Solving Commonsense Question Answering via Search in Knowledge Bases
CPSC 532V 2023W2 Assignment #1

Juntai Cao[♠]
Student #: 50171404
jtcao7@cs.ubc.ca

Yilin Yang[♠]
Student #: 24754350
yangyl17@cs.ubc.ca

Yuwei Yin[♠]
Student #: 36211928
yuweiyin@cs.ubc.ca

Department of Computer Science, University of British Columbia
2366 Main Mall, Vancouver, BC V6T 1Z4, Canada

Abstract

In this paper, we report our solution to Assignment 1, which requires using a knowledge base for commonsense question-answering (QA) tasks. For each QA example, we first extract keywords from the question (appended by the context) and each choice. Then, we implement Dijkstra's algorithm to search for the best path from the source word (the keyword in a choice) to the target word (the keyword in the question). Due to the large scale and density of the knowledge graph, we apply a series of heuristic optimizations to make the search efficient. After the path searching and node matching process, we visualize the path in figures and present the edge information (node pairs with relations) in natural language. At last, we select a choice based on the path attributes and compute the accuracy score as the evaluation. As the experimental results demonstrate, although it is unsatisfactory to directly use the matched path to select a choice in QA tasks, the path can provide a commonsense reasoning process that serves as auxiliary information and explanation to decision-making. All the code, experimental results, and visualization figures are available on GitHub. [1]

1 Introduction

In this Section, we introduce the task in § 1.1, datasets in § 1.2, and the method overview in § 1.3.

1.1 Task

In this assignment, we are dealing with multi-choice commonsense question-answering (QA) tasks (Talmor et al., 2019), where the input is a question with context and a set of options. We are required to select a choice with the help of external knowledge bases (KBs), in particular, ConceptNet (Speer et al., 2017) [2]. The concept path from keywords in the question and those in the options is supposed to explicitly show the commonsense reasoning process and thus assist the decision-making.

1.2 Dataset

We are given 10 data examples from several commonsense QA datasets as follows:

- CommonSenseQA (Talmor et al., 2019)
- COPA (Roemmele et al., 2011)
- Social IQA (Sap et al., 2019)
- WinoGrande (Sakaguchi et al., 2021)
- RocStories (Mostafazadeh et al., 2016)
- MCTaco (Zhou et al., 2019)
- PIQA (Bisk et al., 2020)

A example is shown in the left part of Figure 1. We are asked to answer the question (with context) by choosing "(a)" or "(b)". Some questions do not have a context. Some questions have multiple correct answers.

1.3 Overview

The core idea of our solution is to select the most relevant choice to the question, where the relevance is calculated based on the reasoning paths in ConceptNet between keywords in the question and those in each choice.

As shown in the left part of Figure 1, for each example, we first extract keywords from the question and choices. For each choice in the example, we pair up each keyword of the choice with each question keyword. As the right part of Figure 1 demonstrates, the keyword pairs are denoted as dotted lines between the question keywords ("Q_kw") and choice keywords ("C_kw"). We ignore the red lines because the two words on the ends are the same.

For each keyword pair {source_word, target_word}, we search for valid paths from the

[♠] Authors contributed equally and listed alphabetically.
[1] https://github.com/YuweiYin/UBC_CPSC_532V
[2] https://conceptnet.io/

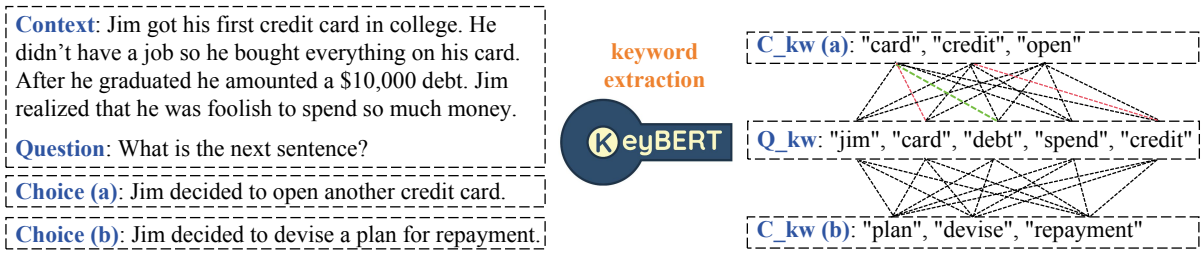

Figure 1: The overview of keyword extraction.

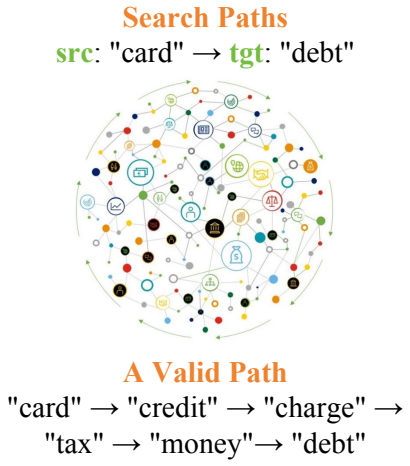

Figure 2: The illustration of path search in a knowledge base.

source word to the target word via the bidirectional weighted breadth-first search and Dijkstra's algorithm (Dijkstra, 1959) in the knowledge base. For instance, a valid reasoning path from "card" to "debt" is shown in Figure 2.

To enhance the efficiency of search in the extensive and dense knowledge graph, we implement a set of heuristic optimizations. Following the processes of path searching and node matching, we illustrate the identified paths through figures and transform the edge information into natural language sentences. At last, we make the option selection based on the attributes of the identified paths and calculate the accuracy score for evaluation.

2 Methodology

In this Section, we introduce the notation and formulation of our method.

2.1 Commonsense QA

Formally, we denote the test set as $\mathbf{E} = \{\mathbf{e_1}, \ldots, \mathbf{e_n}\}$ that has n different examples. The i-th example is $\mathbf{e_i} = \{cx_i, qu_i, \vec{\mathbf{c}}_i\}$, where cx_i is the i-th context text, qu_i is the i-th question text, and $\vec{\mathbf{c}}_i = \{c_i^1, \ldots, c_i^m\}$ are m choices (options). In practice, we concatenate each context cx_i and question qu_i into $q_i = cx_i + qu_i$. Thus, the example set is denotes as follows:

$$\mathbf{E} = \{\mathbf{e_1}, \ldots, \mathbf{e_n}\} \tag{2.1}$$

where a single example $\mathbf{e_i}$ is defined as follows:

$$\mathbf{e_i} = \{q_i, \vec{\mathbf{c}}_i\} \tag{2.2}$$

where a single choice $\vec{\mathbf{c}}_i$ is defined as follows:

$$\vec{\mathbf{c}}_i = \{c_i^1, \ldots, c_i^m\} \tag{2.3}$$

Each question q_i or choice c_i^j is a sentence that contains several words:

$$q_i = \{q_i^1, \ldots, q_i^Q\} \tag{2.4}$$

which means there are Q words in the original q_i sentence.

$$c_i^j = \{c_i^{j;1}, \ldots, c_i^{j;C}\} \tag{2.5}$$

which means there are C words in the original c_i^j sentence.

2.2 Keywords

After extracting keywords of the question and choices, we can denote the extracted keyword set from i-th question q_i and the corresponding j-th choice c_i^j as follows:

$$qkw_i = \{qkw_i^j\}_{j \in \Gamma_q} \subseteq q_i \tag{2.6}$$

where Γ_q is the indices of the keywords extracted from question q_i. There are $|\Gamma_q|$ keywords in the keyword set qkw_i.

$$ckw_i^j = \{ckw_i^{j;k}\}_{k \in \Gamma_c} \subseteq c_i^j \tag{2.7}$$

where Γ_c is the indices of the keywords extracted from choice c_i^j. There are $|\Gamma_c|$ keywords in the keyword set ckw_i^j.

2.3 Knowledge Base

The knowledge base is a graph $\mathcal{G}$ denoted as:

$$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\} \tag{2.8}$$

where $\mathcal{V}$ is the vertex set and $\mathcal{E}$ is the edge set in the graph $\mathcal{G}$.

Since we only focus on the word attribute of each vertex, the vertex set $\mathcal{V}$ can be denoted as:

$$\mathcal{V} = \{v_1, \ldots, v_N\} \tag{2.9}$$

where there are N words v_i in the graph.

For each edge, we focus on the following attributes: the relation, edge direction, edge weight, and two end vertices. The edge set $\mathcal{E}$ can be denoted as:

$$\mathcal{E} = \{\mu_i\}_{i=1}^{M} = \left\{\{r_i, d_i, w_i, v_i^1, v_i^2\}\right\}_{i=1}^{M} \tag{2.10}$$

where there are M edges in the graph. Each edge μ_i has a relation r_i, a direction d_i, a weight w_i, and two vertices v_i^1 and v_i^2. v_i^1 is the neighbor vertex of v_i^2, and vice versa.

2.4 Path Search

To evaluate the j-th choice c_i^j of the i-th example, we pair up all the keywords of this choice and those of the corresponding question q_i, i.e., ckw_i^j and qkw_i respectively. The paired keyword set κ can be denoted as follows:

$$\begin{aligned}\kappa &= ckw_i^j \times qkw_i \\ &= \left\{< ckw_i^{j;k_1}, qkw_i^{k_2} >\right\}_{k_1 \in \Gamma_c \wedge k_2 \in \Gamma_q}\end{aligned} \tag{2.11}$$

For each pair $< ckw_i^{j;k_1}, qkw_i^{k_2} >$, we apply the bidirectional breadth-first search (BFS) based on Dijkstra's algorithm (Dijkstra, 1959). The src-tgt process searches from the source vertex $ckw_i^{j;k_1}$ toward the target vertex $qkw_i^{k_2}$, and the tgt-src process goes in the opposite direction. Initially, the BFS queue of the src-tgt process is $\{v_{src}\}$ and that of the tgt-src process is $\{v_{tgt}\}$, where $v_{src} = ckw_i^{j;k_1}$ and $v_{tgt} = qkw_i^{k_2}$.

For each search step, both two processes move one depth forward. For a vertex v' in the current BFS queue, its linked edge set is $\mu' = \{\mu_i | v_i^1 = v \vee v_i^2 = v\}$, according to Equation 2.10. Then, the neighbor vertices will be put into the BFS queue as the starting points of the next BFS loop. Once the two BFS processes has a matched node in the middle, we find a valid path $p_i^{j;k_1;k_2}$ from the source vertex $ckw_i^{j;k_1}$ to the target vertex $qkw_i^{k_2}$.

Path Rebuilding When searching, we use a "visited set" to avoid putting visited vertices into the next BFS queue. There will not be a circle in the path, which means the search route is a tree-like pattern. We use a mapping dictionary to store the tree structure, making it easy to reversely rebuild the path from the matched vertex to the root vertex.

2.5 Path Selection

There might be multiple valid paths found in the same depth. Here, we choose the best path based on the path length and weights.

Formally, a path is denoted as follows:

$$P = \{\mathcal{V}_p, \mathcal{E}_p\} \tag{2.12}$$

where $\mathcal{V}_p$ is the vertex set in the path and $\mathcal{E}_p$ is the edge set. These two sets have the same definition as Equation 2.9 and Equation 2.10. Also, we have $|\mathcal{V}_p| = |\mathcal{E}_p| + 1$.

The length of the path is:

$$P_{len} = |\mathcal{E}_p| \tag{2.13}$$

The weight sum of the path is:

$$P_w = \sum_i w_i \tag{2.14}$$

where the i-th edge e_i in $\mathcal{E}_p$ has a weight of w_i.

Thus, the average weight of the path is:

$$\overline{P_w} = \frac{P_w}{P_{len}} = \frac{\sum_i w_i}{|\mathcal{E}_p|} \tag{2.15}$$

we choose one path with the largest average weight $\overline{P_w}$ to represent the best path of a search pair.

2.6 Choice Selection

For the j-th choice c_i^j of the i-th example, there are $\gamma = |\Gamma_c| \times |\Gamma_q|$ keyword pairs, as mentioned in Equation 2.11. Ideally, we will find γ paths and have a path result list as follows:

$$P_{res} = \{P_{len}^i, P_w^i\}_{i=1}^{\gamma} \tag{2.16}$$

Considering all edges in these paths, we calculate the score of choice S_i^j using the micro average:

$$S_i^j = \frac{\sum_{i=1}^{\gamma} P_w^i}{\sum_{i=1}^{\gamma} P_{len}^i} \tag{2.17}$$

Finding Ratio To be efficient in searching, we set a `max_depth` to limit the search depth of the BFS process. A `max_depth` of D means both src-tgt and tgt-src BFS process will run $D/2$ loops. However, setting a `max_depth` may result in not finding any valid path from the source word in the choice c_i^j to the target word in the question q_i within the `max_depth`. Therefore, we introduce a "Finding Ratio" η to adjust the score of choice c_i^j:

$$\eta = \frac{n_f}{\gamma} \tag{2.18}$$

where n_f is the number of found paths.

Using η, the score of choice S_i^j is:

$$S_i^j = \eta \frac{\sum_{i=1}^{n_f} P_w^i}{\sum_{i=1}^{n_f} P_{len}^i} \tag{2.19}$$

Source Word = Target Word The path length will be 0 if the source word from the choice c_i^j is the same as the target word from the question q_i. Taking this into account, we ignore such pairs in our calculation, which will affect both the denominator and numerator. The final score is:

$$S_i^j = \frac{n_f - n_s}{\gamma - n_s} \frac{\sum_{i=1}^{n_f - n_s} P_w^i}{\sum_{i=1}^{n_f - n_s} P_{len}^i} \tag{2.20}$$

where n_s is the number of pairs in which the source word and target word are the same.

Select a Choice After calculating the score S_i^j of each choice c_i^j in the example $\mathbf{e_i}$, we choice the α-th choice:

$$\alpha_i = \arg\max_j S_i^j \tag{2.21}$$

Given the answer in the example, we can evaluate the performance of our method on the multi-choice QA task using the accuracy metric.:

$$\alpha = \sum_i \mathbb{1}(\alpha_i = a_i) \tag{2.22}$$

where a_i is the correct answer (golden reference) presented in the i-th example $\mathbf{e_i}$, and $\mathbb{1}(\cdot)$ is the indicator function:

$$\mathbb{1}(x) = \begin{cases} 1 & \text{if x is true} \\ 0 & \text{otherwise} \end{cases} \tag{2.23}$$

3 Implementation

In this Section, we elaborate on each implementation step in different subsections. All the code has been released on GitHub [3].

[3] https://github.com/YuweiYin/UBC_CPSC_532V/tree/master/Assignment_1

3.1 Step 0: Keywords Extraction

Keyword Extraction As described in § 2.2, we extract question keywords qkw_i from the question q_i and choice keywords ckw_i^j from the choice c_i^j using the KeyBERT model, which uses BERT-embeddings (Devlin et al., 2019) and simple cosine similarity to find the sub-phrases in a document that are the most similar to the document itself [4].

Keyword Filtering After the initial extraction, we filter the choice keywords by sorting the extracted keyword list. Specifically, for each choice keyword $ckw_i^{j;k}$, we compute the similarity between $ckw_i^{j;k_1}$ and each $qkw_i^{k_2}$ in the question keywords qkw_i using the Sentence-BERT (Reimers and Gurevych, 2019) model [5].

To avoid dealing with too many pairs, we limit the number of a keyword list to no more than 5 for both question q_i and choice c_i^j. If a keyword list has more than five words, we only keep the first five ones. In addition, we filter out common words shared by all choices to avoid useless computation, because our goal is to compare different choices c_i^j in the given choice set c_i.

3.2 Step 1: Path Search in KG

After keyword extraction and filtering, we pair up all the keywords of this choice c_i^j and those of the corresponding question q_i, as shown in Equation 2.11. For each keyword pair <$ckw_i^{j;k_1}$, $qkw_i^{k_2}$> $\in \kappa$, we find valid paths from the choice keyword $ckw_i^{j;k_1}$ to the question keyword $qkw_i^{k_2}$,

For example, in Figure 1, the word set qkw_i = {"jim", "card", "debit", "spend", "credit"} is from the combined context and question. The word set ckw_i^j = {"card", "credit", "open"} is from the choice (a), so we have $|\kappa| = 3 \times 5$ pairs. However, we will ignore the <"card", "card"> and <"credit", "credit"> pairs, as mentioned in § 2.6. Thus, we will search in the knowledge base for 13 times to score choice (a).

Searching Algorithm Dijkstra's algorithm (Dijkstra, 1959) is a classic searching algorithm. It calculates the shortest path from the source node to the target node in a weighted graph in a breadth first search (BFS) manner. Since the knowledge base ConceptNet (Speer et al., 2017) is a graph with edge weights, it is suitable to apply Dijkstra's algorithm to find the best path in the graph.

[4] https://maartengr.github.io/KeyBERT/

[5] https://www.sbert.net/

The classic Dijkstra's algorithm is single-sourced, which means the BFS queue starts with the initial set {source_vertex} and then expands the searching scope by visiting all the neighbors. Considering the density of ConceptNet, where each vertex usually has about 10 neighbors, the number of nodes to visit grows exponentially as the depth D increases: $\mathcal{O}(10^D)$.

Actually, for each single-source BFS search, a `max_depth` of 3 will take about 2 minutes, a `max_depth` of 4 will take about $5-10$ minutes, and a `max_depth` of 5 will take about 30 minutes or longer, not to mention when `max_depth` is set as 6 or larger. We normally need to search $10-20$ times to evaluate a choice (say, 15), and there are $2-5$ choices in an example (say, 3). If the `max_depth` is 5, we need roughly $0.5 \times 15 \times 3 = 22.5$ hours to handle one example, which is not efficient.

Bidirectional BFS Hence, we improve the original Dijkstra's algorithm to a bidirectional BFS version, where we initialize two BFS queue, i.e., $\{v_s\}$ and $\{v_t\}$. The former denotes src-tgt search (from v_s to v_t) and the latter denotes tgt-src search (from v_t to v_s). Both two BFS processes progress simultaneously until there is a node matched (verbatim match). Then, we can build a valid path by reversely finding the root node of the search routes, as mentioned in § 2.4. The root node and target node of the src-tgt search are v_s and v_t respectively, the other way around for the tgt-src search.

Using the improved method, a `max_depth` of 6 will take about 2 minutes, and a `max_depth` of 8 will take about $5-10$ minutes for each single-source BFS search. This is much more efficient and effective than the single-source search algorithm.

Heuristic Search Pruning During the BFS process, many nodes in the queue are irrelevant to the target word of the current search direction (src-tgt search or tgt-src search). Based on this observation, we propose to prune the search tree by ignoring these irrelevant neighbor nodes. Specifically, we first compute the similarity s_{st} between the root word v_s and the target word v_t using the Sentence-BERT model. When visiting a neighbor node v_n, we put the node into the next BFS queue if the similarity between v_n and the target node (v_t in the src-tgt search or v_s in the tgt-src search) is larger than $\frac{2}{3}s_{st}$. This heuristic pruning method usually remove $\frac{1}{5}$ to $\frac{1}{3}$ of nodes for each BFS loop, which further improves the efficiency.

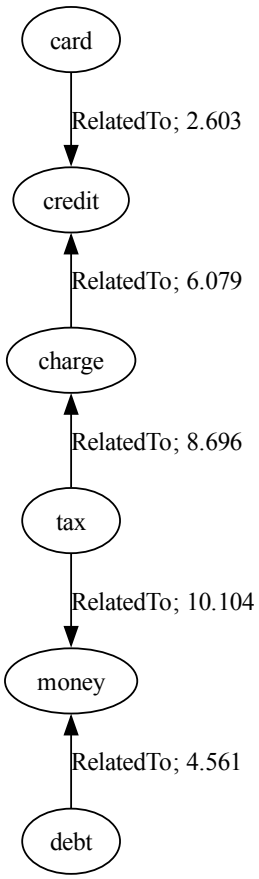

Figure 3: An example of path visualization.

3.3 Step 2: Path Visualization

After searching, we obtain valid concept paths. As illustrated in Figure 2, the green pair in Figure 1 <"card", "debt"> has a valid path "card" → "credit" → "charge" → "tax" → "money" → "debt", where the relations between each two words are all "RelatedTo".

We visualize the best path of every $<ckw_i^{j;k_1}, qkw_i^{k_2}>$ pair using Graphviz[6]. As shown in Figure 3, the path from "card" to "debt" is labeled with word names, relation types ("RelatedTo"), weights, and directions. All visualization figures are available on GitHub.

3.4 Step 3: Path Analysis

Convert Edges into Sentences In instances where language models are not accessible, we utilize the relation templates (Shwartz et al., 2020) from ConceptNet to transform the extracted path into a natural language format. Additionally, we offer the alternative of using `gpt-3.5-turbo` (ChatGPT (OpenAI, 2022)) for sentence generation. In

[6] https://www.graphviz.org/

Prediction results			
Example ID	# Choices	Answer	Prediction
1	5	e	a
2	2	b	b
3	2	b	a
4	3	a	c
5	3	a	a
6	2	a	a
7	2	b	b
8	2	b	a
9	5	a, b	a, c
10	2	a	a
Average	2.8	Accuracy: 50%	

Table 1: Prediction results.

addition, we refine the generation quality by providing a few-shot prompt template for in-context learning (Brown et al., 2020; Dong et al., 2022). This approach is aimed at enhancing the diversity of the generated sentences.

For example, the path in Figure 3 is transformed by the relation templates into the following natural language sentences:

```
# card is like credit
# charge is like credit
# tax is like charge
# tax is like money
# debt is like money
```

The sentences generated by GPT are below. We can observe that the sentences generated by the GPT model are more natural and fluent, with no errors in grammar or part of speech.

```
# Cards are related to credit.
# Charging is related to credit.
# Tax is related to charges.
# Tax is related to money.
# Debt is related to money.
```

3.5 Step 4: Choice Selection

After extracting all valid paths in Step 1 (§ 3.2), we apply the scoring method described in Equation 2.20. Then, we select a choice with the highest score, as described in Equation 2.21.

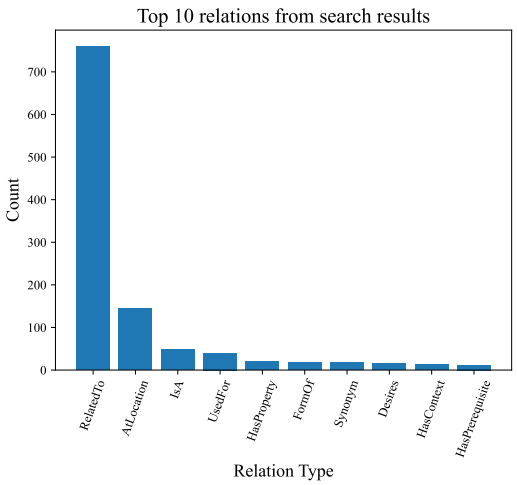

Figure 4: Count of the extracted relations.

4 Results

In this Section, we present the experimental results and analysis in § 4.1 and answer the assignment questions in § 4.2. All the experimental results and visualization figures are available on GitHub.

4.1 Result Analysis

As is shown in Table 1, our approach has an accuracy of 50% in predicting the best choice. Since the average number of choices is 2.8, an accuracy of 50% is better than the random classifier (35.7%).

Path Search is NOT Enough After analyzing the extracted paths in detail, we argue that solely matching keywords is far from solving commonsense QA problems. This is particularly evident in instances where human names are the source (or target) nodes. Take the sentences generated from the paths in example 7 as an example:

```
# lawrence is like usa
# usa is a type of country
# You are likely to find land in country
# land is like ground
# You are likely to find oil in ground
# oil is like hates
```

Further Discussion A remedy for such cases could be using a named entity recognition (NER) model to identify named entities in the keyword set and then filter out them. Yet, this approach may bring searching issues since the commonsense knowledge base often associates human names with specific countries and regions. However, this linkage normally does not convey useful information

about sentiments or behaviors, which are vital aspects especially when the context and question are intricately connected to human characteristics and actions. This pattern shows the limitations of commonsense knowledge bases in representing complex, human-related concepts.

4.2 Answers to Assignment Questions

- **Q1**: *Which types of knowledge are the most common in the extracted paths?*
  A1: The most common type of knowledge is the relationship "RelatedTo", as is shown in Figure 4.
- **Q2**: *Which types of knowledge are the most accurate / reliable?*
  A2: The predicted outcomes are presented in Table 1. Examining all of the paths of the correct predictions, we find that the "RelatedTo" relation consistently emerges as the most accurate/reliable in our approach. This is primarily because edges associated with a "RelatedTo" relation typically possess a greater weight compared to edges linked with other types of relations.
  Nonetheless, if we focus solely on the semantic implications of the extracted pairs without taking their weights into account, the "IsA" relation emerges as the most accurate/reliable. This is because of its nature of denoting that one entity is included within another, implying the entities have various shared attributes.
- **Q3**: *Which types of knowledge are the most useful for each instance (i.e. contain information useful for answering the question)?*
  A3: The "RelatedTo" relation. To begin with, as mentioned earlier, the "RelatedTo" relation is the most frequent and carries the most weight compared to other relations. In our algorithm, we determine the most likely choice by calculating the average weight of edges between keywords from the source text (which includes the context and the question) and the target text (the choices) (§ 2.6). As a result, the "RelatedTo" relation naturally becomes the most significant in our analysis. Furthermore, the "RelatedTo" relation frequently serves as a connector for entities with widely different semantic meanings, effectively creating links between words in the source and target texts. Consequently, it undeniably offers the most valuable information by bridging distant concepts.
- **Q4**: *Do you recognize any problems with the knowledge or path search approach? What can be improved?*
  A4: In our approach to the search algorithm, we integrated Dijkstra's algorithm with bidirectional Breadth-First Search (BFS) to enhance efficiency (§ 3.2). This was achieved by initiating the search concurrently from both the source keywords and the target keywords, thereby decreasing the overall runtime. Additionally, we set a threshold for node pruning based on the cosine similarity between the source node and the target node (§ 3.2). This approach narrowed down the number of nodes to be considered effectively. While our algorithm has delivered promising outcomes, there's potential for further enhancement, which can be achieved by developing a more refined heuristic for both pruning and the path search process.
  Enhancing the keyword extraction process also presents an opportunity for improvement. Our current method utilizes a zero-shot pre-trained model for this task (§ 3.1), but we frequently encounter issues where high-probability keywords turn out to be human names (§ 4.1). This diminishes the efficiency and accuracy of employing path search for choice selection. Therefore significant potential exists to refine the keyword extraction pipeline, focusing on identifying the most critical keywords for commonsense reasoning, rather than relying on conventional keywords.

5 Conclusion

In conclusion, our paper formally defines a comprehensive approach to addressing commonsense reasoning challenges. Initially, we employed KeyBERT for the extraction of pivotal keywords from the given context, questions, and choices. Subsequently, we integrated bi-directional Breadth-First Search (BFS) with Dijkstra's search algorithm to trace the paths between the extracted keywords. To enhance the efficiency of our algorithm, we also implemented heuristic pruning techniques.

Furthermore, we crafted a set of criteria for choice selection for decision-making, leveraging these criteria to pinpoint the most appropriate choice. Visualization of these paths was achieved

Criteria	Fulfillment Statement
The graphs of all examples are included	Yes. (On GitHub)
Which types of knowledge are the most common?	See QA 1 in § 4.2
Which types of knowledge are the most accurate / reliable?	See QA 2 in § 4.2
Which types of knowledge are the most useful for each instance?	See QA 3 in § 4.2
Recognize problems and improvements	See QA 4 in § 4.2
Graph creating implementation	See § 3.3
Path search Implementation	See § 3.2
Path search for a QA example Implementation	See § 3.1, § 3.2, § 3.4, and § 3.5
Graph visualization implementation	See § 3.3

Table 2: Statement of assignment fulfillment according to the marking rubric.

through the Graphviz package to show the connections between keywords. In addition to this, we adopted two distinct methodologies to transform extracted paths into natural language representations: one using pre-defined templates and the other utilizing LLMs.

Finally, we delved into a thorough analysis of our methodologies and results. Our comments shed light on the strengths and potential limitations of the path search approach as a solution for commonsense reasoning. This empirical study not only validates the effectiveness of our methodology but also lays the groundwork for future work focusing on enhancing techniques in commonsense reasoning.

Assignment Fulfillment

In this Section, we claim that we have fulfilled the assignment requirements according to the grading rubric. The criteria-to-section mapping is shown in Table 2.

References

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, volume 33, pages 1877–1901, Virtual Event. NeurIPS.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Edsger W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James F. Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. *CoRR*, abs/1604.01696.

OpenAI. 2022. Chatgpt. *OpenAI Research*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. 2011. Choice of plausible alter-

natives: An evaluation of commonsense causal reasoning. In *Logical Formalizations of Commonsense Reasoning, Papers from the 2011 AAAI Spring Symposium, Technical Report SS-11-06, Stanford, California, USA, March 21-23, 2011*. AAAI.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473, Hong Kong, China. Association for Computational Linguistics.

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Unsupervised commonsense question answering with self-talk. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4615–4629, Online. Association for Computational Linguistics.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 4444–4451. AAAI Press.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4149–4158. Association for Computational Linguistics.

Ben Zhou, Daniel Khashabi, Qiang Ning, and Dan Roth. 2019. "going on a vacation" takes longer than "going for a walk": A study of temporal commonsense understanding. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3363–3369, Hong Kong, China. Association for Computational Linguistics.