NORTHWESTERN UNIVERSITY

Computer Science Department

Technical Report
Number: NU-CS-2022-06

June, 2022

Symbolic Data Augmentation for Assisted Neural Reasoning

Muhan Li

Abstract

This work proposes a novel data augmentation technique based on symbolic methods SDAR that generates annotations in the text space. The SDAR annotator is an ensemble of multiple sub-annotators, each equipped with a searcher and a symbolic knowledge retriever. We demonstrate that SDAR can boost the performance of smaller models to a comparable degree of or even surpass larger models with a magnitude more parameters and establish the STOA single model performance on OpenBookQA.

Keywords

Data Augmentation, Knowledge Retrieval, Symbolic Reasoning, Transformer

NORTHWESTERN UNIVERSITY

Symbolic Data Augmentation for Assisted Neural Reasoning

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

MASTER OF SCIENCE

Field of Computer Science

By

Muhan Li

EVANSTON, ILLINOIS

June 2022

ABSTRACT

Recent progress in various language processing tasks which require multi-step reasoning process in commonsense or scientific domains, are usually achieved by large-scale language models. While these models have achieved decent performance, their parameter number is becoming increasingly difficult for commodity hardware to handle. In this work, we propose a novel data augmentation technique based on symbolic methods **SDAR** that generates annotations in the text space. The SDAR annotator is an ensemble of multiple sub-annotators, each equipped with a searcher (usually neural) and a symbolic knowledge retriever. The annotations by each sub-annotator generated are combined and sent along with the original input to the target language model. We demonstrate that SDAR can boost the performance of smaller models to a comparable degree of or even surpass larger models with a magnitude more parameters, and establish the state-of-the-art single model performance on OPENBOOKQA.

ACKNOWLEDGEMENTS

I would like to express my gratitude for my supervisor Prof. Douglas Downey and Dr. David Demeter who made this work possible. The in-depth guidance and inspirational advice from David have enabled me to make many improvements which are crucial to the final result achieved in this paper. It was a privilege for me to work with these two patient, enthusiastic, and knowledgeable advisors and got constantly enlightened during the whole research.

My sincere appreciation also goes to my loving and supportive family, especially my parents Fan Li and Jing Liu who provided me with all the support I need, spiritually and financially. Without them, I would never be able to conduct this research in the first place or fully devote my effort into the work. Thank you for comforting me whenever I felt defeated and encouraging me to regain the motivation for advancement. My heartfelt thanks for your love.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

Language models based on the transformer architecture[1] and using unsupervised learning objectives for pre-training has achieved promising results on various NLP tasks such as question answering, text classification, sentiment analysis. Many famous publicly available models including BERT [2], GPT [3], RoBERTa [4], and XLNet [5] are pre-trained using large-scale generic corpora such as Wikipedia with millions of entries. However, in order to apply them to domain specific tasks, such as question answering on biomedical datasets [6], the performance of a generic model could be less than ideal even after fine-tuning for two reasons. First, models pre-trained on generic corpora are usually lack of knowledge on domain specific phrases in inputs. Second, even if knowledge of inputs is encoded in parameter space, models may fail to capture intermediate knowledge pertaining to the reasoning process.

For the first challenge, many methods are proposed, such as extending the vocabulary of the model [7], fine-tune an ensemble of generic and domain-specific pre-trained models [6], incorporate domain related knowledge graph encodings [8] [9] [10], using information retrieval technique to enhance input with additional text knowledge [11], or utilizing another language model as the knowledge source [12].

The second challenge is harder to address as (1) Intermediate reasoning steps usually come from a potentially large hypothesis space [13]. (2) As the number of reasoning steps increases, the limited depth of current language models becomes insufficient, and decomposition of the reasoning task is required [14] [15] [16]. Since shallow retrieval algorithms usually fail on this type of task, current methods usually involve using a deep model to generate intermediate steps sequentially using incomplete observations.

Apart from difficulties in training the model to learn the reasoning process, another problem is finding appropriate knowledge sources as the supporting context. While there exists a plethora of sources, many of them are in unstructured textual form, and only a tiny part consists of knowledge bases or knowledge graphs. A knowledge base or a knowledge graph is usually preferred due to their structural representation ideal for multi-hop retrieval which emulates reasoning steps. Therefore, unifying these two sources can lower the requirement for high-quality structured knowledge by utilizing textual corpora. One way to achieve unification is transformation. Several methods aiming at converting knowledge from one form to another in both directions are proposed [17] [18] [19], but they are either troubled by over-generation of artifacts or loss of information.

In this paper, we propose a novel annotator architecture **S**ymbolic **D**ata **A**ugmentation for assisted neural **R**easoning (**SDAR**) designed to cope with three problems listed above. The annotator system is an ensemble of several individual sub-annotators designed for different domains. Each annotator comprises of a searcher which outputs a clue for knowledge retrieval guidance, and a symbolic knowledge retriever taking the clue as input and outputs factual knowledge. This architecture ensures that the searcher can deal with the noisy data from fine-tuning datasets, while the symbolic nature of the retriever can guarantee the correctness of generated knowledge. In one of the sub-annotators, we introduce a unification method that can utilize knowledge in the textual form as well as the triple form. We then demonstrate that **SDAR** can assist models with lesser parameters to achieve a performance comparable to models with a magnitude more parameters on multiple question answering datasets, and set the state-of-the-art single model performance on one dataset.

The main contributions of this work are summarized as follows:

- We propose **SDAR**, a data augmentation framework based on mixed neural-symbolic methods for reasoning, in Chapter 3

- A knowledge unification and retrieval method without the need of performing conversion on sources, and avoid over-generation or under-generation, will be introduced in Chapter 3.
- In Chapter 4, experiments are performed to demonstrate that **SDAR** can assist models achieve 3-5% of performance gain across three question answering datasets: OPENBOOKQA [20] COMMONSENSEQA [21] ARC [22], and achieve STOA performance on OPENBOOKQA. A comprehensive analysis of factors influencing the performance of **SDAR** is also included.
- Conclusion and possible improvements in Chapter 5.

CHAPTER 2
RELATED WORK

2.1 Recent general advancements

The transformer architecture [1] started a revolution in the natural language processing field. The once popular recurrence mechanism was completely replaced by the self attention mechanism, the backbone of various transformer models. In essence, the self-attention mechanism lets the model to compute a relevance score which is then multiplied with each token, generating useful internal representations that cover over the whole input sequence. This process removes the necessity to introduce recurrent or convolution layers which are usually required to achieve aggregation of inputs. Although self-attention was not only proposed in the natural language processing literature [23] [22], employing it in an end-to-end manner is unique in the transformer.

Another major advancement is the appearance of language models pre-trained on massive corpora such as C4 [24] with unsupervised training objectives. Although there exists some subtle differences between different proposed methods, all unsupervised training procedure includes the prediction of words that are either manually removed from the context, also known as the masked language modeling (MLM) objective [2] [4] [25] [24], or dependent on previously appeared words, also known as the language modeling (LM) objective [26]. However, both objectives are not perfect and are accompanied by unique challenges. The MLM objective trains the model in an auto-encoding fashion with bidirectional attention, which requires the independence assumption on input tokens. Additionally, the artificial [MASK] token creates a discrepancy between pre-training and fine-tuning, and since only a small portion of tokens are masked, inefficient utilization of text corpora is unavoidable. In contrast, the LM objective seeks to estimate the probability distribution of inputs auto-repressively, conditioning each predicted token on the previous (or succeeding) tokens in a uni-directional manner. This approach allows joint probability of multiple words to be modeled correctly, but the loss of bidirectional information may significantly weakens model performance on downstream tasks.

Several improvements are made to counteract the deficiencies of both methods. Permutation language modeling (PLM), proposed in XLNet [5], uses LM as the base and models the distribution of the predicted token using all possible permutations to achieve a similar effect of bidirectional attention supported by MLM only. ELECTRA [27] uses a discriminator on top of the model trained by MLM to eliminate the discrepancy issue brought by the `[MASK]` token. We summarize the optimization targets of each training objective in Eq. 2.1. The meaning of each symbol is listed below.

- θ: Model parameters.
- T: The length of input tokens.
- x_t: Token at position t.
- $\tilde{\mathbf{x}}$: Whole sequence with multiple tokens replaced with `[MASK]`.
- $\mathbf{x}_{<t}$: Sequence of tokens before the token x_t.
- $\mathcal{Z}_T$: All permutations of index sequence $[1, 2, \ldots, T]$
- $\mathbf{x}_{\mathbf{z}<t}$: Sequence of tokens selected with partial index sequence $\mathbf{z}_{<t}$.

More recent task-agnostic researches either has a concentration on enhancing pre-training procedures or model architectures. For pre-training procedures, Liu et al. proposed to remove the next sentence prediction (NSP) objective of BERT [2], increase pre-training batch-size and uses a 10x larger corpus compared to what BERT uses in RoBERTa [4]. Lan et al. replaced the NSP objective with a new sentence-order prediction (SOP) objective and reduced parameter number by decomposing the word embedding matrix into wo smaller matrices in [25]. Sanh et al. used knowledge distillation to train a much smaller student BERT model with the dense distribution signal from the larger teacher model and achieve a similar performance as the original BERT [28].

$$\begin{aligned} MLM &: \max_{\theta} \sum_{t=1}^{T} \log p_{\theta}(x_t|\tilde{\mathbf{x}}) \\ LM &: \max_{\theta} \sum_{t=1}^{T} \log p_{\theta}(x_t|\mathbf{x}_{<t}) \\ PLM &: \max_{\theta} \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T} \left[\sum_{t=1}^{T} \log p_{\theta}(x_t|\mathbf{x}_{\mathbf{z}<t}) \right] \end{aligned} \tag{2.1}$$

On the other hand, generic architecture researches focus on two methods of improvement. The first is optimizing current model architectures with more complex designs, such as the DeBERTa [29], which incorporates additional relative position encoding for each token and disentangled attention for both the content encoding and relative position encoding. In the TUPE architecture proposed by Ke et al. [30], a similar approach is employed in the self attention layer. Another line of research takes advantage of the computational power of newer hardware and larger pre-training datasets, backing their performance improvement on the sheer increase in parameter number. As dense models are hindered by the memory limit, sparsely-activated model with a gigantic amount of parameters become the new focus. The most popular way to implement sparser models is through mixture-of-experts (MoE) [31], which usually involves using a switch like architecture to route inputs to the correct sub-model. Newer architectures usually contains hundreds of billions [32] [33] or even trillions of parameters [34]. This raises concerns about the environmental and financial cost of training such large models, and most importantly, that language models with such a large parameter count is only memorizing shallow probabilistic information of how linguistic forms combine, rather than the internal meaning of words and higher order relations between them [35].

2.2 Knowledge enhanced text understanding

2.2.1 Sources of knowledge

In general, knowledge is the presentation of understanding related to specific subjects within a given context. Whether the form of the system is designed for natural language generation (NLG) or natural language understanding (NLU) tasks, both forms may need access to two forms of knowledge: internal and external. Internal knowledge includes features from input text(s), such as keywords, topics, relational graph structure, and various linguistic features. Naturally, pre-trained models already comes with this type of knowledge encoded in their parameter space. A wide area of research aims at probing or extracting such knowledge from pre-trained models. Petroni et al. designed the LAMA probe [36] based on manually curated templates, demonstrating that pre-trained BERT has an outstanding memory of facts related to geological names, special entities, general capabilities and properties of common subjects such as animals, etc. Jiang et al. designed a mixed prompt system named LPAQA [37], consisting of manually designed, paraphrasing-based and mining based prompts to test the accuracy of different ensembles of prompting methods. Yang et al. used the GPT-2 model to generate synthetic examples and implemented a filtration method based on influence functions [38] to enrich existing fine-tuning datasets.

Methods designed to utilize external knowledge has a much wider range of sources compared to internal knowledge, include but not limited to commonsense triples [39] [40], ontology used for reasoning [41] [42] [43], keywords [44], graph-embeddings [45], retrieved background text [11] [46]. From the perspective of complexity and hierarchy richness, external knowledge could be roughly categorized into structral and non-structral sources.

2.2.2 Challenges in knowledge-enhanced text understanding

To start with, the first challenge met by knowledge enhanced systems is **obtaining** useful knowledge from a potentially large and diverse collection. The second challenge is finding a way to **integrate** the knowledge into existing models and leverage it by a thorough **understanding**.

(1) Retrieval and integration of unstructured text

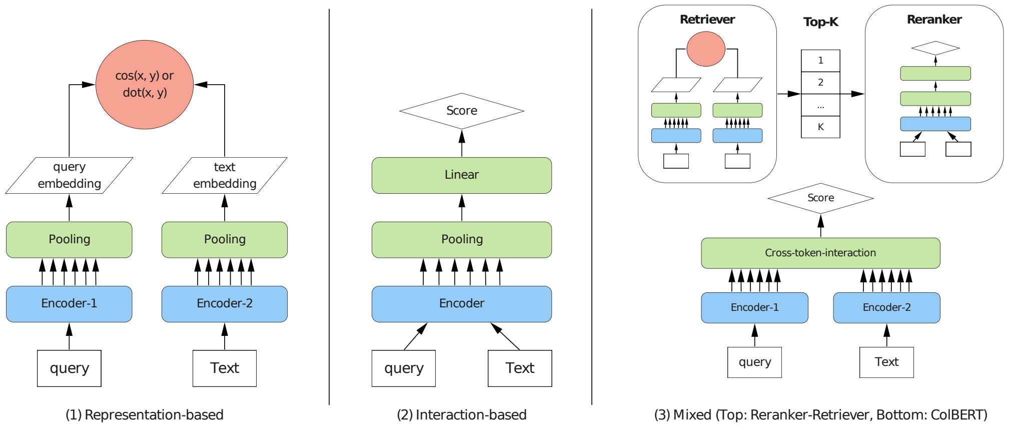


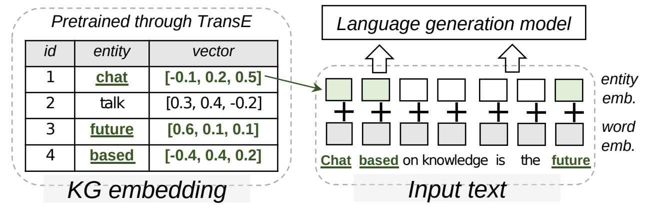
Figure 2.1: Three types of dense retrieval methods

Integration of text is trivial for transformer based models such as BERT [2], and there exists a great variety of methods to obtain useful information from an unstructured text corpus. For classical methods, variants of TF-IDF [47] [48] [49] [50], BM25 [50] [51] are most commonly used methods, one special method used by Ghazvininejad et al. [52] is retrieving target text entry by first ranking them by text similarity and then perform matches with an index of named entities. In recent years, deep retrieval models has became the new standard, dense retrievers can be generally divided into three types: representation-based, interaction-based, and mixed style retrievers [53].

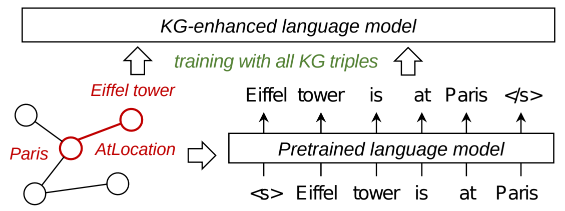
Representation-based retrievers such as the dense passage retrieval [54], uses two independent encoders to produce encoding for queries and supporting texts respectively, then perform matching based on cosine-similarity or unnormalized dot-product. Interactive-based retrievers take one query and one supporting text at the same time and output a relevance score [55], this design allows cross-attention between the query and document to be computed and enables rich interaction between two parts. However, for large-scale document bases with millions of entries, since one forward process needs to be performed for every pair of query and document combination, computation costs become prohibitively expensive. Mixed style retrievers combine the benefits of both retrievers, one simple way is to let the interactive-based retriever select a relatively small set of candidates, then perform fine-grained filtration using interaction-based retrievers [56]. Khattab et al. [57] proposed the ColBERT architecture, adding a token-interaction layer on top of the bi-encoder structure used in interactive-based retrievers to calculate the similarity score.

(2) Retrieval and integration of structured knowledge graphs

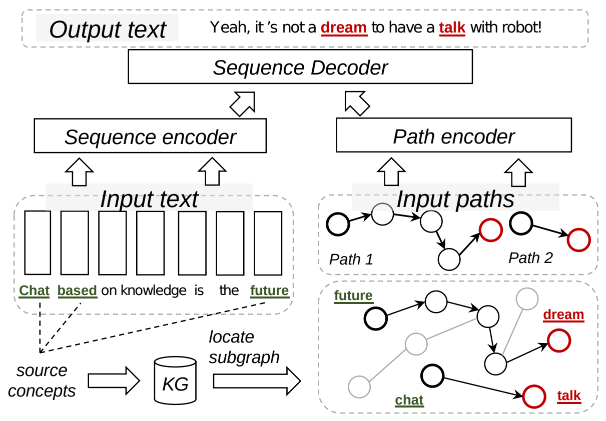
There exists several methods of integration [58], One category of methods use the knowledge graph as a whole and do not perform a selective retrieval process, or just retrieve nearest triples withing a fixed depth [9]. In [10], they use a 2-layer MLP with the concatenation of the output from BERT and graph meta data, to combine textual information with graph information. In the study of Jun et al. [8], the same combination method is also used. KI-BERT proposed in [45] use a special token embedding transformed from word embeddings along with knowledge graph embeddings from ConceptNet for atomic entity representation with text, as shown in 2.2 (M1). Guan et al. [59] directly fine-tuned their model on senetences converted from knowledge triples in ConceptNet and ATOMIC, as shown in 2.2 (M2). Wang et al. [60] proposed a GNN architecture based on edge counting to integrate information from all entities as show in 2.2 (M4). K-BERT poposed in [9]


Pretrained through TransE
id | entity | vector
1 | chat | [-0.1, 0.2, 0.5]
2 | talk | [0.3, 0.4, -0.2]
3 | future | [0.6, 0.1, 0.1]
4 | based | [-0.4, 0.4, 0.2]
KG embedding
Language generation model
entity emb.
word emb.
Chat based on knowledge is the future
Input text

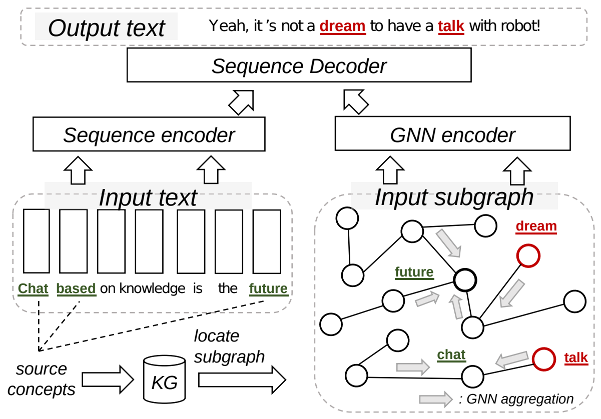
(M1) Incorporate KGE into language generation


KG-enhanced language model
training with all KG triples
Eiffel tower
Paris
AtLocation
Eiffel tower is at Paris </s>
Pretrained language model
<s> Eiffel tower is at Paris

(M2) Transfer knowledge into pretrained LM



(M3) Performing path reasoning on KG



(M4) Aggregating sub-KG via GNN

Figure 2.2: Four typical methodologies for incorporating KG semantics. Graph from [58]

encode a sentence tree comprised of a trunk representing the input textual sequence and fixed-depth branches for triples in knowledge graphs. They flatten this structure using a visible matrix which uniquely encodes the topology, however, both KI-BERT and K-BERT requires manual injection of knowledge triples and therefore entity detection and normalization must be performed in advance.

The pattern most related to methods used in this work is 2.2 (M3), since reasoning usually requires to focus on multiple inter-related decisions, a subset of the whole knowledge graph [61]. Lao et al. [62] proposed the PRA algorithm which first performs random walks on the whole graph and then selects multiple paths by a set of metrics. Wang et al. [63] proposed to perform random works locally and globally to prevent bias encoded in local neighborhood graph structures. Another line of research are reinforcement learning methods [61] [64] [65] which trains a policy network that takes a starting node n_0 and performs multi-hop searching until they reach the target node n_t, n_0 is usually retrieved by correlation with input texts on the whole graph. The extracted path is then converted to the text form which can be processed by encoders. However, these researches only provide reward signal to agents when the target node is reached, which may potentially result in the sparse reward problem in RL literature [66], especially when the average degree of nodes and length between source and target nodes are large. Another potential weakness is that, although these works includes factors for promoting sampling diversity and efficiency (shorter path length), the coherency between each sampled node is not considered. Xu et al. [67] proposed to use TransE [68] and add a global coherence penalalty to the reward signal.

2.3 Step by step reasoning

Even if we have the ground truth knowledge that is most related to the question and desired answer, there is still no guarantee that the language model is correctly understanding the reasoning process. May be its just a more localized ”statistical parrot” [35] that selects the answer most related to the given text. In order to elicit the reasoning process in an explicit manner, letting the model perform intermediate steps becomes a common approach in several researches. Bhagavatula et al. [13], proposed to use an intermediate hypothesis, and condition the final conclusion of the model on the hypothesis using the Bayes Rule. Wei et al. [14] proposed to train the model with human annotated intermediate reasoning process, and output answer with the reasoning process during inference. Khot et al. [15] trains the model to generate intermediate questions and condition the final answer on intermediate questions and answers from other QA systems. Nye et al. [69] proposed to decompose a series of input computation queries, and let model generate intermediate states to perform reasoning.

CHAPTER 3
THE SDAR FRAMEWORK

3.1 Task definition

We use a similar probabilistic definition as Bhagavatula et al. [13] used in their studies. Let $\mathcal{A}$ be the annotator ensemble and $\mathcal{Q}$ be the question answering model taking all annotations. Let q be the query and a be the desired answer, we formulate the generic goal for natural language inference (NLI) tasks as Eq. 3.1, where $\theta_{\mathcal{A}}$ and $\theta_{\mathcal{Q}}$ are the parameters of $\mathcal{A}$ and $\mathcal{Q}$ respectively.

$$\theta^*_{\mathcal{A}}, \theta^*_{\mathcal{Q}} = \underset{\theta_{\mathcal{A}}, \theta_{\mathcal{Q}}}{\arg\max}\, P_{\theta_{\mathcal{Q}}}(a|h, q) P_{\theta_{\mathcal{A}}}(h|q) \tag{3.1}$$

We choose to use the non-differentiable text form as the medium of hypothesis h to decouple the annotator ensemble from the question answering model, this allows the question answering model to utilize a sub part of all annotators $\mathbf{A} \subset \mathcal{A}$ and ignore information irrelevant to the down-stream task. Therefore the surrogate optimization target becomes Eq. 3.2

$$\begin{aligned} \theta^*_{\mathbf{A}} &= \underset{\theta_{\mathbf{A}}}{\arg\max}\, P_{\theta_{\mathbf{A}}}(h|q) \\ \theta^*_{\mathcal{Q}} &= \underset{\theta_{\mathcal{Q}}}{\arg\max}\, P_{\theta_{\mathcal{Q}}}(a|h, q) P_{\theta^*_{\mathbf{A}}}(h|q) \end{aligned} \tag{3.2}$$

For multiple-choice question answering tasks specifically, suppose there are M sub-annotators, each one denoted as $A_i \in \mathbf{A}, i \in \{1, 2, \ldots, M\}$, and N choices, the detailed form of Eq. 3.2 can be represented as Eq. 3.3.

$$\begin{aligned}\theta^*_{A_i} &= \arg\max_{\theta_{A_i}} P_{\theta_{A_i}}(h|q) \\ \theta^*_{\mathcal{Q}} &= \arg\max_{\theta_{\mathcal{Q}}} P_{\theta_{\mathcal{Q}}}(a|(h^1_{a_1},\ldots h^N_{a_1},\ldots h^N_{a_M}),(h_1,\ldots,h_M),q) \\ &\prod_{i=1}^{M}\prod_{j=1}^{N} P(h^j_{a_i}|h_i,q)P_{\theta^*_{A_i}}(h_i|q)\end{aligned} \tag{3.3}$$

The second optimization target may seem complicated at the first glance, it actually means first generate a hypothesis using each individual annotator A_i, then perform answer specific symbolic knowledge retrieval for each answer, then let $\mathcal{Q}$ reason over all collected information. The term $P(h^j_{a_i}|h_i,q)$ actually represents the symbolic retrieval process and therefore is a constant and can be ignored during optimization.

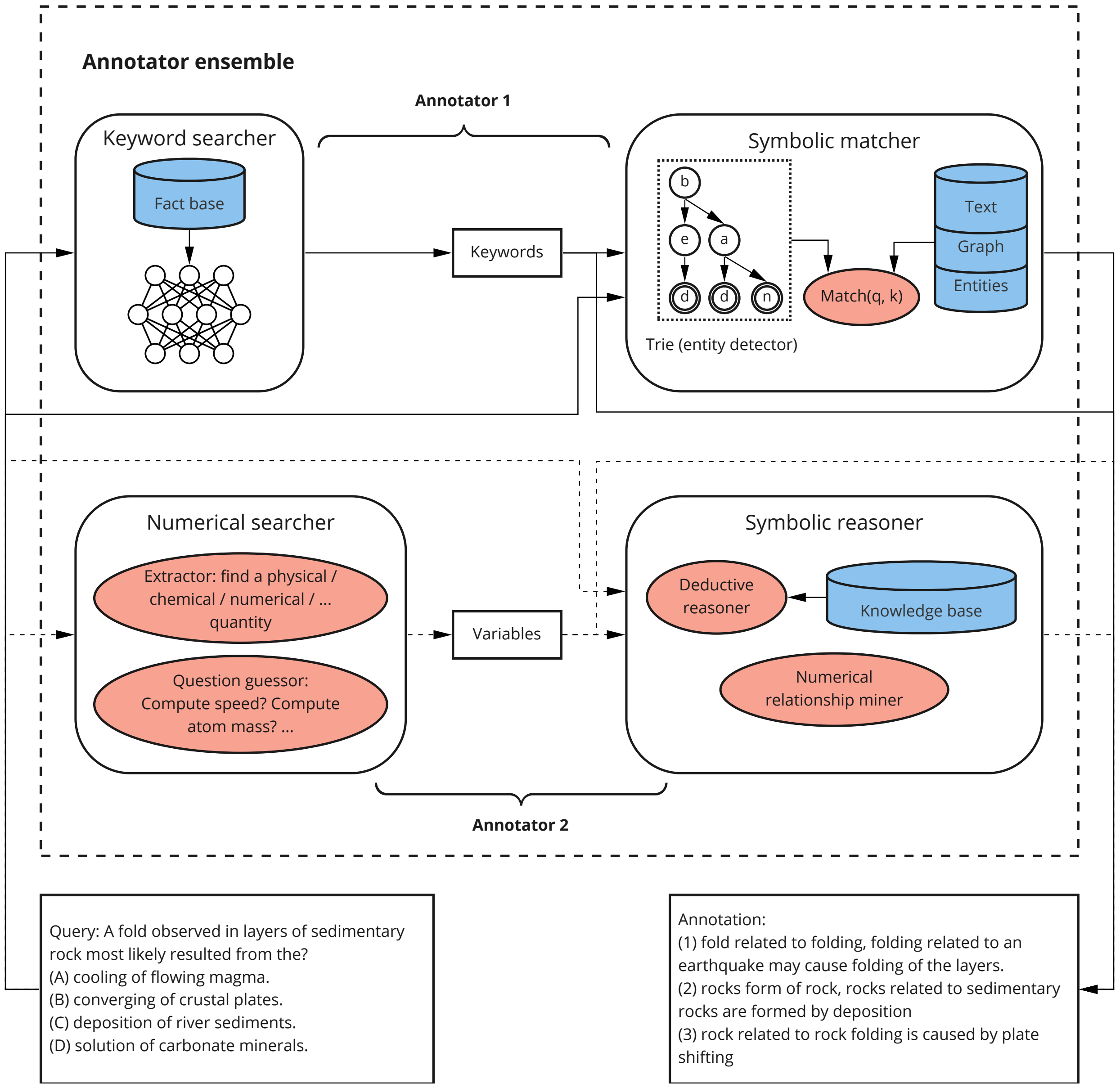
This task formulation corresponds to the **Fully Connected** model described in [13]. In experiments we observe that this design constantly outperforms the simpler **Linear Chain** model where $\mathcal{Q}$ is not conditioned on hypotheses generated by searchers.

We also note that while this design does not support reasoning with more than two steps, one can use this architecture as a sub module and chain it into multiple stages as described in [15].

3.2 Architecture of the annotator

The annotator ensemble $\mathcal{A}$ is the core of this work. In general, there exists several annotators working in parallel, each annotator has a searcher module which provides the initial hypothesis h. A knowledge retriever based on symbolic methods then takes in h and a potential a and outputs answer specific hypothesis h_a. This process is illustrated in Fig. 3.1.

Two types of annotators are implemented. The first type traverse an enhanced knowledge graph



Query: A fold observed in layers of sedimentary rock most likely resulted from the?
(A) cooling of flowing magma.
(B) converging of crustal plates.
(C) deposition of river sediments.
(D) solution of carbonate minerals.

Annotation:
(1) fold related to folding, folding related to an earthquake may cause folding of the layers.
(2) rocks form of rock, rocks related to sedimentary rocks are formed by deposition
(3) rock related to rock folding is caused by plate shifting

Figure 3.1: Structure of the annotator assembly. (1) Dashed arrows of annotator 2 indicate that there is no information passed through this route, since no numerical value is detected in the query. (2) In practice, keywords are not directly passed to $\mathcal{G}$ since we discover that passing the facts where keywords are extracted has a better performance. (3) The question answering model uses the concatenation of query and annotation as input, therefore the conditional dependence between a and q in $P_{\theta_Q}(a|h_a, h, q)$ is not shown in this graph.

to retrieve necessary text and triple information. The second type performs deductive reasoning on input variables and perform required computation.

For the first annotator, in experiments, we discover that a common problem of directly employing generative models is redundant artifacts in the generated facts or keywords. For decoder models like GPT-2 [26], we generate a continuation of the question, and for encoder-decoder models like T5 [24], we combine question and choices as the query. When trained to generate a complete sentence resembling the fact, models, sampled outputs are coherent but usually missing critical keywords. In contrast, training the model to generate keywords directly usually leads to unrelated ones appearing as artifacts, the low occurrence frequency of keywords has a significant negative impact on the down-stream retriever performance. This issue could not be alleviated by the token dropout method proposed in [70] or the order permutation method introduced in [71]. We discover that by supplying a manually curated fact base and use the reranker-retriever architecture in Fig. 2.1 (3) to select top-5 facts this base, then perform keyword extraction on selected facts lead to the best accuracy. The keyword extraction method is a simple function which perform tag-of-speech tagging and select nouns that do not belong to the stopword set, the nouns are then normalized to their singular form by stemming. This function can be trivially implemented using the popular framework NLTK [72].

The symbolic matcher in the first annotator is a heuristic based complex program which will be described in details in section 3.4 of this chapter. We first construct a mixed graph structure of text and a base knowledge graph (eg: ConceptNet [39]), and a Trie structure which stores all basic entities in the base knowledge graph. The Trie serves as a shallow entity matcher, recognizing possible entry points in the query, and a match function $Match(q, k)$ is used to perform the retrieval process. Although more complicated named-entity-recognizers exist [73] [74] [75], this simple method performs good enough when path length is limited to 2 or 3 hops.

The second annotator focus on providing information related to numerical reasoning, which is currently one known weakness in pre-trained language models [14]. For simplicity we do not apply the neural-based extraction mentioned in [43] to the searcher, and instead use template-based matching to extract variables. Specifically, when numerical entities are detected in the question or choices, such as the chemical equation in question ”Which statement is best supported by the chemical equation shown? $6CO_2 + 12H_2O + light \rightarrow C_6H_{12}O_6 + 6O_2 + 6H_2O$”, extractor will be triggered to extract the equation as a single variable, and question guessor will then output question type **chemical-equation-explain** since there is no specific requirement in the question. This question type assists the reasoner to narrow down the search space, and only select all rules with the type tag **chemical-equation-explain**, which converts the equation into multiple variables that can be converted to text explanations such as ”There are $6CO_2$ molecules on the reaction side”. The same process applies to other questions like ”An airplane takes off from Boston for the 980 km trip to Detroit. The plane lands two hours later. Which of the following best describes the average speed and direction of the airplane’s flight?”.

The symbolic reasoner is based on the forward-chaining algorithm which is commonly used in automated theorem proving [76] and planners [77] [78]. In each reasoning step, rules with all prerequisites satisfied will applied on existing variables, and output new variable. The process we use differs from traditional boolean forward chaining which generates new grounded literals, since numerical computation needs to be performed which is not supported by classical implementations.

Another component of the symbolic reasoner is the numerical relationship miner. Due to the fact that there exists too many diverse rules and manually encoding them all infeasible, the miner just perform random walks of common arithmetic steps on all numerical values appearing in the text, two at a time (since common arithmetic operands are binary), until the value mentioned in the choice is reached. An example would be ”John takes care of 10 dogs. Each dog takes 0.5 hours a

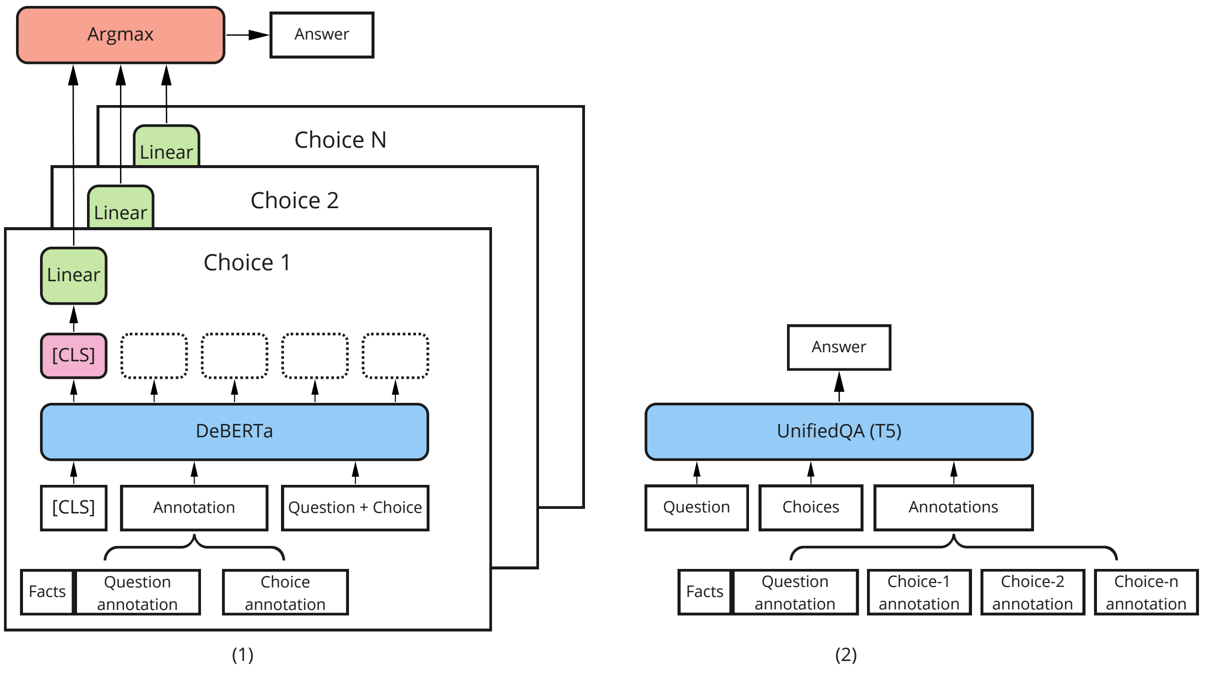


Figure 3.2: Question answering models. (Note: Following the approach of [79], we removed the pooling layer from DeBERTa and only use a shared linear layer to compute a score for each question-choice pair.)

day to walk and how many hours a day does he spend taking care of dogs?", the miner first detects 10 and 0.5, then detects that $10 * 0.5 = 5$ can reach one of the choices and generates annotation "10 multiplies 0.5 is 5". This method is merely a complement for the symbolic reasoner and can be easily fooled by adversarial choices generate by humans, such as 10.5 hours would lead the miner to generate "10 plus 0.5 is 10.5".

3.3 Architecture of the question answering model

Two language models are used to demonstrate the efficiency of our method compared to pure parameter size up-scaling. For OPENBOOKQA and COMMONSENSEQA, we use the DeBERTa model [29], for ARC, the UnifiedQA model [11] based on T5-11B is used. Following previous practices, we collect all annotations as a paragraph of supporting context, and concatenate it with the question and choice pair as shown in Fig. 3.2. In actual practice we discover that concatenating annotations as one segment (segments are region of inputs divided by the separator token [`SEP`]) is important and has a profound impact on the fine-tuning performance. Other input schemes such as inserting annotations inside the source sentence, or concatenating them after the source sentence result in inferior performance. We group the facts retrieved by the searcher and question annotations together since they represent the hypothesis h sampled from $P_{\theta^*_{\mathbf{A}}}(h|q)$, the choice-specific annotations corresponds to h_a sampled from the non-parameterized distribution $P(h_a|h, q)$.

3.4 A novel way to unify and retrieve knowledge

Each text entry in a corpus could be considered as a tiny knowledge graph with its internal relations implicitly represented in the sequence. Although currently several works [80] [17] [18] [19] aiming to extract useful relations has interesting results, the outcome either: (1) contains a limited representation of relations. [81] or (2) has false relations or missing important relations. Since language models already possess the ability to understand this internal relationship, we decide to present text as it is rather than performing some kind of transformation.

In order to efficiently retrieve the text knowledge, we use the constructed Trie data structure coming along with the knowledge graph to perform a shallow extraction of all occurring concepts inside the sentence. The sentence is considered as a black-box super concept node (referred to as **composite**) with an unknown internal graph structure, whereas nodes made up of the graph are identified. During traversal, we simply ignore the internal graph and select any of its component node as an entry / exit point.

Following [82], we define the similarity metric used in graph traversal as follows. Let x_i be a target concept in all target concepts x with unit embedding $\mathbf{x_i}$, and $\hat{x}_j$ be a source concept in

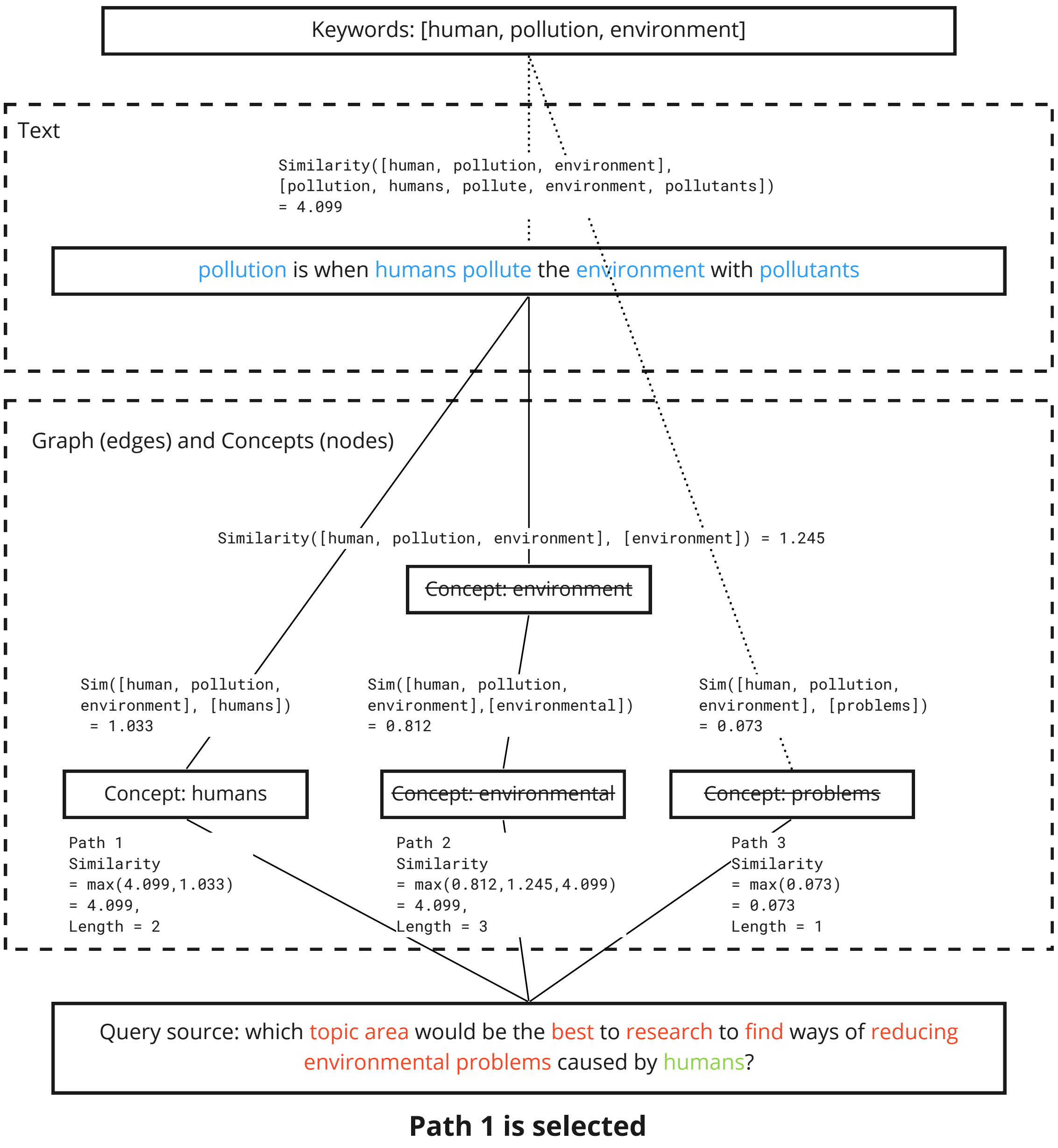

Keywords: [human, pollution, environment]
Text
Similarity([human, pollution, environment], [pollution, humans, pollute, environment, pollutants]) = 4.099
pollution is when humans pollute the environment with pollutants
Graph (edges) and Concepts (nodes)
Similarity([human, pollution, environment], [environment]) = 1.245
~~Concept: environment~~
Sim([human, pollution, environment], [humans]) = 1.033
Sim([human, pollution, environment],[environmental]) = 0.812
Sim([human, pollution, environment], [problems]) = 0.073
Concept: humans
~~Concept: environmental~~
~~Concept: problems~~
Path 1 Similarity = max(4.099,1.033) = 4.099, Length = 2
Path 2 Similarity = max(0.812,1.245,4.099) = 4.099, Length = 3
Path 3 Similarity = max(0.073) = 0.073 Length = 1
Query source: which topic area would be the best to research to find ways of reducing environmental problems caused by humans?
Path 1 is selected

Figure 3.3: Unified knowledge base and retrieval process. (1) Blue words are matched concepts of a composite concept node. (2) Red and green words are concepts in the query source, green words are selected source nodes, red words are not selected because of lower similarity or longer length.

the candidate $\hat{x}$ with unit embedding $\mathbf{\hat{x}_j}$. Two things should be noted here: (1) Usually, each keyword has a corresponding concept in ConceptNet, and x is equivalent to the keyword set. (2) A candidate could be a single concept, or a composite node consisting of multiple concepts. With this definition, we formally define the similarity score as Eq. 3.4.

$$\begin{aligned} Recall &= \frac{\sum_{x_i \in x} tfidf(x_i)\ max_{\hat{x}_j \in \hat{x}} \mathbf{x_i^T} \mathbf{\hat{x}_j}}{\sum_{x_i \in x} tfidf(x_i)} \\ Precision &= \frac{\sum_{\hat{x}_j \in \hat{x}} tfidf(\hat{x}_j)\ max_{x_i \in x} \mathbf{x_i^T} \mathbf{\hat{x}_j}}{\sum_{\hat{x}_j \in \hat{x}} tfidf(\hat{x}_j)} \\ Similarity &= F_\beta(Precision, Recall, \beta = 2) \end{aligned} \tag{3.4}$$

Typically, the $tfidf$ score is computed from the corpus of questions and choices from all splits of the fine-tuning QA dataset to make the matcher adapt to important words better.

The retrieval algorithm performs multiple fixed-depth Monte Carlo tree searches (MCTS) to find the best undirected path. The direction is ignored since sometimes the right path are splitted in half, where source and target node both connects to an intermediate node in an uni-directional way. We also ignore the incomplete word sense annotation in ConceptNet during a search. In each MCTS search, the algorithm picks a concept node in the source sentence as a starting point, and select top-K nodes connected to the current node. The similarity of a generated path is computed by selecting the maximum similarity of all component nodes. The paths are sorted by similarity score first, then sorted by length if their scores are equal.

To better enforce context coherence, we add concepts within a symmetric window around the source concept to the target concepts. Using Fig. 3.3 as an example, when window size is 1, and ”environmental” is selected as source concept, the target concept set becomes: [human, polution, environment] + [reducing, problems].

Since dangling concpet nodes exist in the ConceptNet, whenever nodes have a degree below the specified threshold, we perform another Trie matching and split them into sub nodes. For splitted nodes, similarity scores to the original node are computed, only those with a similarity score above the given minimum will be selected as new nodes.

Algorithm 1 KnowledgeMatcher

```
1: procedure KNOWLEDGEMATCHER($S, M_S, T, M_T, R, D, K, C$)
2:     $\mathcal{S} \leftarrow Split(TrieMatch(S, M_S))$    ▷ Search for concepts in the source sentence.
3:     $\mathcal{T} \leftarrow Split(TrieMatch(T, M_T))$    ▷ Search for concepts in the target sentence.
4:     $\mathcal{P} \leftarrow \emptyset$    ▷ Create the set of all selected paths.
5:     for $i \leftarrow 1$ $to$ $R$ do
6:         $N \in S$    ▷ Select a source node as the current node.
7:         $P \leftarrow [N]$    ▷ Create a new path sequence.
8:         $T \leftarrow \mathcal{T} \cup \{Window(\mathcal{S}, N, C)\}$    ▷ Add context concepts around the starting node.
9:         for $j \leftarrow 1$ $to$ $D$ do
10:            $N \in Top^{K}_{n \in Adj(N)} Similarity(n, T)$    ▷ Select next neighbor from top-K.
11:            $P \leftarrow append(P, N)$
12:        end for
13:        $\mathcal{P} \leftarrow \mathcal{P} \cup \{P\}$
14:    end for
15:    return $RankAndSelect(\mathcal{P}, N)$    ▷ Rank paths by best node similarity, then by length.
16: end procedure
```

We formally define the retrieval procedure in Alg. 1. Meaning of each symbol are listed below. The mask M_S and M_T are used to stop the searcher from using common stopwords as a starting point.

- S, M_S: The source sentence and its mask.
- T, M_T: The target sentence and its mask.

- R: Times to repeat the MCTS search.
- D: Number of steps, or depth, of each MCTS search.
- K: Number of top nodes selected in each step of the MCTS search.
- C: Context window size.
- N: The number of selected paths.

3.5 Implementation of symbolic reasoning

The symbolic reasoning process is based on the forward chaining algorithm shown in Fig. 3.4, in each step, the algorithm selects a valid function to execute, and generate intermediate variables, which can be utilized in the following reasoning steps. Step 3 generates the desired answer "Aluminum", using input variables and one intermediate variable. Step2 generates valid knowledge, but it is redundant. In practice, such redundant steps are eliminated by providing additional hints extracted from the question to the reasoner.

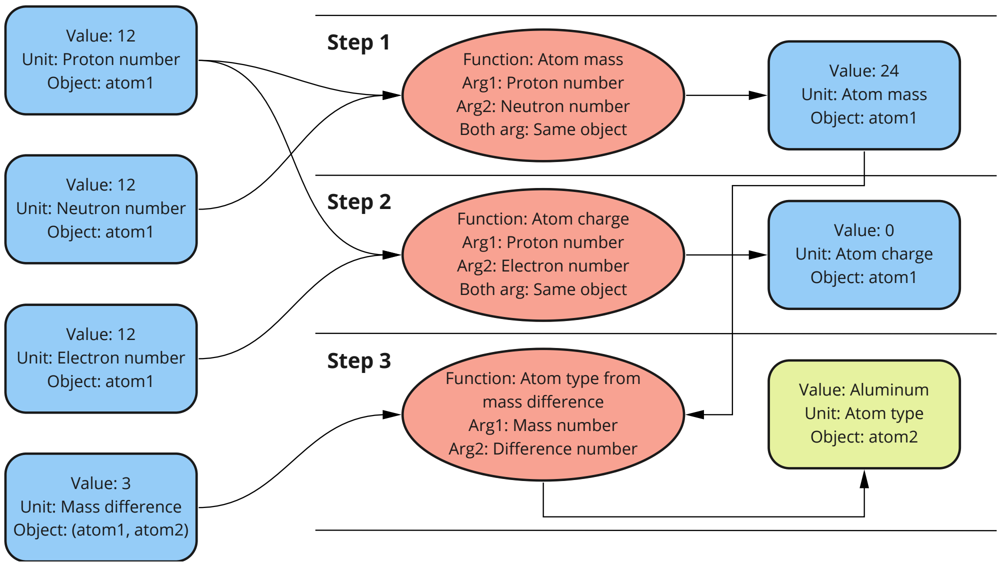


Figure 3.4: An example of the forward chaining algorithm.

The most similar work that resembles our approach is [15], which also use a template-based matching to identify question types from presence of certain terms such as ”what’s the average speed of”, ”how fast”, ”the voltage of”, etc. These hints helps to narrow down the search scope and improves annotation accuracy by removing unnecessary information. We broadly lists several functions and related question triggers supported by the reasoner below.

1. **Speed**: This type of question is identified by the presence of term ”average speed” in the question, and whether two physical quantities, the distance d and time t could be extracted from the query. When the condition is satisfied, function type hint of ”physics-kinematics” is selected.

2. **Time**: This type of question is determined by term ”how long” or the term ”how much time”, as well as two physical quantities, the distance d and speed v in the question. If satisfied, function type hint of ”physics-kinematics” is added.

3. **Acceleration**: If two physical quantities v_1, v_2 of type ”speed” and a time quantity t is detected in the sentence, function type hint of ”physics-kinematics” is selected.

4. **Voltage**: If two physical quantities, current I and resistance R are detected, function type hint of ”physics-electric” is chosen.

5. **Power**: If two physical quantities, current I and voltage U are detected, or current I and resistance R are detected with the presence of term ”power” in question, function type hint of ”physics-electric” is chosen.

CHAPTER 4
EXPERIMENTS

4.1 Data preparation

Three datasets, OpenBookQA, CommonsenseQA and ARC are used to evaluate the performance of our model. For ARC, easy and challenge parts are both used during training, but testing is only conducted on the Challenge portion. We also use the fact annotation from QASC when training our model on the challenge split of ARC. Properties of the used datasets are summarized in Table 4.1.

For OpenBookQA, the keyword searcher only operates on the core facts (1326). We concatenate the provided core facts (1326) with crowd-sourced facts (5167) into one as the corpus indexed by the symbolic matcher. Since several questions in the dataset contain choices requiring logical thinking, such as ”none of these”, ”all of the above”, we remove these choices from the choice set when performing inference, and chain a logical OR/AND operation on the predicted scores afterwards.

For CommonsenseQA, since the dataset doesn't come with a pre-made corpus, we follow the approach used in [79] and create an artificial corpus with each entry being the combination of questions and correct choices in the train dataset. During training, we mask the corresponding entry before performing the symbolic matching to prevent the model from exploiting superficial string equivalence. The searcher is trained on the annotated question concept.

For ARC, the scale and noise of the default corpus becomes a challenge to process. Therefore we reuse the retrieved corpus used by [11] and perform a custom cleaning process to ensure the

Table 4.1: Properties of each dataset.

Dataset	#Train	#Dev	#Test	Corpus size	Annotation?	#Choice
OPENBOOKQA	4957	500	500	1326 + 5167	True	4
COMMONSENSEQA	9741	1221	1140	N/A	Partial	5
ARC (Challenge)	1119 + 5197 (Easy)	299	1172	>15,000,000	False	4/5
QASC	8134	926	920	>17,000,000	True	8

completeness of each entry. A list of detailed operations is provided in Table A.1. We also adds the core fact list. Apart from the cleaned corpus, we also use the crowd-sourced fact list from OPENBOOKQA, and all the annotated facts from the train and validate split of QASC, to create the corpus used by the symbolic matcher. For the keyword searcher, the fact base consists of the same facts from OPENBOOKQA and QASC. This fact has a broad coverage on the grade-school level scientific topics involved in ARC. The keyword searcher is then trained on OPENBOOKQA and QASC to perform the retrieval. Since QASC has two annotated facts, we only use the first fact as the ground truth to prevent noisy retrieval.

4.2 Quality of Generated annotations

One major difference between our approach and other graph traversal approaches is the guidance of a neural model. Although it's also possible to directly extract words that do not belong to stopwords, and has a part-of-speech categorization of a noun, adjective or verb, the outcome is far less ideal. Table 4.2 and Table 4.3 illustrates several annotation examples generated by two methods. The correct choice is colored in blue and facts useful for answering the question are colored in red. We see that **SDAR** is capable of discovering the intermediate knowledge necessary for constructing the complete logical path while matching by comparing to simply select nouns from the question fails to discover the implicit reasoning step.

Table 4.2: Results retrieved by **SDAR**.

SDAR retrieval
George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat? A: dry palms B: wet palms C: palms covered with oil D: palms covered with lotion Annotation: (facts) rubbing your palms together causes friction, a hand dryer produces heat, if heat is conducted to an object then that object will become hot, friction occurs when surf -aces rub together (question) heat related to there is also heat from friction - - rub your hands together and feel the heat : - - (dry palms) dry related to a hand dryer produces heat, dry related to the arrow was designed for high speeds, which means heat from friction (wet palms) wet related to and he felt it on his body, in his hand, and on the soles of his feet: cold, clammy tile, and the chill of shower - water evaporating from his wet skin (palms covered with oil) oil related to as the parts move, friction creates heat which in turn reduces the viscosity of the oil (palms covered with lotion) lotion related to you may want to keep hand lotion handy to help keep your skin from drying out
Which land form is the result of the constructive force of a glacier? A: valleys carved by a moving glacier B: piles of rocks deposited by a melting glacier C: deposition of river sediments D: bedrock hills roughened by the passing of a glacier Annotation: (facts) sometimes piles of rock are formed by melting glaciers depositing rocks (valleys carved by a moving glacier) valleys related to blocks of earth dropped down to form valleys, and the jefferson river eroded a channel through rock to form the jefferson river canyon (piles of rocks deposited by a melting glacier) piles related to sedimentary rock forms when sediment - - such as rock particles or organic matter: become compressed and cemented together as it piles up (bedrock hills roughened by the passing of a glacier) bedrock related to the regolith forms a layer of weathered rock debris which overlies unweathered bedrock and marks the beginning of the soil forming process

Table 4.3: Results retrieved by only use meaningful nouns from the question.

Simple retrieval
George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat? A: dry palms B: wet palms C: palms covered with oil D: palms covered with lotion Annotation: (question) surface related to surface ocean currents, on the other hand, are caused by the wind, heat related to drought heat and drought go hand in hand (dry palms) dry related to a hand dryer produces heat, dry related to don't use the dryers to dry your hands (wet palms) wet related to and he felt it on his body, in his hand, and on the soles of his feet: cold, clammy tile, and the chill of shower - water evaporating from his wet skin (palms covered with oil) oil related to too much heat and the oil burns carbon black (palms covered with lotion) lotion related to you may want to keep hand lotion handy to help keep your skin from drying out
Which land form is the result of the constructive force of a glacier? A: valleys carved by a moving glacier B: piles of rocks deposited by a melting glacier C: deposition of river sediments D: bedrock hills roughened by the passing of a glacier Annotation: (question) land related to as a result, a glacier is formed that covers a great deal of land surface, which is why it is called a continental glacier (valleys carved by a moving glacier) moving related to generally the direction of ocean currents is determined by the effects of surface winds moving surface waters, and the effects of land masses (piles of rocks deposited by a melting glacier) rocks related to topics covered will include origin of earth, plate tectonics, formation of minerals and rocks, mountain building, formation of oceans, continents, etc (grooves created in a granite surface by a glacier) surface related to as a result, a glacier is formed that covers a great deal of land surface, which is why it is called a continental glacier (bedrock hills roughened by the passing of a glacier) bedrock related to as the glaciers scoured this landscape, the mass of bedrock forming the hill proved more resistant than the surrounding soil, forcing the bottom of the glacier up and over the hill

Table 4.4: Accuracy of **SDAR** on OPENBOOKQA.

Split	Top-1 accuracy	Top-3 accuracy	Top-5 accuracy
Validate	61.1	81.9	89.4
Test	60.2	82.1	88.3

We take 50 questions from the validation split of the ARC and COMMONSENSEQA datasets we use to benchmark our model and perform manual inspection of annotated data, we do not select samples containing numerical computation from ARC. For OPENBOOKQA, since the ground truth fact is provided in all splits, we run accuracy testing on the annotations and test whether the ground truth fact is contained in the result. Table 4.4 shows the accuracy of the annotations generated by **SDAR**. For ARC, since it has a high topic overlap with OPENBOOKQA and QASC, 36 out of 50 annotations contains at least one reasonable fact that's related to the question. For COMMONSENSEQA, since commonsense knowledge is implicit, only 25 out of 50 annotations contain useful information helpful for reasoning.

4.3 Contribution of each part of annotation

Since our annotator assembly produce two types of annotation: numerical and factual, and the factual annotation contains three parts: facts, question-related annotation, and answer-related annotation, we dissect the contribution of performance into individual parts in Table 4.5 and Table 4.6 to further analyze the importance of each sub-annotator. For OPENBOOKQA and COMMONSENSEQA, the reported performance gain is the fine-grained contribution of each type of annotation from the factual annotator. The numerical annotator is used on ARC only, and since we have limited resource to train T5-11B, only coarse-grained performance gain from each annotator is reported. We also include the performance of using texts retrieved by the method used in [11] for reference.

Table 4.5: Contribution of each part of annotation to OpenBookQA and CommonsenseQA.

Part	OpenBookQA Val	OpenBookQA Test	CommonsenQA Val
DeBERTa-V3-Large	84.2	83.8	84.6
+Facts	4.0	4.2	N/A
+Only Choice	3.2	3.4	2.9
+Question & Choice	2.0	0.2	3.4
Total	90.2	91.6	88.0

Table 4.6: Contribution of each annotator to ARC.

Part	ARC Val	ARC Test
UnifiedQA-v2-11B	77.6	77.4
+Factual annotator	4.7	4.7
+Numerical annotator	2.3	2.1
Total	84.6	84.2
+UnifiedQA IR	N/A	3.75
Total	N/A	81.1

Table 4.7: Performance of our model on all three datasets.

Dataset	Split	Model	Parameters	Accuracy
OpenBookQA	Test	GenMC (ensemble)	11B * 7	92.0
OpenBookQA	Test	GenMC	11B	89.8
OpenBookQA	Test	**SDAR**	435M	91.6
ARC	Test	ST-MoE-32B	269B	86.5
ARC	Test	**SDAR**	11B	84.2
CommonsenseQA	Val	KEAR (ensemble)	435M * 39	93.4
CommonsenseQA	Val	KEAR	435M	91.2
CommonsenseQA	Val	**SDAR**	435M	88.0

4.4 Comparison to other models

We are most interested in the efficiency of our method compared to simply boosting models to a larger size and achieve performance gain. In Table 4.7, we list the performance and parameter number of state-of-the-art models. We have successfully achieved the new STOA single model performance on OPENBOOKQA, and a decent performance gain on ARC. On COMMONSENSEQA we are not on-par with the current STOA model probably because our annotator does not include the distributional information of graph edges used in [79]. In general our method can substantially boost the performance of base models while still working on a smaller parameter size, this indicates that models can leverage their limited internal knowledge to leverage the information generated by external symbolic methods and enhance their reasoning capabilities.

CHAPTER 5
CONCLUSION AND FUTURE WORK

5.1 Summary of key findings and significance

We introduced the **SDAR** framework, which provides an efficient means of knowledge annotation that incorporates most relevant information through various symbolic methods into the reasoning process conducted by large neural-based language models. Most importantly, the generated annotation is human-readable and provides an easy way to interpret framework behavior and check for errors. The framework also have a decent degree of freedom which makes it applicable to datasets with no ground truth reasoning annotations, such as ARC and COMMONSENSEQA.

Our experiments demonstrate the power of **SDAR**, showing that it could reach a performance comparable to much larger models. This property is crucial since current researches need to find a way to avoid limiting most advanced models only to those with enormous computation resources.

5.2 Limitations and opportunities for future research

There are several perceivable weaknesses in this work. Firstly, the factual annotator depends on indexing a core set of facts to generate keywords, which is not available when there is no source of explicit representations (Eg: For question ”Is is more likely to have a better cafeteria in a polytechnic or a high school?”, we know that polytechnics are richer and therefore has a larger possibility of owning a better cafeteria, this is implicit commonsense). In this case, using reinforcement learning may be a better way to find such hidden indications. Secondly, the coherence between nodes in the generated path is not guaranteed, this problem can be solved by introducing the global coherence penalty used in [67]. The last problem is that the unparameterized numerical annotator front-end is not able to dynamically process unseen patterns, or complex patterns. The neural based extraction solution used in [43] could be a much more robust replacement.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. arXiv: `1706.03762`.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. arXiv: `1810.04805`.

[3] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[4] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[5] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *CoRR*, vol. abs/1906.08237, 2019. arXiv: `1906.08237`.

[6] Y. Gu, R. Tinn, H. Cheng, M. Lucas, N. Usuyama, X. Liu, T. Naumann, J. Gao, and H. Poon, "Domain-specific language model pretraining for biomedical natural language processing," *CoRR*, vol. abs/2007.15779, 2020. arXiv: `2007.15779`.

[7] W. Tai, H. T. Kung, X. Dong, M. Comiter, and C.-F. Kuo, "ExBERT: Extending pre-trained models with domain-specific vocabulary under constrained training resources," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, Online: Association for Computational Linguistics, Nov. 2020, pp. 1433–1439.

[8] J. Yan, M. Raman, T. Zhang, R. A. Rossi, H. Zhao, S. Kim, N. Lipka, and X. Ren, "Learning contextualized knowledge structures for commonsense reasoning," *CoRR*, vol. abs/2010.12873, 2020. arXiv: `2010.12873`.

[9] W. Liu, P. Zhou, Z. Zhao, Z. Wang, Q. Ju, H. Deng, and P. Wang, "K-bert: Enabling language representation with knowledge graph," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 2901–2908.

[10] M. Ostendorff, P. Bourgonje, M. Berger, J. Moreno-Schneider, G. Rehm, and B. Gipp, “Enriching bert with knowledge graph embeddings for document classification,” *arXiv preprint arXiv:1909.08402*, 2019.

[11] D. Khashabi, T. Khot, A. Sabharwal, O. Tafjord, P. Clark, and H. Hajishirzi, “Unifiedqa: Crossing format boundaries with a single QA system,” *CoRR*, vol. abs/2005.00700, 2020. arXiv: `2005.00700`.

[12] Z. Huang, A. Wu, J. Zhou, Y. Gu, Y. Zhao, and G. Cheng, *Clues before answers: Generation-enhanced multiple-choice qa*.

[13] C. Bhagavatula, R. L. Bras, C. Malaviya, K. Sakaguchi, A. Holtzman, H. Rashkin, D. Downey, S. W.-t. Yih, and Y. Choi, “Abductive commonsense reasoning,” *CoRR*, vol. abs/1908.05739, 2019. arXiv: `1908.05739`.

[14] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. H. Chi, Q. Le, and D. Zhou, “Chain of thought prompting elicits reasoning in large language models,” *CoRR*, vol. abs/2201.11903, 2022. arXiv: `2201.11903`.

[15] T. Khot, D. Khashabi, K. Richardson, P. Clark, and A. Sabharwal, “Text modular networks: Learning to decompose tasks in the language of existing models,” *CoRR*, vol. abs/2009.00751, 2020. arXiv: `2009.00751`.

[16] E. Perez, P. S. H. Lewis, W.-t. Yih, K. Cho, and D. Kiela, “Unsupervised question decomposition for question answering,” *CoRR*, vol. abs/2002.09758, 2020. arXiv: `2002.09758`.

[17] N. Kertkeidkachorn and R. Ichise, “T2kg: An end-to-end system for creating knowledge graph from unstructured text,” in *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[18] R. Koncel-Kedziorski, D. Bekal, Y. Luan, M. Lapata, and H. Hajishirzi, “Text generation from knowledge graphs with graph transformers,” *CoRR*, vol. abs/1904.02342, 2019. arXiv: `1904.02342`.

[19] P. Ke, H. Ji, Y. Ran, X. Cui, L. Wang, L. Song, X. Zhu, and M. Huang, “Jointgt: Graph-text joint representation learning for text generation from knowledge graphs,” *arXiv preprint arXiv:2106.10502*, 2021.

[20] T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal, “Can a suit of armor conduct electricity? A new dataset for open book question answering,” *CoRR*, vol. abs/1809.02789, 2018. arXiv: `1809.02789`.

[21] A. Talmor, J. Herzig, N. Lourie, and J. Berant, “Commonsenseqa: A question answering challenge targeting commonsense knowledge,” *CoRR*, vol. abs/1811.00937, 2018. arXiv: `1811.00937`.

[22] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord, “Think you have solved question answering? try arc, the ai2 reasoning challenge,” *ArXiv*, vol. abs/1803.05457, 2018.

[23] V. Mnih, N. Heess, A. Graves, *et al.*, “Recurrent models of visual attention,” *Advances in neural information processing systems*, vol. 27, 2014.

[24] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *CoRR*, vol. abs/1910.10683, 2019. arXiv: `1910.10683`.

[25] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “ALBERT: A lite BERT for self-supervised learning of language representations,” *CoRR*, vol. abs/1909.11942, 2019. arXiv: `1909.11942`.

[26] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, “Language models are unsupervised multitask learners,” *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[27] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, “ELECTRA: pre-training text encoders as discriminators rather than generators,” *CoRR*, vol. abs/2003.10555, 2020. arXiv: `2003.10555`.

[28] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter,” *CoRR*, vol. abs/1910.01108, 2019. arXiv: `1910.01108`.

[29] P. He, X. Liu, J. Gao, and W. Chen, “Deberta: Decoding-enhanced BERT with disentangled attention,” *CoRR*, vol. abs/2006.03654, 2020. arXiv: `2006.03654`.

[30] G. Ke, D. He, and T.-Y. Liu, “Rethinking positional encoding in language pre-training,” *CoRR*, vol. abs/2006.15595, 2020. arXiv: `2006.15595`.

[31] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. V. Le, G. E. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," *CoRR*, vol. abs/1701.06538, 2017. arXiv: `1701.06538`.

[32] Y. Sun, S. Wang, S. Feng, S. Ding, C. Pang, J. Shang, J. Liu, X. Chen, Y. Zhao, Y. Lu, *et al.*, "Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation," *arXiv preprint arXiv:2107.02137*, 2021.

[33] B. Zoph, I. Bello, S. Kumar, N. Du, Y. Huang, J. Dean, N. Shazeer, and W. Fedus, *St-moe: Designing stable and transferable sparse expert models*, 2022.

[34] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *CoRR*, vol. abs/2101.03961, 2021. arXiv: `2101.03961`.

[35] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big?" In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, ser. FAccT '21, Virtual Event, Canada: Association for Computing Machinery, 2021, pp. 610–623, ISBN: 9781450383097.

[36] F. Petroni, T. Rocktäschel, P. S. H. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel, "Language models as knowledge bases?" *CoRR*, vol. abs/1909.01066, 2019. arXiv: `1909.01066`.

[37] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig, "How can we know what language models know?" *CoRR*, vol. abs/1911.12543, 2019. arXiv: `1911.12543`.

[38] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," *ArXiv*, vol. abs/1703.04730, 2017.

[39] R. Speer, J. Chin, and C. Havasi, "Conceptnet 5.5: An open multilingual graph of general knowledge," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, ser. AAAI'17, San Francisco, California, USA: AAAI Press, 2017, pp. 4444–4451.

[40] M. Sap, R. L. Bras, E. Allaway, C. Bhagavatula, N. Lourie, H. Rashkin, B. Roof, N. A. Smith, and Y. Choi, "ATOMIC: an atlas of machine commonsense for if-then reasoning," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, AAAI Press, 2019, pp. 3027–3035.

[41] C. Matuszek, J. Cabral, M. Witbrock, and J. Deoliveira, “An introduction to the syntax and content of cyc,” in *Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, 2006, pp. 44–49.

[42] K. D. Forbus and T. Hinrich, “Analogy and relational representations in the companion cognitive architecture,” *AI Magazine*, vol. 38, no. 4, pp. 34–42, 2017.

[43] P. Lu, R. Gong, S. Jiang, L. Qiu, S. Huang, X. Liang, and S.-C. Zhu, “Inter-gps: Interpretable geometry problem solving with formal language and symbolic reasoning,” *CoRR*, vol. abs/2105.04165, 2021. arXiv: 2105.04165.

[44] H. Li, J. Zhu, J. Zhang, C. Zong, and X. He, “Keywords-guided abstractive sentence summarization,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 8196–8203.

[45] K. Faldu, A. P. Sheth, P. Kikani, and H. Akabari, “KI-BERT: infusing knowledge context for better language and domain understanding,” *CoRR*, vol. abs/2104.08145, 2021. arXiv: 2104.08145.

[46] P. Clark, O. Etzioni, D. Khashabi, T. Khot, B. D. Mishra, K. Richardson, A. Sabharwal, C. Schoenick, O. Tafjord, N. Tandon, S. Bhakthavatsalam, D. Groeneveld, M. Guerquin, and M. Schmitz, “From ’f’ to ’a’ on the N.Y. regents science exams: An overview of the aristo project,” *CoRR*, vol. abs/1909.01958, 2019. arXiv: 1909.01958.

[47] D. Chen, A. Fisch, J. Weston, and A. Bordes, “Reading wikipedia to answer open-domain questions,” *CoRR*, vol. abs/1704.00051, 2017. arXiv: 1704.00051.

[48] J. Lee, S. Yun, H. Kim, M. Ko, and J. Kang, “Ranking paragraphs for improving answer recall in open-domain question answering,” *CoRR*, vol. abs/1810.00494, 2018. arXiv: 1810.00494.

[49] B. Kratzwald, A. Eigenmann, and S. Feuerriegel, “Rankqa: Neural question answering with answer re-ranking,” *CoRR*, vol. abs/1906.03008, 2019. arXiv: 1906.03008.

[50] B. Kratzwald and S. Feuerriegel, “Adaptive document retrieval for deep question answering,” *CoRR*, vol. abs/1808.06528, 2018. arXiv: 1808.06528.

[51] Z. Wang, P. Ng, X. Ma, R. Nallapati, and B. Xiang, "Multi-passage BERT: A globally normalized BERT model for open-domain question answering," *CoRR*, vol. abs/1908.08167, 2019. arXiv: `1908.08167`.

[52] M. Ghazvininejad, C. Brockett, M.-W. Chang, B. Dolan, J. Gao, W.-t. Yih, and M. Galley, "A knowledge-grounded neural conversation model," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[53] F. Zhu, W. Lei, C. Wang, J. Zheng, S. Poria, and T.-S. Chua, "Retrieving and reading: A comprehensive survey on open-domain question answering," *CoRR*, vol. abs/2101.00774, 2021. arXiv: `2101.00774`.

[54] V. Karpukhin, B. Oguz, S. Min, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," *CoRR*, vol. abs/2004.04906, 2020. arXiv: `2004.04906`.

[55] K. Nishida, I. Saito, A. Otsuka, H. Asano, and J. Tomita, "Retrieve-and-read: Multi-task learning of information retrieval and reading comprehension," *CoRR*, vol. abs/1808.10628, 2018. arXiv: `1808.10628`.

[56] R. Agarwal, A. Koniaev, and R. Schaefer, "Exploring argument retrieval for controversial questions using retrieve and re-rank pipelines," *Working Notes of CLEF*, 2021.

[57] O. Khattab, C. Potts, and M. Zaharia, "Relevance-guided supervision for openqa with colbert," *CoRR*, vol. abs/2007.00814, 2020. arXiv: `2007.00814`.

[58] W. Yu, C. Zhu, Z. Li, Z. Hu, Q. Wang, H. Ji, and M. Jiang, "A survey of knowledge-enhanced text generation," *CoRR*, vol. abs/2010.04389, 2020. arXiv: `2010.04389`.

[59] J. Guan, F. Huang, Z. Zhao, X. Zhu, and M. Huang, "A knowledge-enhanced pretraining model for commonsense story generation," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 93–108, 2020.

[60] K. Wang, Y. Zhang, D. Yang, L. Song, and T. Qin, "GNN is a counter? revisiting GNN for question answering," *CoRR*, vol. abs/2110.03192, 2021. arXiv: `2110.03192`.

[61] W. Xiong, T. Hoang, and W. Y. Wang, "Deeppath: A reinforcement learning method for knowledge graph reasoning," *CoRR*, vol. abs/1707.06690, 2017. arXiv: `1707.06690`.

[62] N. Lao, T. Mitchell, and W. Cohen, "Random walk inference and learning in a large scale knowledge base," in *Proceedings of the 2011 conference on empirical methods in natural language processing*, 2011, pp. 529–539.

[63] P. Wang, N. Peng, P. A. Szekely, and X. Ren, "Connecting the dots: A knowledgeable path generator for commonsense question answering," *CoRR*, vol. abs/2005.00691, 2020. arXiv: `2005.00691`.

[64] Z. Liu, Z.-Y. Niu, H. Wu, and H. Wang, "Knowledge aware conversation generation with reasoning on augmented graph," *CoRR*, vol. abs/1903.10245, 2019. arXiv: `1903.10245`.

[65] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. J. Smola, and A. McCallum, "Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning," *CoRR*, vol. abs/1711.05851, 2017. arXiv: `1711.05851`.

[66] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013. arXiv: `1312.5602`.

[67] J. Xu, H. Wang, Z.-Y. Niu, H. Wu, W. Che, and T. Liu, "Conversational graph grounded policy learning for open-domain conversation generation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 1835–1845.

[68] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in neural information processing systems*, vol. 26, 2013.

[69] M. I. Nye, A. J. Andreassen, G. Gur-Ari, H. Michalewski, J. Austin, D. Bieber, D. Dohan, A. Lewkowycz, M. Bosma, D. Luan, C. Sutton, and A. Odena, "Show your work: Scratchpads for intermediate computation with language models," *CoRR*, vol. abs/2112.00114, 2021. arXiv: `2112.00114`.

[70] S. Park, J. R. Chowdhury, T. Kundu, and C. Caragea, "Kpdrop: An approach to improving absent keyphrase generation," *CoRR*, vol. abs/2112.01476, 2021. arXiv: `2112.01476`.

[71] R. Meng, X. Yuan, T. Wang, S. Zhao, A. Trischler, and D. He, "An empirical study on neural keyphrase generation," *CoRR*, vol. abs/2009.10229, 2020. arXiv: `2009.10229`.

[72] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[73] X. Ling and D. S. Weld, "Fine-grained entity recognition," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[74] F. Dernoncourt, J. Y. Lee, and P. Szolovits, "Neuroner: An easy-to-use program for named-entity recognition based on neural networks," *arXiv preprint arXiv:1705.05487*, 2017.

[75] C. Liang, Y. Yu, H. Jiang, S. Er, R. Wang, T. Zhao, and C. Zhang, "BOND: BERT-assisted open-domain named entity recognition with distant supervision," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery &amp Data Mining*, ACM, Aug. 2020.

[76] W. Bibel, *Automated theorem proving*. Springer Science & Business Media, 2013.

[77] J. Kvarnström and P. Doherty, "Talplanner: A temporal logic based forward chaining planner," *Annals of mathematics and Artificial Intelligence*, vol. 30, no. 1, pp. 119–169, 2000.

[78] A. Coles, A. Coles, M. Fox, and D. Long, "Forward-chaining partial-order planning," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 20, 2010, pp. 42–49.

[79] Y. Xu, C. Zhu, S. Wang, S. Sun, H. Cheng, X. Liu, J. Gao, P. He, M. Zeng, and X. Huang, "Human parity on commonsenseqa: Augmenting self-attention with external attention," *CoRR*, vol. abs/2112.03254, 2021. arXiv: `2112.03254`.

[80] D. Chen and C. D. Manning, "A fast and accurate dependency parser using neural networks," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 740–750.

[81] J. Nivre, M.-C. de Marneffe, F. Ginter, Y. Goldberg, J. Hajič, C. D. Manning, R. McDonald, S. Petrov, S. Pyysalo, N. Silveira, R. Tsarfaty, and D. Zeman, "Universal Dependencies v1: A multilingual treebank collection," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, Portorož, Slovenia: European Language Resources Association (ELRA), May 2016, pp. 1659–1666.

[82] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with BERT," *CoRR*, vol. abs/1904.09675, 2019. arXiv: `1904.09675`.

SYMBOLIC DATA AUGMENTATION FOR ASSISTED NEURAL REASONING

Approved by:

Douglas Downey
Computer Science
Northwestern

David Demeter
Computer Science
Northwestern

Chris Riesbeck
Computer Science
Northwestern

Date Approved: May 12, 2022

APPENDIX A

APPENDIX

Table A.1: Steps used to clean the corpus retrieved by [11].

Steps	Adds / Removes entries?
Sentence tokenize using nltk	Add
Cleaning using cleantext	No
Remove html characters	No
Remove parenthesis/brackets/bracelets at start	No
Remove non-alphabetical symbols at start	No
Remove non-alphanumerical symbols at end	No
Remove escapes and repeated special symbols	No
Remove sentences ends with question marks	Remove
Remove sentences with multiple choices (Eg: Question-ChoiceA-ChoiceB-...)	Remove
Remove sentences with a high ratio of non-english word tokens	Remove
Remove short sentences	Remove
Remove incomplete sentences using the ”textattack/roberta-base-CoLA” model.	Remove
Remove duplicates	Remove
Remove parenthesis that looks like an annotation (Parenthesis used in chemical questions are not)	No
Remove incomplete parenthesis/ brackets/bracelets using a stack parser	No