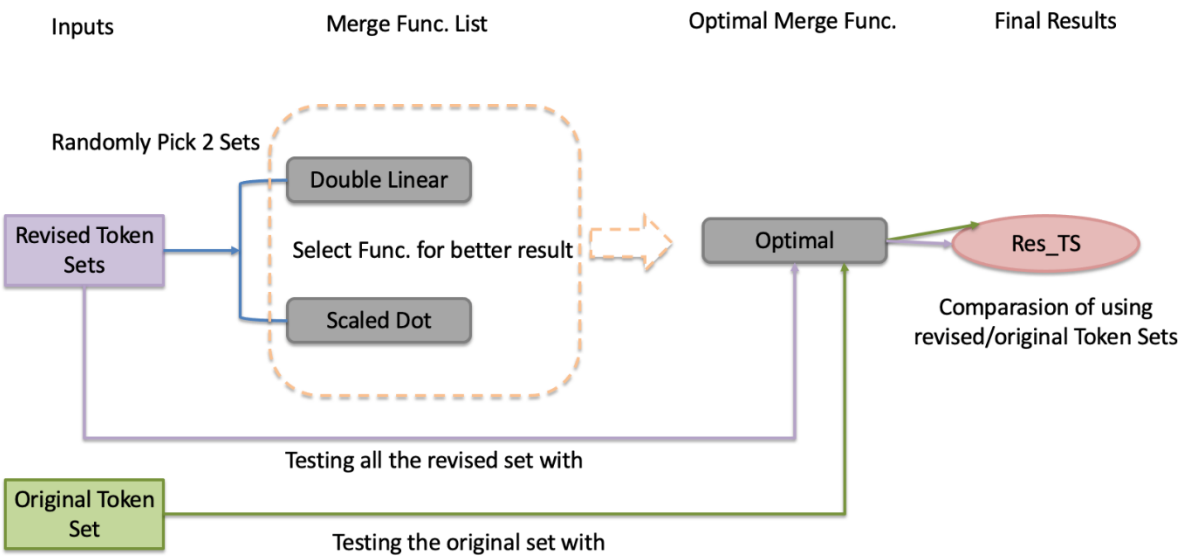

Figure 11: The flowchart of testing methods with different token sets.

We firstly randomly select two input sets and test with the two merging functions. Then, by comparing the results, we obtain the optimal merging function to test with all the input sets, and we apply the original input set to obtain the results of different token sets (Res_TS shown on Figure 11) to check the improvement given by using different input tokens and the optimal merging function.

The function with optimum performance in our experiment is the Scaled Dot Product method, as shown in the results in Table 6, on RoBERTa-base:

Methods	Dev_accuracy	Test_accuracy
Double Linear	0.6446	0.6374
Scaled Dot Product	0.6577	0.6438

Table 6: The results of testing merge functions.

Based on the result, the rest of the experiments are processed with this method. Firstly, we want to check how much improvement can be provided by adding merging functions. So, we test our final merged model with the original input directly. The result is clearly shown in Table 3, as

the optimal result for this KGPLM model is increased by 1.85% from the baseline to 65.59%. This result reflects the success of our approach. Then we can move onto the next step: testing updated input tokens.

Through Table 7, the result is clear that by using *Set 4: {[Question Tokens] + [Triple Relation Tokens] + [Statement Tokens] + [Wikitionary Tokens]}* with Scaled Dot Product and Sigmoid+Minmax can provide an optimal result in our research with average accuracy of 66.08%. This result has improved by 2.34% from the baseline model QAGNN, and also shows the efficiency of the method we attempted for this task. By using triple tokens replacing the position of answer tokens, the improvement of 2.1% is achieved from the results of Set 2 and Set 4, that may imply that triple tokens provide higher confidence for helping our model to catch the features of correct answers. By using the statement tokens, its accuracy becomes higher, as shown by comparing the results between Set 1 and Set 2. With the help of statement tokens, triple tokens and Wikitionary tokens we obtained the best result (Set 4) for our research.

Methods	Dev_accuracy	Test_accuracy
Set 1	0.6724	0.6334
Set 2	0.6658	0.6398
Set 3	0.6355	0.6261
Set 4	0.6675	0.6608

Table 7: The results for different input sets.

5.4 Experiment Analysis

First of all, the performance of the RoBERTa + BiLSTM model is better than using only one pretrained language model because we can obtain more contextual representations after the language model. For the modified MLP with different activation functions, the accuracy has improved, that is because the LeakyReLU is a modified one based on the ReLU and the Mish function obtained the lowest loss and Mish either matches or improves neural network architectures' performance compared to using ReLU activation functions. However, the accuracy decreases when we use Ghost batch normalization instead of batch normalization. This may be because Ghost batch normalization allows the use of large batch sizes, but the batch normalization

parameters are calculated on smaller sub-batches. Therefore, its performance may be not good on this QA task.

For the self-attention graph pooling method, its performance goes down, and is lower than the baseline's, possibly because of the duplication of the pooling strategies. In the original model QAGNN, it already has a pooling method to prune the paths conditioned on the relevance score between each node in the KG and topic entities in the QA context. By using SAGPooling, it may be also because some related paths or nodes are incorrectly defined as irrelevant. Such information may be missed on the knowledge graph during message passing. If one of pooling methods misses some related paths or nodes, downstream units may not realize such information, and the effectiveness could be weaker. Thus, its performance could not be good as we expected.

For the methods of KGPLM and additional information, the performance on RoBERTa-base and RoBERTa-large has increased compared to the baseline model QAGNN. When we add the information of the KG to the PLM, it can help the nodes in the PLM to gain information about adjacency relations in the KG. It also helps the PLM to more easily obtain features between the choices of each QA input and improve the precisions. Thus, the performance of this method has increased.

Adding triples, statement and Wiktionary as segment tokens to PLM improves the testing accuracy as well. The reason for that would be the information in the original QA input is too less. The only difference would be the answer tokens, which may not be reliable enough for the PLM to distinguish the correct answer from all the choices. However, triples, statement and Wiktionary can be regarded as reasonable supplement information for input tokens. The QA input form would be set up by more segments with clear relations (triples) and extra evidences (statement and Wikitionary) to support the PLM catch the possibilities for answer choices. Therefore, its performance has improved as well.

Several methods shown results in the RoBERTa-base and RoBERTa-large are different in trends. That is, the performances of several methods are good on RoBERTa-base, but they are not good enough as we expected on RoBERTa-large. In this case, we have no idea why this is happening so far.

6. Conclusion & Future Work

In conclusion, for this QA task, our goal is to predict the final answer from commonsense question answering with five choices using pretrained language model (PLM), knowledge graph (KG) and additional information. For achieving higher performance of our model, we have attempted several methods, for example, adding BiLSTM, modifying MLP, trying SAGPooling, applying KGPLM and adding additional information to the input. Based on the baseline model QAGNN, we added one more BiLSTM layer after the pretrained language model in order to obtain more contextual representations, and we modified the MLP to optimize our model. We try adding a SAGPooling to save the related paths and prune the irrelevant nodes from the retrieved KG subgraph. We also merge the information of the KG to the PLM. Finally, we add more information to the original input, consisting of triple relation, statement, answer choice meaning (from the Wikitionary). The results of these methods have both improved and decreased on RoBERTa-base and RoBERTa-large. In our experiments, we first test our methods with RoBERTa-base. We tried our experiments many times on RoBERTa-base, and when the results are good enough, we then test it again on RoBERTa-large. However, when we obtained the best results on RoBERTa-base, the results of some methods are not good as we expected on RoBERTa-large.

In our model, there still exist many noisy nodes or paths in the working graph when we retrieve the subgraph from the large ConceptNet knowledge graph. Also, the information of triple relation, statement and answer choice meaning may not be enough to help the model predicting the final answer with such additional information. Therefore, in the future work, we can add more information, such as the meaning of the question. Also, we can come up a new method to prune the noisy nodes or paths in the knowledge graph part. Since the results of several methods are not good enough on RoBERTa-large, we should find out the problem and solve it.

Acknowledgement

During these semesters, a lot of people helped me academically. Here, I would like to thank all the people who helped me, especially my professor, IWAIHARA, Mizuho. When I was confused about my research direction, it was the professor who gave me advice. The professor gave me a lot of guidance when I didn't know how to improve my modeling methods. Thus, I am here to thank the professor IWAIHARA for his help.

In the process of learning, I always study in the research room with my seniors and classmates. Sometimes we could have academic discussions, which could really help me expand my knowledge and be of great help to my research. Therefore, I would like to thank them here for their help and company.

References

[1] Aissi, H., Bazgan, C., & Vanderpooten, D. (2009). Min–max and min–max regret versions of combinatorial optimization problems: A survey. European journal of operational research, 197(2), 427-438.

[2] Banerjee, P., Pal, K. K., Mitra, A., & Baral, C. (2019). Careful selection of knowledge to solve open book question answering. arXiv preprint arXiv:1907.10738.

[3] Bao, J., Duan, N., Yan, Z., Zhou, M., & Zhao, T. (2016, December). Constraint-based question answering with knowledge graph. In Proceedings of COLING 2016, the 26th international conference on computational linguistics: technical papers (pp. 2503-2514).

[4] Bauer, L., Wang, Y., & Bansal, M. (2018). Commonsense for generative multi-hop question answering tasks. arXiv preprint arXiv:1809.06309.

[5] Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934.

[6] Bosselut, A., Rashkin, H., Sap, M., Malaviya, C., Celikyilmaz, A., & Choi, Y. (2019). COMET: Commonsense transformers for automatic knowledge graph construction. arXiv preprint arXiv:1906.05317.

[7] Fiedler, J. (2021). Simple modifications to improve tabular neural networks. arXiv preprint arXiv:2108.03214.

[8] Hironsan. 2020. “Building an application of question answering system from scratch”. https://towardsdatascience.com/building-an-application-of-question-answering-system-from-scratch-2dfc53f760aa.

[9] Ji, H., Ke, P., Huang, S., Wei, F., Zhu, X., & Huang, M. (2020). Language generation with multi-hop reasoning on commonsense knowledge graph. arXiv preprint arXiv:2009.11692.

[10] Khashabi, D., Min, S., Khot, T., Sabharwal, A., Tafjord, O., Clark, P., & Hajishirzi, H. (2020). Unifiedqa: Crossing format boundaries with a single qa system. arXiv preprint arXiv:2005.00700.

[11] Lee, J., Lee, I., & Kang, J. (2019, May). Self-attention graph pooling. In International conference on machine learning (pp. 3734-3743). PMLR.

[12] Lin, J. (2020). Knowledge chosen by relations. https://github.com/jessionlin/csqa/blob/master/Model_details.md.

[13] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

[14] Mishra, P. (2020). Understanding T5 Model: Text to Text Transfer Transformer Model. https://towardsdatascience.com/understanding-t5-model-text-to-text-transfer-transformer-model-69ce4c165023

[15] Misra, D. (2019). Mish: A self-regularized non-monotonic activation function. arXiv preprint arXiv:1908.08681.

[16] Pan, L., Rashid, T., Peng, B., Huang, L., & Whiteson, S. (2021). Regularized softmax deep multi-agent q-learning. Advances in Neural Information Processing Systems, 34, 1365-1377.

[17] Pearce, K., Zhan, T., Komanduri, A., & Zhan, J. (2021). A comparative study of transformer-based language models on extractive question answering. arXiv preprint arXiv:2110.03142.

[18] Qiu, X. (2020). Neural Networks and Deep Learning. China Machine Press. https://nndl.github.io/, 2020.

[19] Siami-Namini, S., Tavakoli, N., & Namin, A. S. (2019, December). The performance of LSTM and BiLSTM in forecasting time series. In 2019 IEEE International Conference on Big Data (Big Data) (pp. 3285-3292). IEEE.

[20] Speer, R., Chin, J., & Havasi, C. (2017, February). Conceptnet 5.5: An open multilingual graph of general knowledge. In Thirty-first AAAI conference on artificial intelligence.

[21] Talmor, A., Herzig, J., Lourie, N., & Berant, J. (2018). Commonsenseqa: A question answering challenge targeting commonsense knowledge. arXiv preprint arXiv:1811.00937.

[22] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:1710.10903.

[23] Wikipedia contributors. (2022, December 16). Knowledge graph. In Wikipedia, The Free Encyclopedia. Retrieved 07:49, January 11, 2023, from https://en.wikipedia.org/w/index.php?title=Knowledge_graph&oldid=1127735659

[24] Wikipedia contributors. (2023, January 4). Wiktionary. In Wikipedia, The Free Encyclopedia. Retrieved 12:27, January 14, 2023, from https://en.wikipedia.org/w/index.php?title=Wiktionary&oldid=1131513200

[25] Xu, Y., Zhu, C., Xu, R., Liu, Y., Zeng, M., & Huang, X. (2020). Fusing Context Into Knowledge Graph for Commonsense Reasoning. ArXiv, abs/2012.04808.

[26] Yasunaga, M., Ren, H., Bosselut, A., Liang, P., & Leskovec, J. (2021). QA-GNN: Reasoning with language models and knowledge graphs for question answering. arXiv preprint arXiv:2104.06378.