Learning Knowledge by Using Sentences with Relational Contexts

Ryo Kamoi and **Jiehao Liao**
Department of Computer Science
The University of Texas at Austin
{ryokamoi,jhliao}@utexas.edu

1 Introduction

Answering questions that requires knowledge or common sense is still currently a challenging task for neural language models. Recent works (e.g. Bauer et al., 2018; Lin et al., 2019; Ding et al., 2019; Wang et al., 2020) popularize using Knowledge Graphs (KG) in models, which present knowledge to models through graph structures such as paths. However, we observe that these models use KGs in a complicated manner, and such complications may impact performance. In addition, these strategies have the disadvantage of not allowing the recently proposed transfer learning using multiple QA datasets in the same format (Khashabi et al., 2020). Instead, we propose to construct sentences from graphs to generate sentence-based representations of knowledge for models to learn, which may help models to better assimilate knowledge from sentences for answering common sense questions. We will evaluate our idea on the CommonsenseQA (Talmor et al., 2019) dataset.

Our experiments show that the proposed method does not introduce major improvement compared to existing methods providing knowledge graph to QA models. However, we demonstrate some novel observations. First, finetuning based method may be sensitive to hyperparameters, and it is difficult to reproduce the results on the leaderboard. Second, the proposed method improves performance on training data significantly, but does not on test data. Third, better sentence generation strategies may improve performance. We are looking forward to study this approach more to achieve better experimental performance.

1.1 Motivation

There are two observations that motivate this work. First, prior work providing information in KG to QA models dose not seem to be successful. Second, recent work shows that QA models are able to learn linguistic reasoning abilities that generalize across formats, and training QA models on multiple QA datasets improve performance.

Existing QA Models Using KG Models that use KGs may be unnecessarily complicated for reasoning. For example, in KagNet (Lin et al., 2019), a generated KG is encoded by a combination of Graph Convolutional Networks, LSTMs, and the Attention Mechanism into a vector representation. Because there are multiple steps for encoding and weighting paths in a KG, we believe KagNet may find it difficult to identify relationships between concepts through the vector representation alone. In contrast, on the CommonsenseQA leaderboard,[1] the BERT + OMCS method fine-tuned a BERT-large model on extracted sentences of the Open Mind Common Sense (OMCS) Database (Speer et al., 2016). Then, the model was fine-tuned to the CommonsenseQA dataset (Talmor et al., 2019) for the task, and it outperformed KagNet by 3.6% in terms of accuracy. This indicate that it may be better for a Transformer to learn knowledge through fine-tuning instead of identifying knowledge from representations of KGs. The result motivate our hypothesis that graph-based methods using KGs are difficult, and sentence-based knowledge learning methods may be better. Our goal is to show that this is indeed the case with empirical evidence.

Transfer Learning Recent progress in QA models by using transfer learning motivate our approach. Transfer among various NLP tasks (McCann et al., 2018; Raffel et al., 2020) or QA datasets (Khashabi et al., 2020) have been recently studied. As a study specifically related to this project, UnifiedQA (Khashabi et al., 2020) takes an approach to pre-train a QA-model on multiple QA

[1] https://www.tau-nlp.org/csqa-leaderboard

datasets. The model is a big T5 model pre-trained on multiple datasets to assimilate more knowledge for knowledge-based question answering in general, and achieves state-of-the-art performance by simply finetuning the model on target datasets. This approach is applicable to any kinds of QA datasets containing only statements, but not to models using KG. To make use of recent progress in transfer learning of QA models, we propose to convert information in KG to sentences.

2 Background

2.1 Datasets

ConceptNet (Speer et al., 2016) is a knowledge graph that connects words and phrases of natural language with labeled, weighted edges. The graph is a version of the Open Mind Common Sense (OMCS) project (Singh, 2002), which is a common sense knowledge base of the most basic things a person knows. In the graph, concepts are nodes and relationships are edges. We define a relationship triplet in ConceptNet as (c_1, r, c_2), where c_1, c_2 are two distinct concept nodes and r is the relationship edge between them. We will use the graph to create a knowledge base of sentences formed from these relationship triplets in ConceptNet. In subsection 3, we will try out different sentence generation models that can embed knowledge in sentences for answering commonsense questions.

CommonsenseQA (Talmor et al., 2019) is a question answering dataset that requires commonsense knowledge to correctly identify relationships between concepts. Since the dataset is built upon ConceptNet (Speer et al., 2016), a model will be more accurate in answering questions in CommonsenseQA if it has knowledge from ConceptNet. In the dataset, each question have 5 answer choices. 3 of the answer choices come from direct relationships between a question concept and an answer concept in ConceptNet. The other 2 answer choices do not have a direct relationship in ConceptNet but are still related to the question. We plan to use this dataset to evaluate how well our model learned from sentences created from ConceptNet.

2.2 KagNet

KagNet (Lin et al., 2019) is one of the first models providing information in knowledge graph to transformer-based models. KagNet use ConceptNet to solve CommonsenseQA. The proposed method retrieve a schema graph, a sub-graph related to each question, and encode the schema graph with graph convolutional networks (GCN) and LSTM. Here, we describe a process to extract a schema graph from ConceptNet. First, it retrieves answer concepts C_a and question concepts C_q, which can be noisy. Then, we find paths between question and answer concepts shorter than k ($k = 4$ in the original setting). Finally, the method prune the schema graph by using scores calculated from knowledge graph embeddings.

3 Injecting Knowledge as Text

3.1 Building on KagNet

To make a direct comparison with KagNet (Lin et al., 2019), we did experiments using information gathered by KagNet. Specifically, we take paths generated by KagNet and generate sentences for each path.

KagNet scans the question and answers of CommonsenseQA for concept nodes in ConceptNet. We define question concept as a concept word in the question and answer concept as a concept word in the answer. For each question concept and answer concept, KagNet will find multiple directed paths from the question concept to the answer concept. Refer to Figure 1 for an example of a path.

Two concepts in a path are connected by a relationship triplet (c_1, r, c_2) from ConceptNet. We generate a sentence for each relationship triplet in the path, and then concatenate the sentences together along a path to give path context to the model.

3.2 Pseudo-Sentences (Templated)

For the simplest generation method, we used generation from templates from (Feldman et al., 2019). Since every relation in ConceptNet has a template sentence, we simply inserted the concept words into the template for the relation. An example can be found in Figure 2.

3.3 Perplexity Ranked Sentences

We also used the GPT-2 method from (Feldman et al., 2019) to generate better sentences.[2] Specifically, we generated multiple sentences for each relationship triplet by using multiple templates, adding articles and prepositions, and changing verbs into

[2] https://github.com/JoshFeldman95/Extracting-CK-from-Large-LM

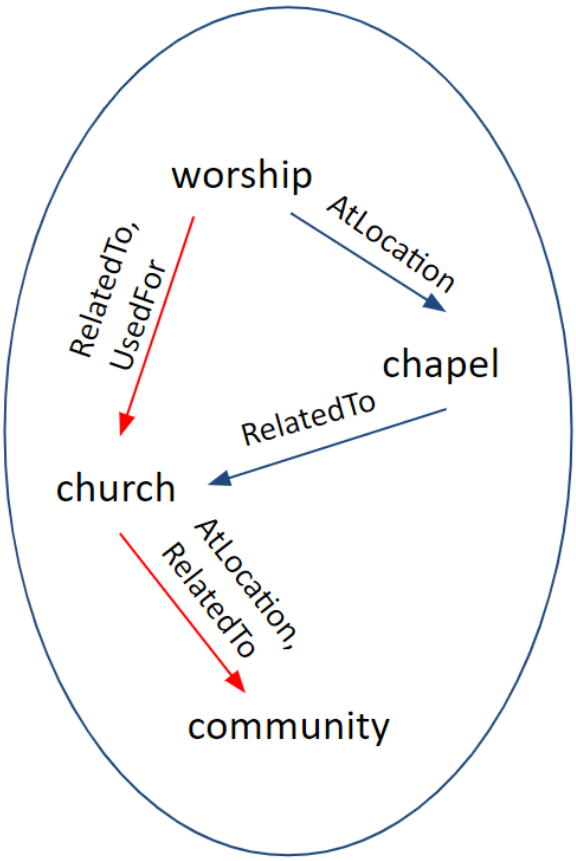

Figure 1: An example of paths generated by KagNet. One such path is worship → chapel → church → community, with (worship, AtLocation, chapel), (chapel, RelatedTo, church), (church, RelatedTo, community) as the relationship triplets along the path. The red arrows indicate that there can be more than one relation between two nodes.

Relationship Triplet: (birth, Antonym, death)
Template: {0} is the opposite of {1}
Sentence: birth is the opposite of death

Figure 2: Here's an example of templated sentence generation. We put the first concept in {0} and second concept in {1}.

Relationship Triplet: (songs, PartOf, album)
Templates: {1} has {0}, {0} is part of {1}, {0} is a part of {1}
Top 3 Sentences: (the album has songs, -18.48), (songs is part of the album, -19.0), (the album has the songs, -19.46)

Figure 3: Here's an example of GPT-2 evaluated sentence generation. In each tuple, the left is the sentence and the right is the perplexity score of the sentence from GPT-2. After putting the first concept in {0} and second concept in {0}, we generate more sentences by adding articles such as 'the' and 'a' in front of the concepts. The sentence with the highest perplexity score will be used.

gerunds. Then, we pass these sentences to a pre-trained GPT-2 to get a perplexity score for each. The sentence with the highest perplexity score will be used for context. An example of this generation can be found in Figure 3.

3.4 Document Per Question

For each question-answer pair in CommonsenseQA, KagNet generates a ConceptNet subgraph from multiple paths using multiple question concepts and multiple answer concepts. We concatenate multiple sentences together along a path into one sentence, then concatenate these sentence per path together into one big paragraph for each question-answer pair. Since there are 5 question-answer pairs per question in CommonsenseQA, we concatenate these 5 paragraphs together to form a document of knowledge.

To reduce the amount of information put into a document, we also filtered out unnecessary knowledge paths using KagNet's path scoring method (Lin et al., 2019). Specifically, we kept a certain number of paths for each question-answer concept pairs and removed paths with scores that did not meet a certain threshold.

4 Experimental Setup

4.1 Fine-tuning Framework

We are currently using KagNet (Lin et al., 2019) as a baseline in comparison to the BERT + OMCS model. Both model uses BERT-large (Devlin et al., 2019) for encoding question-answer pairs. However, knowledge is presented to each model in a different manner. KagNet encodes a KG using combinations of different neural models. BERT + OMCS fine-tunes BERT-large to learn knowl-

edge from sentences directly into the encoder. This allows us to measure which method is better for understanding common sense while keeping the question-answer context constant.

4.2 Classifier Framework

Proof-of-Concept Experiment Instead of fine-tuning a LM directly such that it can store knowledge, we can take a retrieval approach. First, we will concatenate a question-answer pair with the question-answer concept sentences generated in subsection 3. The concatenation can be used with BERT and a classifier on the $[CLS]$ token to check if the question-answer pair is most probable. Namely, a sequence of token in a style of $\{\,[CLS]\,C\,[SEP]\,Q\,a_i\,\}$ is passed to the models such as BERT where C is a context (pseudo-sentence) for each question, Q is a question, and $a_i \in \{a_1, \ldots, a_n\}$ is an answer choice. This method is based on an implementation of reading comprehension task proposed in the BERT paper (Devlin et al., 2019) and a model in Mitra et al. (2019). We use the final hidden vector $h_i \in R^H$ corresponding to the first input token ($[CLS]$) for each answer choice, and feed them into the linear classifier $W \in R^{n \times H}$. We maximize the log probability of the correct label during fine-tuning.

Multiple Choice Framework We converted CommonsenseQA questions with the KagNet documents created in section 3.4 into RACE Dataset form (Lai et al., 2017). Each question is paired with a document and 5 answer choices. We fine-tune pre-trained BERT-Large to answer each question.

For BERT, we give input examples in the form of "[CLS] document tokens [SEP] question tokens ? answer tokens [SEP]". An example of this can be found at Figure 4.

The document tokens are truncated such that the only 320 tokens are given to BERT for context. We pair each answer choice with each document-question pair such that there are 5 input examples per question, and extract the BERT output at the [CLS] token for each input to get 5 logit outputs per question. Then, we take the softmax of these 5 logit outputs to get a probability for each answer choice as the correct label. This is defined as the following,

$$P(ans_i) = \frac{e^{logit_{i[CLS]}}}{\sum_{j=1}^{5} e^{logit_{j[CLS]}}} \quad (1)$$

where $logit_{i[CLS]}$ correspond to the logit output at the [CLS] token for answer choice i, and $P(ans_i)$ is the probability of answer i being the correct label. During fine-tuning, the log-probability of the correct label is maximized. The argmax of these probabilities is the answer.

Document: Death is created by a cyanide. The poison causes death, the death is related to poison; the poison is related to cyanide, the poison is related to cyanide.is the opposite of death, living is like death; love causes live, love wants to live. Death can last forever; love can last forever.
Question: When can a person's death not be wanted by the person?
Answers: cyanide, love, burial, suicide, murder
BERT Input Example (for murder): `[CLS] death is created by a cyanide . the poison causes death, ... ; love can last forever . [SEP] when can a person 's death not be wanted by the person ? murder [SEP]`

Figure 4: A BERT input example when given the document, question, and an answer. There are tokens in the omission in the input example, but not shown to save space for this report.

4.3 Chunking Framework

Usually, if we used only 320 tokens per input example, a majority of the document tokens are truncated, which means a lot of necessary information to answer the question are lost from truncation. We propose to alleviate this issue by chunking a document into multiple documents for input examples.

To keep information per path intact, we chunk by appending tokens of a KagNet path into an input example as long as the input example do not overflow 320 tokens. Then, we do the same procedure in the Classifier Framework, section 4.2, per chunked document. This generates 5 input examples for the 5 answer choices for each chunked document-question pair.

The outputs of the chunked documents are aggregated together for selecting the answer choice. Specifically, for each chunked document-question pair, we generate 5 probabilities for the 5 answer choices from the softmax of 5 BERT [CLS] logit outputs. We take the mean of the 5 probabilities across chunked document-question pairs, defined as the following,

$$P(ans_i) = \frac{1}{M} \sum_{j=1}^{M} P(ans_i \mid doc_j) \quad (2)$$

Where $P(ans_i \mid doc_j)$ is the probability of answer i being the correct choice when given the chunked document j as context and M is the number of chunked documents. $P(ans_i)$ is the mean probability across chunked documents for answer i.

For fine-tuning, we maximize the log-probability of the correct label for each chunked document. We take the argmax of the 5 mean probabilities for the answer.

4.4 Implementation

Fine-Tuning Framework For preliminary experiments, which we report in section 5.1, we are using the training code from the Multi-Hop Graph Relation Networks (Feng et al., 2020) GitHub repository.[3] Since there are pre-defined training code, this is easier for fast experiments.

Proof-of-Concept Experiment For the proof-of-concept experiment which report in subsection 5.2, we used code for a BERT model on the RACE dataset from a public repository.[4]

Multiple Choice Framework We modified the code from the `huggingface` repository for multiple choice QA task[5] such that it can fine-tune a BERT model to answer our CommonsenseQA questions with context in RACE Dataset form. The models are coded in Python using the PyTorch library (Paszke et al., 2019). We train each model using 4 parallel NVidia 1080-TI GPUs on the Maverick2 machines at the TACC Computing Center.[6] Training with truncated documents usually take up to 24 hours. Training with multiple chunked documents increases the training time to up to 48 hours.

4.5 Evaluation

Accuracy We evaluated how accurately our model answered questions from the CommonsenseQA dataset. This is scored by dividing the number of correctly answered questions by the total number of questions.

Human Evaluation of Sentences We also evaluated a sample of the generated sentences from section 3 on the Likert scale from 1 to 5 under the following criterion:

1. **Coherency**: Is the sentence grammatically correct and fluent?

2. **Factuality**: Does the sentence convey factual information?

We sampled 3 sentences of each generation type from the 17 usable relations in KagNet to get a total of 102 sentences. 3 raters were asked to rate these sentences on the above criterion. We take the mean of the 3 ratings for each criterion and for each sentence, and then take the mean and the standard deviation of each criteria for each generation type. The raters did not know if a sentence is from templated or GPT-2 evaluated so that the results are not biased. The evaluation is presented in section 6.1.

5 Results

5.1 Fine-Tuning Framework

We are trying to replicate the results of BERT + OMCS which motivates our research. The BERT + OMCS model on the leaderboard uses whole-word-masking scheme (Devlin et al., 2019) for fine-tuning, but we use a normal masking scheme here. This is because the current BERT-large model on the `huggingface` repository do not split words into subwords. We first fine-tune BERT-large on the OMCS Corpus, and then fine-tune the model on the CommonsenseQA training set. We evaluate the model on the CommonsenseQA test set. The model comes from the `MHGRN` repository, as indicated in section 4.4.

Table 1 shows results for different number of training steps and learning rate on fine-tuning on OMCS corpus. For all models, the training step and learning for fine-tuning after on the CommonsenseQA dataset are the same. The best performance of our model (59.2) is worse than BERT + OMCS on the leaderboard (62.5), but it is still better than BERT-large (56.7) and the KagNet (58.9) on the leaderboard. We observe that the model performance is sensitive to hyperparameters, but this still suggest that providing knowledge in a form of sentences is effective.

[3] https://github.com/INK-USC/MHGRN

[4] https://github.com/NoviScl/BERT-RACE

[5] https://github.com/huggingface/transformers/tree/master/examples/multiple-choice

[6] https://portal.tacc.utexas.edu/user-guides/maverick2

step \learning rate	3e-5	6e-6	3e-6	3e-7
1e+5	44.40	55.84	58.34	55.12
2e+5	47.78	57.61	57.05	55.92
3e+5	48.91	**58.74**	57.37	54.55
4e+5			57.13	
5e+5			**59.23**	
6e+5			57.53	

Table 1: Performance of BERT + OMCS model on CommonsenseQA. We show result for different number of training step and learning rate for finetuning on OMCS corpus. We use identical settings for finetuning on CommonsenseQA.

Batch Size	No Context	Templated	GPT-2
32			59.79
64	**60.65±0.94**	58.19±1.68	57.93±1.86
128			47.34
256			47.83

Table 2: Results for classifier framework without any chunking. Using no context is better using truncated contexts.

Finally, we finetune BERT by using pseudo sentences generated by all triplets in ConceptNet. However, the pretrained model only achieves 57.70 in the current setting. Further modification would be needed to improve performance by using pseudo sentences.

5.2 Classifier Framework

Proof-of-Concept Experiment We evaluate a simple model for a proof of concept. We extracted the paths between the question concept and the answer concept in the same way as in KagNet using attention. We generate context (pseudo-sentences) by generating all triplets in the paths. To reduce the number of triplets, we removed paths of weights less than 0.5. Our model achieved 63.67 compared to the accuracy of the model without context, which was 60.31 on dev set. However, we observe that our method is sensitive to hyperparameters and initialization. We plan to analyze the current setting to stabilize and improve the performance.

Multiple Choice Framework For these results using the `huggingface` repository, if not indicated, we used a batch size of 64 and a learning rate of 1e-5.

We report the results of our multiple choice experiments without any chunking in Table 2. Compared to not using any context, using truncated context seems to distract the model from answering correctly.

We believe we can get a better representation

Context Type	Accuracy
No Context	60.65
GPT (Top 1)	60.1
GPT (Top 2)	60.28
Templated (Top 2)	60.28
GPT-2 (Top 3, Score ≥ 0.3)	**61.02**
GPT-2 (Top 5, Score ≥ 0.3)	59.68
Templated (Top 5, Score ≥ 0.3)	59.28
KagNet	**64.46**

Table 3: Results for classifier framework with chunking. Top # indicates using the top # of paths for every question-answer concept pairs. Score ≥ 0.3 indicates including any paths with score greater than or equal to 0.3.

Generation Type	C_{mean}	C_{std}	F_{mean}	F_{std}
Templated	4.065	0.959	4.327	0.862
GPT-2	**4.411**	**0.796**	**4.412**	**0.823**

Table 4: Human evaluation of different sentence generation. C_{mean} denotes the mean coherency, C_{std} denotes the standard deviation of coherency, F_{mean} denotes the mean factuality, F_{std} denotes the standard deviation of factuality.

of how well the model used the knowledge from context through the chunking framework, which are reported in Table 3. Our experiments found that using GPT-2 evaluated sentences do slightly better than using pseudo-sentences. Increasing or decreasing the information from using less or more paths per question-answer concept pair do not seem to drastically affect the accuracy. So, it is possible that the model is ignoring the context given.

6 Qualitative Analysis

6.1 Human Evaluation of Sentences

According to the evaluation results in Table 4, raters gave 0.346 more in coherency mean score for GPT-2 evaluated sentences compared to Templated sentences. While Templated sentences are already quite coherent with a mean score of 4.065, GPT-2 evaluated sentences are considered more coherent. There is a slight improvement in factuality in using GPT-2 sentences but not quite noticeable. We believe our statistics is reliable as the raters agreed often on the ratings as shown in Table 5.

6.2 Generation Highlights

Of the 51 sampled sentences, we present some generation highlights in Table 6. We discover that correcting the templated sentence grammar by adding articles, prepositions, and changing verbs into gerunds can improve sentence coherency, but

Generation Ratings Agreed	CO	FA
Templated (2 or more)	38	44
Templated (all 3)	17	22
GPT-2 (2 or more)	46	46
GPT-2 (all 3)	19	23

Table 5: Out of 51 sentences for each type of generation, raters agreed more often on ratings for GPT-2. 2 or more indicates that 2 or more out of the 3 ratings on a criteria are the same for a sentence. All 3 indicates that all 3 ratings on a criteria are the same for a sentence.

this method is still quite limited. A majority of the problems comes from needing to remove or add words in the concepts, which we do not touch upon in templates, and neither can GPT-2 do when it is only used for scoring perplexity. We believe using a true neural generative model with sampling such as nucleus sampling may be able to remove unnecessary or add unseen words to the sentences, but doing so requires gathering a good dataset of good sentences, which we did not have enough time for.

7 Related Work

7.1 QA with Knowledge Graph

Recent works using KG for QA often uses recently proposed neural method for graphs such as attention (e.g. Bauer et al., 2018; Lin et al., 2019) and graph convolution (e.g. Lin et al., 2019; Ding et al., 2019).

Specifically for CommonsenseQA, which is used in this project, models that use KG are quite popular. Wang et al. (2020) presents another work similar to KagNet, but differs in encoding a KG by using GPT-2. Lv et al. (2019) uses XLNet to provide reasoning on graphs. In general, a lot of work on CommonsenseQA differ in either the underlying model for encoding graphs or the way graphs are constructed.

7.2 Knowledge in Pre-trained Model

It has been observed that large deep learning models can internalize a sort of implicit knowledge after pre-training (Petroni et al., 2019; Kwon et al., 2019; Talmor et al., 2019; Feldman et al., 2019; Jiang et al., 2020; Roberts et al., 2020). However, recent research demonstrated that knowledge stored in pre-pretrained language models generalize better than supervised approaches (Feldman et al., 2019) and scales with model size (Roberts et al., 2020), but do not accurately understand the semantic meaning of relations (Kwon et al., 2019).

7.3 Exploiting Commonality across Multiple Tasks for QA

Transfer among various NLP tasks (McCann et al., 2018; Raffel et al., 2020) or QA datasets (Khashabi et al., 2020) have been recently studied. UnifiedQA (Khashabi et al., 2020) takes an approach to pre-train a QA-model on multiple QA datasets. The model is a big T5 model pre-trained on multiple datasets to assimilate more knowledge for knowledge-based question answering in general, and achieves state-of-the-art performance by simply finetuning the model on target datasets. This idea more closely resemble pre-training a Language Model for fine-tuning on downstream tasks. These studies motivate us to provide information in knowledge graph in a form of sentences.

7.4 Sentence Generation for Knowledge-Based Tasks

There are many tasks that also used sentence generation for knowledge-based tasks. Latcinnik and Berant (2020) generated the intermediate information used to answer a question, which improves the explainability of a language model in question answering. Feldman et al. (2019) trained a model on generated sentences from relationship triples in ConceptNet for commonsense knowledge base completion tasks. We take both of their approaches for sentence generation to improve upon models for commonsense question answering.

7.5 Data-to-Text

The proposed approach converting knowledge graph to sentences can be regarded as a variant of data-to-text, natural language generation from structured input. As a prior work most relevant to this papre, WebNLG (Gardent et al., 2017) is a set of triples extracted from DBpedia and a verbalisation of these triples. Our project differs from ordinary data-to-text in that supervised learning cannot be performed because there is no reference text. However, applying existing data-to-text methods to our model can be another approach of the future work.

8 Future Work

There are several studies we could not cover in this project.

Language Models We only evaluated BERT-large model, but more recent language models achieves higher performance on the leaderboard. A

Triplets	Generation
Concept 1: lawns **Concept 2**: need_cutting_in_winter **Relationship**: NotCapable	**Templated**: lawns is not capable of need cutting in winter **GPT-2**: lawns do not need cutting in winter
Concept 1: beautiful_day **Concept 2**: take_car_for_drive **Relationship**: Causes	**Templated**: sometimes beautiful day causes take car for drive **GPT-2**: sometimes a beautiful day causes you to take a car for drive
Concept 1: people **Concept 2**: attack_what_hate **Relationship**: CapableOf	**Templated**: people can attack what hate **GPT-2**: people can attack what hate
Concept 1: mount_whitney **Concept 2**: united_states **Relationship**: AtLocation	**Templated**: you are likely to find mount whitney in united states **GPT-2**: the place you are likely to find mounting whitney is in the united states
Concept 1: releasing_energy **Concept 2**: may_jump **Relationship**: Causes	**Templated**: sometimes releasing energy causes may jump **GPT-2**: the effect of releasing energy is may jump

Table 6: We present few examples of generation here from triplets. In the first two examples, GPT-2 improves coherency by correcting the sentence grammar and using less words by using other templates. The next three examples present some limitations. Our generation is unable to remove some unnecessary words to make the sentences coherent and sensible as presented in the third and fifth examples. Occasionally, prefixes such as 'mount' and other words can be treated as a verb such that the generation will modify it into a gerund. GPT-2 is unable to fix this by purely scoring perplexity alone. We believe making a true generative model can make generation more robust to these mistakes.

promising direction of the future work is to evaluate this approach on recent language models such as RoBERTa and T5 (UnifiedQA).

Transfer Learning is one of the motivations of our approach, but we could not perform sufficient experiments on this direction. We assume that transfer learning by using multiple QA dataset help model when we provide information in knowledge graph in a form of sentences. We simply combined our CommonsenseQA dataset with context and RACE dataset to avoid overfitting, but we did not observe any improvement in this setting. Further study is needed to verify this approach.

Attention Output As suggested from the peer feedback for our presentation in CS 395T, we believe that taking the mean of the output probabilities from chunked documents weighed both unnecessary and necessary information equally. In the future, we can implement a limit on the number of chunked documents such that we can apply attention over them to see which chunked documents convey the most important information. This may help us improve model accuracy.

Better Generation As mentioned in section 6.2, we should look into using a true neural generative method for sentence generation. Doing so requires gathering a dataset of triplets mapped to good, coherent, and sensible sentences. We discover in our experiments that using GPT-2 evaluated sentences slightly improves our QA model, so this area can be a good direction for future research for better conveying information to improve QA.

9 Conclusion

In this work, we replaced KagNet paths using sentences with relational contexts and generated a text document of knowledge per question in CommonsenseQA. Our experiments indicate that such a method is not as effective in conveying information to BERT as KagNet's graph-based method can be. Using code from the official `huggingface` repository, we gain an accuracy of 61.02% at best in the official development set, which is far from KagNet's accuracy on the official development set, 64.46%. However, we found that using GPT-2 evaluated sentences introduced noises in sentences that can be slightly beneficial for learning knowledge compared to a fixed representation of knowledge from using templated sentences. We hope to experiment and improve this framework by representing knowledge in text instead of graphs for other models in the future.

Acknowledgments

We like to thank Professor Eunsol Choi[7] and Professor Jessy Li[8] for supervising this project for the CS 395T and LIN 393 courses at UT Austin. We also would like to thank Yasumasa Onoe[9] for his great insights and suggestions for improving our project and helping us in doing the human evaluation of sentences.

References

Lisa Bauer, Yicheng Wang, and Mohit Bansal. 2018. Commonsense for Generative Multi-Hop Question Answering Tasks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *North American Chapter of the ACL (NAACL)*.

Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. 2019. Cognitive Graph for Multi-Hop Reading Comprehension at Scale. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2694–2703. Association for Computational Linguistics.

Joshua Feldman, Joe Davison, and Alexander M. Rush. 2019. Commonsense Knowledge Mining from Pretrained Models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1178. Association for Computational Linguistics.

Yanlin Feng, Xinyue Chen, Bill Yuchen Lin, Peifeng Wang, Jun Yan, and Xiang Ren. 2020. Scalable Multi-Hop Relational Reasoning for Knowledge-Aware Question Answering. *arXiv preprint arXiv:2005.00646*.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. *International Natural Language Generation Conference (INLG)*, 298:124–133.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020. How Can We Know What Language Models Know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UnifiedQA: Crossing Format Boundaries With a Single QA System. *arXiv preprint arXiv:2005.00700*.

[7] https://www.cs.utexas.edu/~eunsol/

[8] https://jjessyli.github.io/

[9] https://www.cs.utexas.edu/~yasumasa/

Sunjae Kwon, Cheongwoong Kang, Jiyeon Han, and Jaesik Choi. 2019. Why Do Masked Neural Language Models Still Need Common Sense Knowledge? *arXiv preprint arXiv:1911.03024*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 785–794.

Veronica Latcinnik and Jonathan Berant. 2020. Explaining Question Answering Models through Text Generation. *arXiv preprint arXiv:2004.05569*.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. KagNet: Knowledge-Aware Graph Networks for Commonsense Reasoning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2829–2839.

Shangwen Lv, Daya Guo, Jingjing Xu, Duyu Tang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, and Songlin Hu. 2019. Graph-Based Reasoning over Heterogeneous External Knowledge for Commonsense Question Answering. *arXiv preprint arXiv:1909.05311*.

Bryan McCann, Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The Natural Language Decathlon: Multitask Learning as Question Answering. *arXiv preprint arXiv:1806.08730*.

Arindam Mitra, Pratyay Banerjee, Kuntal Pal, Swaroop Mishra, and Chitta Baral. 2019. How Additional Knowledge can Improve Natural Language Commonsense Question Answering? *arXiv preprint arXiv:1909.08855*.

Adam Paszke, Adam Lerer, Trevor Killeen, Luca Antiga, Edward Yang, Sam Gross, James Bradbury, Francisco Massa, and Benoit Steiner. 2019. PyTorch : An Imperative Style , High-Performance Deep Learning Library. (NeurIPS).

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. Language Models as Knowledge Bases? In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 2463–2473.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21:1–67.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How Much Knowledge Can You Pack Into the Parameters of a Language Model? *arXiv preprint arXiv:2002.08910*.

Push Singh. 2002. The Public Acquisition of Commonsense Knowledge. In *AAAI Spring Symposium: Acquiring (and Using) Linguistic (and World) Knowledge for Information Access*, pages 47–53.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2016. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. *arXiv preprint arXiv:1612.0397*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. COMMONSENSEQA: A Question Answering Challenge Targeting Commonsense Knowledge. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4149–4158.

Peifeng Wang, Nanyun Peng, Pedro Szekely, and Xiang Ren. 2020. Connecting the Dots: A Knowledgeable Path Generator for Commonsense Question Answering. *arXiv preprint arXiv:2005.00691*.

A Templates

For the simplest generation method, we used generation from templates from (Feldman et al., 2019). The templates used in our experiments are listed in Table 7.

Relation	Template(s)
relatedto	{0} is like {1} {1} is related to {0} {0} is related to {1}
externalurl	{0} is described at the following URL {1}
formof	{0} is a form of the word {1}
isa	{0} is {1} {0} is a type of {1} {0} are {1} {0} is a kind of {1} {0} is a {1}
notisa	{0} is not {1} {0} is not a type of {1} {0} are not {1} {0} is not a kind of {1} {0} is not a {1}
partof	{1} has {0} {0} is part of {1} {0} is a part of {1}
hasa	{0} has {1} {0} contains {1} {0} have {1}
usedfor	{0} is used for {1} {0} is for {1} You can use {0} to {1} You can use {0} for {1} {0} are used to {1} {0} is used to {1} {0} can be used to {1} {0} can be used for {1}
capableof	{0} can {1} {0} can be {1} An activity {0} can do is {1} {0} sometimes {1} {0} often {1}
atlocation	The place you are likely to find {0} is in {1} The place you are likely to find {0} at {1} Something you find on {1} is {0} Something you find in {1} is {0} Something you find at {1} is {0} Somewhere {0} can be is {1}
causes	Sometimes {0} causes {1} Something that might happen as a consequence of {0} is {1} Sometimes {0} causes you to {1} {0} causes {1} The effect of {0} is {1}
hassubevent	Something you might do while {0} is {1} One of the things you do when you {0} is {1} Something that might happen while {0} is {1} Something that might happen when you {0} is {1}

	One of the things you do when you {1} is {0} Something that might happen when you {1} is {0} You {1} when you {0}
hasfirstsubevent	the first thing you do when you {0} is {1}
haslastsubevent	the last thing you do when you {0} is {1}
hasprerequisite	something you need to do before you {0} is {1} If you want to {0} then you should {1} {0} requires {1}
hasproperty	{0} is {1} {0} are {1} {0} can be {1}
motivatedbygoal	You would {0} because you want to {1} You would {0} because you want {1} You would {0} because {1}
obstructedby	{0} can be prevented by {1}
desires	{0} wants {1} {0} wants to {1} {0} like to {1}
createdby	{0} is created by {1}
synonyms	{0} and {1} are have similar meanings {0} and {1} are similar
antonym	{0} is the opposite of {1}
distinctfrom	it cannot be both {0} and {1}
derivedfrom	the word {0} is derived from the word {1}
symbolof	{0} is a symbol of {1}
definedas	{0} is defined as {1} {0} is the {1}
entails	if {0} is happening, {1} is also happening
mannerof	{0} is a specific way of doing {1}
locatednear	{0} is located near {1}
dbpedia	{0} is conceptually related to {1}
hascontext	{0} is used in the context of {1}
similarto	{0} is similar to {1}
etymologicallyrelatedto	the word {0} and the word {1} have the same origin
etymologicallyderivedfrom	the word {0} comes from the word {1}
causesdesire	{0} makes people want {1} {0} would make you want to {1}
madeof	{0} is made of {1} {0} can be made of {1} {0} are made of {1}
receivesaction	{0} can be {1} {0} is something that you can {1} {1} can be done to {0}
instanceof	{0} is an example of {1}
notdesires	{0} does not want {1} {0} doesn't want to {1} {0} doesn't want {1}
notusedfor	{0} is not used for {1}
notcapableof	{0} is not capable of {1} {0} do not {1}

nothasproperty	{0} does not have the property of {1}
notmadeof	{0} is not made of {1}

Table 7: Templates used for pseudo sentence generation.