AdaLoGN: Adaptive Logic Graph Network for Reasoning-Based Machine Reading Comprehension

Xiao Li and **Gong Cheng** and **Ziheng Chen** and **Yawei Sun** and **Yuzhong Qu**
State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
{xiaoli.nju,chenziheng,ywsun}@smail.nju.edu.cn
{gcheng,yzqu}@nju.edu.cn

Abstract

Recent machine reading comprehension datasets such as ReClor and LogiQA require performing logical reasoning over text. Conventional neural models are insufficient for logical reasoning, while symbolic reasoners cannot directly apply to text. To meet the challenge, we present a neural-symbolic approach which, to predict an answer, passes messages over a graph representing logical relations between text units. It incorporates an adaptive logic graph network (AdaLoGN) which adaptively infers logical relations to extend the graph and, essentially, realizes mutual and iterative reinforcement between neural and symbolic reasoning. We also implement a novel subgraph-to-node message passing mechanism to enhance context-option interaction for answering multiple-choice questions. Our approach shows promising results on ReClor and LogiQA.

1 Introduction

Machine reading comprehension (MRC) has drawn much research attention. Early MRC datasets are not difficult for state-of-the-art neural methods. Indeed, BERT (Devlin et al., 2019) has outperformed humans on SQuAD (Rajpurkar et al., 2016). Recent datasets become more challenging. For example, ReClor (Yu et al., 2020) and LogiQA (Liu et al., 2020) require understanding and reasoning over logical relations described in text, where neural methods showed unsatisfactory performance.

For instance, consider the MRC task in Figure 1. The context consists of a set of textual propositions describing logical relations between elementary discourse units (EDUs) (Mann and Thompson, 1988). For example, the first sentence describes an implication between two EDUs: "the company gets project A" implies that "product B can be put on the market on schedule". With the help of propositional calculus, humans can formalize propositions and then apply inference rules in propositional logic to prove the proposition in option C. However, how can machines solve such a task?

Context: *If* the company gets project A, product B can be put on the market on schedule. Product B is put on schedule *if and only if* the company's fund can be normally turned over. *If* the company's fund *cannot* be turned over normally, the development of product C *cannot* be carried out as scheduled. The fact is that the development of product C is carried out as scheduled.

Question: This shows:

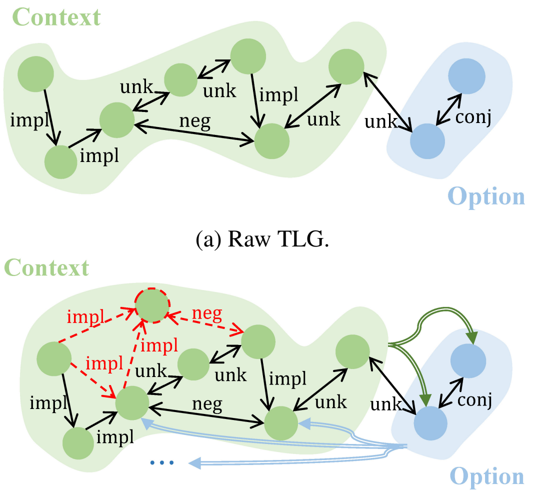
Options:
A. The company gets project A and product B is put on the market on schedule.
B. The company does not get project A and product B is not put on the market on schedule.
✓**C.** Product B is put on the market on schedule and the company's fund is turned over normally.
D. Product B is not put on the market on schedule, and the company's fund turnover is extremely abnormal.

Figure 1: An example MRC task (adapted from a task in LogiQA). Logical connectives are highlighted in italics. ✓marks the correct answer.

Existing Methods and Limitations To solve it, conventional neural models are insufficient for providing the required reasoning capabilities, while symbolic reasoners cannot directly apply to unstructured text. One promising direction is to consider a neural-symbolic solution, such as the recent DAGN method (Huang et al., 2021a). It breaks down the context and each option into a set of EDUs and connects them with discourse relations as a graph. Then it performs graph neural network (GNN) based reasoning to predict an answer.

However, we identify two limitations in this method. **L1:** Despite the graph representation, it is predominantly a neural method over discourse relations. It is debatable whether the required symbolic reasoning over logical relations (e.g., implication, negation) can be properly approximated. **L2:** The graph is often loosely connected and composed of long paths. Node-to-node message passing implemented in existing GNN models (Kipf and Welling, 2017; Schlichtkrull et al., 2018; Velickovic et al., 2018) is prone to provide insufficient interaction be-

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 7147 - 7161
May 22-27, 2022 ©2022 Association for Computational Linguistics



(a) Raw TLG.

(b) Extended TLG. Dashed nodes and edges represent adaptively inferred EDUs and logical relations, respectively. Double edges represent subgraph-to-node message passing.

Figure 2: Two TLGs for exemplifying our approach. For readability, we omit `rev` edges.

tween the context and the option, which is critical to answering a multiple-choice question.

Our Approach. While we follow the general framework of DAGN, i.e., graph construction and then graph-based reasoning, we overcome its two limitations with a novel neural-symbolic approach.

To address L1, Figure 3 sketches out our idea. Specifically, we propose to construct a text logic graph (TLG) representing EDUs and their logical relations as opposed to discourse relations, so we can *explicitly perform symbolic reasoning* to extend the TLG with inferred logical relations, as illustrated in Figure 2. The inferred relations may provide crucial connections to be used in the subsequent graph-based message passing, i.e., *symbolic reasoning reinforces neural reasoning*.

Further, while trivially computing and admitting the deductive closure may extend the TLG with irrelevant connections which would mislead message passing, we leverage signals from neural reasoning to adaptively admit relevant extensions, i.e., *neural reasoning reinforces symbolic reasoning*.

Moreover, we *iterate the above mutual reinforcement* by restarting inference in each iteration with signals from the previous iteration to accommodate corrections to the reasoning process and allow sufficient neural-symbolic interaction.

To address L2, we aggregate the information in the context subgraph of TLG and employ a novel subgraph-to-node message passing mechanism to *enhance the interaction from the holistic context subgraph to each node in the option subgraph, and vice versa*, as illustrated in Figure 2b.

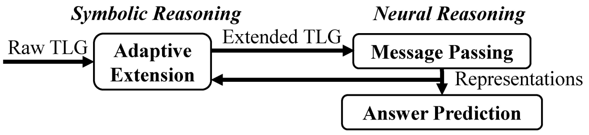


Figure 3: Our main idea: mutual and iterative reinforcement between symbolic and neural reasoning.

We incorporate the above two ideas into our new Adaptive Logic Graph Network (AdaLoGN). To summarize, our technical contributions include

- a novel neural-symbolic approach where neural and symbolic reasoning mutually and iteratively reinforce each other, and
- a novel aggregation-based enhancement of message passing in graph-based neural reasoning.

Outline. We elaborate our approach in Section 2, present experiments in Section 3, discuss related work in Section 4, and conclude in Section 5.

Our code is available on GitHub: `https://github.com/nju-websoft/AdaLoGN`.

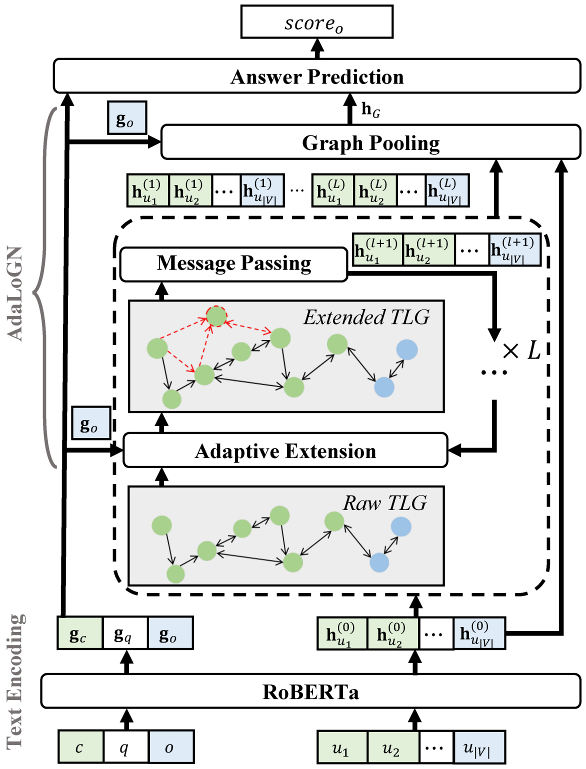
2 Approach

A MRC task $\langle c, q, O \rangle$ consists of a context c, a question q, and a set of options O. Only one option in O is the correct answer to q given c. The goal of the task is to find this option.

Figure 4 outlines our implementation. For each option $o \in O$, we generate the representations of c, q, o (i.e., $\mathbf{g}_c, \mathbf{g}_q, \mathbf{g}_o$, respectively) by a pre-trained language model (Section 2.1), and we construct a raw TLG where nodes (i.e., $u_1, \ldots, u_{|V|}$) represent EDUs extracted from c, q, o and edges represent their logical relations (Section 2.2). With their initial representations (i.e., $\mathbf{h}_{u_1}^{(0)}, \ldots, \mathbf{h}_{u_{|V|}}^{(0)}$) obtained from the pre-trained language model, in an iterative manner, we adaptively extend the TLG (i.e., symbolic reasoning) and then pass messages (i.e., neural reasoning) to update node representations (i.e., $\mathbf{h}_{u_1}^{(l+1)}, \ldots, \mathbf{h}_{u_{|V|}}^{(l+1)}$) for generating the representation of the TLG (i.e., $\mathbf{h}_G$) (Section 2.3). Finally, we predict the correctness of o (i.e., $score_o$) based on the above representations (Section 2.4).

2.1 Text Encoding

We use RoBERTa (Liu et al., 2019), a pre-trained language model, to encode three token sequences $c = c_1 \cdots c_{|c|}$, $q = q_1 \cdots q_{|q|}$, and $o = o_1 \cdots o_{|o|}$ which are concatenated by the classifier token <s>



$score_o$
Answer Prediction
$\mathbf{g}_o$
$\mathbf{h}_G$
Graph Pooling
$\mathbf{h}_{u_1}^{(1)}$ $\mathbf{h}_{u_2}^{(1)}$ ··· $\mathbf{h}_{u_{|V|}}^{(1)}$ ··· $\mathbf{h}_{u_1}^{(L)}$ $\mathbf{h}_{u_2}^{(L)}$ ··· $\mathbf{h}_{u_{|V|}}^{(L)}$
Message Passing
$\mathbf{h}_{u_1}^{(l+1)}$ $\mathbf{h}_{u_2}^{(l+1)}$ ··· $\mathbf{h}_{u_{|V|}}^{(l+1)}$
Extended TLG
× L
AdaLoGN
$\mathbf{g}_o$
Adaptive Extension
Raw TLG
$\mathbf{g}_c$ $\mathbf{g}_q$ $\mathbf{g}_o$
$\mathbf{h}_{u_1}^{(0)}$ $\mathbf{h}_{u_2}^{(0)}$ ··· $\mathbf{h}_{u_{|V|}}^{(0)}$
Text Encoding
RoBERTa
c q o
u_1 u_2 ··· $u_{|V|}$

Figure 4: Overview of our approach.

and the separator token </s>:

$$[\mathbf{g}_{\text{<s>}}; \mathbf{g}_{c_1}; \ldots; \mathbf{g}_{\text{</s>}}; \mathbf{g}_{q_1}; \ldots; \mathbf{g}_{o_1}; \ldots; \mathbf{g}_{\text{</s>}}] = \text{RoBERTa}(\text{<s>}\ c_1 \cdots \text{</s>}\ q_1 \cdots o_1 \cdots \text{</s>})\,. \tag{1}$$

The output vector representations are averaged to form the representations of c, q, o:

$$\mathbf{g}_c = \frac{1}{|c|}\sum_{i=1}^{|c|} \mathbf{g}_{c_i}\,,\ \mathbf{g}_q = \frac{1}{|q|}\sum_{i=1}^{|q|} \mathbf{g}_{q_i}\,,\ \mathbf{g}_o = \frac{1}{|o|}\sum_{i=1}^{|o|} \mathbf{g}_{o_i}\,. \tag{2}$$

2.2 Text Logic Graph (TLG)

Besides directly encoding text, we extract logical relations from text as a graph called TLG.

2.2.1 Definition of TLG

For a piece of text, its TLG is a directed graph $G = \langle V, E \rangle$ where V is a set of nodes representing EDUs of the text (Mann and Thompson, 1988), and $E \subseteq V \times R \times V$ is a set of labeled directed edges representing *logical relations* between EDUs described in the text. We consider six types of common logical relations $R = \{\texttt{conj}, \texttt{disj}, \texttt{impl}, \texttt{neg}, \texttt{rev}, \texttt{unk}\}$:

- conjunction (conj), disjunction (disj), implication (impl), and negation (neg) are standard logical connectives in propositional logic;

Rhetorical Relation	Logical Relation
LIST, CONTRAST	conj
DISJUNCTION	disj
RESULT	impl
CAUSE, PURPOSE, CONDITION, BACKGROUND	rev

Table 1: Mapping from rhetorical relations in Graphene to logical relations in TLG.

- reversed implication (rev) is introduced to represent the inverse relation of impl;
- unk represents an unknown relation.

Since conj, disj, neg, and unk are symmetric relations, edges labeled with them are bidirectional.

Observe the difference between our TLG and the discourse-based logic graph considered in DAGN (Huang et al., 2021a): edges in the former represent logical relations, while those in the latter represent discourse relations. Therefore, we can explicitly perform symbolic reasoning on TLG.

2.2.2 Construction of Raw TLG

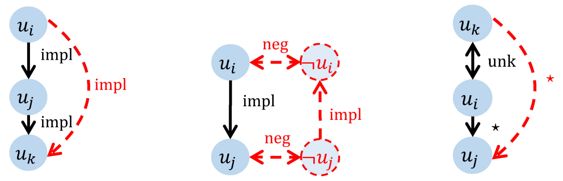
We initialize a raw TLG from c and o. Following Huang et al. (2021a), we ignore q as it is usually uninformative in existing datasets. Specifically, we use Graphene (Cetto et al., 2018) to extract EDUs and their rhetorical relations (Mann and Thompson, 1988) from c and o. Rhetorical relations are converted to logical relations via the mapping in Table 1. Note that each impl edge is always paired with an inverse rev edge, and vice versa.

We also define a small number of syntactic rules to identify EDUs that negate each other and connect them with neg. The rules are based on part-of-speech tags and dependencies. For example, one such rule checks whether two EDUs differ from each other only by an antonym of an adverb.

In addition, for each pair of EDUs that are adjacent in the text (including the last EDU of c and the first EDU of o) but have none of the above logical relations, we connect them with unk because Graphene may fail to identify their relation.

2.3 Adaptive Logic Graph Network (AdaLoGN)

Since TLG consists of logical relations, we explicitly perform symbolic reasoning by applying inference rules to extend the TLG with inferred logical relations to benefit the subsequent neural reasoning. However, rather than computing the deductive closure which may undesirably provide many relations that are irrelevant to answering the question



(a) Hypothetical syllogism. (b) Transposition. (c) Adjacency-transmission.

Figure 5: Dashed nodes and edges are inferred by applying an inference rule. $\star$ represents any logical relation in $\{\texttt{conj}, \texttt{disj}, \texttt{impl}\}$. We omit rev edges.

and mislead neural reasoning, we perform adaptive extension by leveraging signals from neural reasoning to identify and admit relevant extensions. For neural reasoning, we perform message passing to update node representations, which finally are pooled into the representation of the TLG to be used in the subsequent answer prediction. We iterate the above process by restarting inference on the raw TLG in each iteration with signals from the previous iteration to accommodate corrections to the reasoning process and let symbolic and neural reasoning sufficiently interact with each other. We transform the above idea into a new model named AdaLoGN outlined in Figure 4 and detailed below.

2.3.1 Inference Rules

Let $G = \langle V, E \rangle$ be a raw TLG. For symbolic reasoning over the logical relations in G, we apply two *inference rules about implication in propositional logic*. Other rules are left for future work.

- Hypothetical Syllogism:

$$((u_i \rightarrow u_j) \wedge (u_j \rightarrow u_k)) \vdash (u_i \rightarrow u_k)\,. \quad (3)$$

  Specifically, if E contains two edges $\langle u_i, \texttt{impl}, u_j \rangle$ and $\langle u_j, \texttt{impl}, u_k \rangle$, we can add two edges $\langle u_i, \texttt{impl}, u_k \rangle$ and $\langle u_k, \texttt{rev}, u_i \rangle$ to E, as illustrated in Figure 5a.

- Transposition:

$$(u_i \rightarrow u_j) \vdash (\neg u_j \rightarrow \neg u_i)\,. \quad (4)$$

  Specifically, if E contains an edge $\langle u_i, \texttt{impl}, u_j \rangle$, we can add two edges $\langle \neg u_j, \texttt{impl}, \neg u_i \rangle$ and $\langle \neg u_i, \texttt{rev}, \neg u_j \rangle$ to E, as illustrated in Figure 5b. Note that if u_i (resp. u_j) is not incident from/to any neg edge, i.e., $\neg u_i$ (resp. $\neg u_j$) is not a node in V, we will add $\neg u_i$ (resp. $\neg u_j$) to V whose text negates that of u_i (resp. u_j), and then add a bidirectional neg edge between u_i and $\neg u_i$ (resp. u_j and $\neg u_j$) to E.

Besides, recall that unk represents a potential logical relation between EDUs that are adjacent in text. Considering that an EDU often inherits logical relations from its adjacent EDUs, we heuristically define and apply the following inference rule.

- Adjacency-Transmission:

$$((u_i \star u_j) \wedge (u_i \sim u_k)) \vdash (u_k \star u_j)\,, \quad (5)$$

  where $\star \in \{\wedge, \vee, \rightarrow\}$ and $\sim$ represents adjacency in text. For example, if E contains two edges $\langle u_i, \texttt{conj}, u_j \rangle$ and $\langle u_i, \texttt{unk}, u_k \rangle$, we can add a bidirectional conj edge between u_k and u_j to E, as illustrated in Figure 5c.

While this rule may generate false propositions, we expect our adaptive reasoner to apply it properly. For example, it is useful for handling the following sentence: "... only 1 person in the group knew 3 of the group (u_k), 3 people knew 2 of the group (u_i), and 4 people know 1 of the group (u_j)." Graphene identifies $\langle u_i, \texttt{conj}, u_j \rangle$ and $\langle u_i, \texttt{unk}, u_k \rangle$ but misses $\langle u_k, \texttt{conj}, u_j \rangle$, which can be generated by applying this rule.

2.3.2 Adaptive Extension of TLG

Our symbolic reasoning is *adaptive*. We rely on signals from neural reasoning to decide which inference steps are relevant to answering the questions and hence are admitted to extend the TLG. Specifically, each candidate extension ϵ applies an inference rule over a set of nodes $V_\epsilon \subseteq V$. We average their vector representations (which will be detailed later) to form the representation of ϵ:

$$\mathbf{h}_\epsilon = \frac{1}{|V_\epsilon|} \sum_{u_i \in V_\epsilon} \mathbf{h}_{u_i}\,. \quad (6)$$

Since ϵ is for predicting the correctness of o, we interact $\mathbf{h}_\epsilon$ with the representation of o, i.e., $\mathbf{g}_o$ in Equation (2), to predict the relevance score of ϵ:

$$rel_\epsilon = \texttt{sigmoid}(\texttt{linear}(\mathbf{h}_\epsilon \,\|\, \mathbf{g}_o))\,, \quad (7)$$

where $\|$ represents vector concatenation. We admit all possible ϵ to extend G such that $rel_\epsilon > \tau$ where τ is a predefined threshold.

Moreover, our neural-symbolic reasoning is *iterative*. In the $(l+1)$-th iteration, we restart symbolic reasoning with the raw TLG and recompute Equation (6) with node representations $\mathbf{h}_{u_i}^{(l)}$ from neural reasoning in the l-th iteration (which will be detailed in Section 2.3.3). The initial node representations $\mathbf{h}_{u_i}^{(0)}$ are obtained from a pre-trained language model. Specifically, we flatten V into a sequence of nodes in the order they appear in the text. Recall that V is divided into $V_c = \{u_1, \ldots, u_{|V_c|}\}$ and $V_o = \{u_{|V_c|+1}, \ldots, u_{|V|}\}$ representing the nodes extracted from c and o, respectively. Each node u_i is a token sequence $u_i = u_{i_1} \cdots u_{i_{|u_i|}}$. We use RoBERTa to encode V_c and V_o which are concatenated by <s> and </s>, where nodes inside V_c and V_o are separated by a special token "|":

$$\begin{aligned}&[\mathbf{h}_{\text{<s>}}; \mathbf{h}_{u_{1_1}}; \ldots; \mathbf{h}_{|}; \ldots; \mathbf{h}_{\text{</s>}}; \mathbf{h}_{u_{|V_c|+1_1}}; \ldots; \mathbf{h}_{|}; \ldots; \mathbf{h}_{\text{</s>}}] \\ &= \text{RoBERTa}(\text{<s>}\ u_{1_1} \cdots \mid \cdots\ \text{</s>}\ u_{|V_c|+1_1} \cdots \mid \cdots\ \text{</s>})\,.\end{aligned} \tag{8}$$

The output vector representations are averaged to form the initial representation of each node $u_i \in V$:

$$\mathbf{h}^{(0)}_{u_i} = \frac{1}{|u_i|} \sum_{j=1}^{|u_i|} \mathbf{h}_{u_{i_j}}\,. \tag{9}$$

2.3.3 Message Passing

To let the nodes in TLG interact with each other and fuse their information, our neural reasoning performs graph-based message passing (Gilmer et al., 2017) to update node representations in each iteration from $\mathbf{h}^{(l)}_{u_i}$ to $\mathbf{h}^{(l+1)}_{u_i}$. Since TLG is a heterogeneous graph containing multiple types of edges, we incorporate the node-to-node message passing mechanism in R-GCN (Schlichtkrull et al., 2018) as a basis. Further, observe that TLG is usually loosely connected and prone to cause insufficient interaction between V_c and V_o via long paths in limited iterations, which cannot be alleviated by simply increasing the number of iterations because it would raise other issues such as over-smoothing (Li et al., 2018; Chen et al., 2020). To enhance such interaction which is critical to predicting the correctness of o, we incorporate a novel *subgraph-to-node message passing mechanism* to holistically pass the information aggregated from a subgraph (e.g., V_c) to a node (e.g., each $u_i \in V_o$).

Specifically, without loss of generality, for each $u_i \in V_o$, we compute the u_i-attended aggregate representation of V_c by an attention-weighted sum of node representations over V_c:

$$\begin{aligned}\mathbf{h}^{(l)}_{V_c,u_i} &= \sum_{u_j \in V_c} \alpha_{i,j} \mathbf{h}^{(l)}_{u_j}\,, \text{ where} \\ \alpha_{i,j} &= \mathtt{softmax}_j([a_{i,1}; \ldots; a_{i,|V_c|}]^\intercal)\,, \\ a_{i,j} &= \mathtt{LeakyReLU}(\mathtt{linear}(\mathbf{h}^{(l)}_{u_i} \parallel \mathbf{h}^{(l)}_{u_j}))\,.\end{aligned} \tag{10}$$

Let N^i be the set of neighbors of u_i. Let $N^i_r \subseteq N^i$ be the subset under logical relation $r \in R$. We update the representation of u_i by passing messages to u_i from its neighbors and from V_c:

$$\begin{aligned}\mathbf{h}^{(l+1)}_{u_i} &= \mathtt{ReLU}(\sum_{r \in R} \sum_{u_j \in N^i_r} \frac{\alpha_{i,j}}{|N^i_r|} \mathbf{W}^{(l)}_r \mathbf{h}^{(l)}_{u_j} + \mathbf{W}^{(l)}_0 \mathbf{h}^{(l)}_{u_i} \\ &\qquad + \beta_i \mathbf{W}^{(l)}_{\text{subgraph}} \mathbf{h}^{(l)}_{V_c,u_i})\,, \text{ where} \\ \alpha_{i,j} &= \mathtt{softmax}_{\mathtt{idx}(a_{i,j})}([\ldots; a_{i,j}; \ldots]^\intercal) \text{ for all } u_j \in N^i\,, \\ a_{i,j} &= \mathtt{LeakyReLU}(\mathtt{linear}(\mathbf{h}^{(l)}_{u_i} \parallel \mathbf{h}^{(l)}_{u_j}))\,, \\ \beta_i &= \mathtt{sigmoid}(\mathtt{linear}(\mathbf{h}^{(l)}_{u_i} \parallel \mathbf{h}^{(l)}_{V_c,u_i}))\,,\end{aligned} \tag{11}$$

$\mathbf{W}^{(l)}_r, \mathbf{W}^{(l)}_0, \mathbf{W}^{(l)}_{\text{subgraph}}$ are matrices of learnable parameters, and $\mathtt{idx}(a_{i,j})$ returns the index of $a_{i,j}$ in the $|N^i|$-dimensional vector $[\ldots; a_{i,j}; \ldots]^\intercal$.

In an analogous way, for each $u_i \in V_c$, we compute the u_i-attended aggregate representation of V_o denoted by $\mathbf{h}^{(l)}_{V_o,u_i}$ and update $\mathbf{h}^{(l+1)}_{u_i}$.

Observe two differences between Equation (11) and its counterpart in the original R-GCN. First, we incorporate subgraph-to-node message passing and control it by a gating mechanism (i.e., β_i). Second, we weight node-to-node message passing by an attention mechanism (i.e., $\alpha_{i,j}$).

2.3.4 Graph Pooling

After L iterations where L is a hyperparameter, for each node $u_i \in V$, we fuse its representations over all the iterations with a residual connection:

$$\mathbf{h}^{\text{fus}}_{u_i} = \mathbf{h}^{(0)}_{u_i} + \mathtt{linear}(\mathbf{h}^{(1)}_{u_i} \parallel \cdots \parallel \mathbf{h}^{(L)}_{u_i})\,. \tag{12}$$

Inspired by Huang et al. (2021a), we feed all $\mathbf{h}^{\text{fus}}_{u_i}$ into a bidirectional residual GRU layer (Cho et al., 2014) to finalize node representations:

$$[\mathbf{h}^{\text{fnl}}_{u_1}; \ldots; \mathbf{h}^{\text{fnl}}_{u_{|V|}}] = \mathtt{Res\text{-}BiGRU}([\mathbf{h}^{\text{fus}}_{u_1}; \ldots; \mathbf{h}^{\text{fus}}_{u_{|V|}}])\,. \tag{13}$$

We aggregate these node representations by computing an o-attended weighted sum:

$$\begin{aligned}\mathbf{h}_V &= \sum_{u_i \in V} \alpha_i \mathbf{h}^{\text{fnl}}_{u_i}\,, \text{ where} \\ \alpha_i &= \mathtt{softmax}_i([a_1; \ldots; a_{|V|}]^\intercal)\,, \\ a_i &= \mathtt{LeakyReLU}(\mathtt{linear}(\mathbf{g}_o \parallel \mathbf{h}^{\text{fnl}}_{u_i}))\,,\end{aligned} \tag{14}$$

and $\mathbf{g}_o$ is the representation of o in Equation (2). We concatenate $\mathbf{h}_V$ and the relevance scores to form the representation of G:

$$\begin{aligned}\mathbf{h}_G &= (\mathbf{h}_V \parallel rel_{\mathcal{E}^{(1)}} \parallel \cdots \parallel rel_{\mathcal{E}^{(L)}})\,, \text{ where} \\ rel_{\mathcal{E}^{(l)}} &= \frac{1}{|\mathcal{E}^{(l)}|} \sum_{\epsilon \in \mathcal{E}^{(l)}} rel_\epsilon\,,\end{aligned} \tag{15}$$

$\mathcal{E}^{(l)}$ is the set of candidate extensions in the l-th iteration, and rel_ϵ is in Equation (7). In this way, we are able to train the network in Equation (7).

2.4 Answer Prediction

We fuse the representations of c, q, o and the TLG to predict the correctness of o:

$$score_o = \texttt{linear}(\tanh(\texttt{linear}(\mathbf{g}_c \parallel \mathbf{g}_q \parallel \mathbf{g}_o \parallel \mathbf{h}_G))) \,, \tag{16}$$

where $\mathbf{g}_c, \mathbf{g}_q, \mathbf{g}_o$ are in Equation (2).

2.5 Loss Function

Let $o_{\text{gold}} \in O$ be the correct answer. We optimize the cross-entropy loss with label smoothing:

$$\mathcal{L} = -(1-\gamma)score'_{o_{\text{gold}}} - \gamma\frac{1}{|O|}\sum_{o_i \in O} score'_{o_i} \,,$$

$$\text{where } score'_{o_i} = \log\frac{\exp(score_{o_i})}{\sum_{o_j \in O}\exp(score_{o_j})} \,, \tag{17}$$

and γ is a predefined smoothing factor.

3 Experiments

3.1 Datasets

We used two reasoning-based MRC datasets.

ReClor (Yu et al., 2020) consists of 6,138 four-option multiple-choice questions collected from standardized exams such as GMAT and LSAT. The questions were divided into 4,638 for training, 500 for development, and 1,000 for testing. The test set was further divided into 440 easy questions (Test-E) where each question could be correctly answered by some strong baseline method using only the options and ignoring the context and the question, and the rest 560 hard questions (Test-H).

LogiQA (Liu et al., 2020) consists of 8,768 four-option multiple-choice questions collected from the National Civil Servants Examination of China, which were translated into English. The questions were divided into 7,376 for training, 651 for development, and 651 for testing.

3.2 Implementation Details

We experimented on NVIDIA V100 (32GB).

We tuned hyperparameters on the development set of each dataset. Specifically, for text encoding, we used RoBERTa-large with hidden layer $= 24$ and hidden units $= 1,024$ implemented by Hugging Face (Wolf et al., 2020). For message passing, our implementation was based on DGL (Wang et al., 2019). For both datasets, we used the Adam optimizer, and set attention heads $= 16$, dropout rate $= 0.1$, epochs $= 10$, batch size $= 16$ selected from $\{8, 16, 24\}$, number of iterations $L = 2$ from $\{2, 3\}$, and maximum sequence length $= 384$. For ReClor, we set warm-up proportion $= 0.1$ from $\{0.1, 0.2\}$, learning rate $= 7e{-}6$ from $\{6e{-}6, 7e{-}6, 8e{-}6, 1e{-}5\}$, and seed $= 123$ from $\{123, 1234, 42, 43\}$. For LogiQA, we set warm-up proportion $= 0.2$ from $\{0.1, 0.2\}$, learning rate $= 8e{-}6$ from $\{6e{-}6, 7e{-}6, 8e{-}6, 1e{-}5\}$, and seed $= 42$ from $\{123, 1234, 42, 43\}$.

For the relevance score threshold τ below Equation (7), we set $\tau = 0.6$ from $\{0.4, 0.5, 0.6, 0.7\}$ for both datasets. For the smoothing factor γ in Equation (17), we set $\gamma = 0.25$ for both datasets.

To fit in our GPU's memory, we restricted a raw TLG to contain at most 25 nodes and 50 edges by, if needed, randomly merging nodes connected by an `unk` edge and/or deleting non-bridge edges while keeping the graph connected.

3.3 Baselines

We compared our approach, referred to as AdaLoGN, with popular pre-trained language models and with other known methods in the literature.

Reasoning-based MRC, like other MRC tasks, can be solved by using a pre-trained language model with a classification layer. Yu et al. (2020) reported the results of BERT$_{\text{LARGE}}$, RoBERTa$_{\text{LARGE}}$, and XLNet$_{\text{LARGE}}$ on ReClor. Huang et al. (2021a) reported the results of BERT$_{\text{LARGE}}$ and RoBERTa$_{\text{LARGE}}$ on LogiQA.

In the literature, we found the results of DAGN (Huang et al., 2021a), Focal Reasoner (Ouyang et al., 2021), and LReasoner (Wang et al., 2021a,b) on both datasets. For a fair comparison with our approach, we presented their results on RoBERTa$_{\text{LARGE}}$, while LReasoner achieved better results with ALBERT. Between the two variants of LReasoner, one without data augmentation (w/o DA) and the other with data augmentation (w/ DA), we presented both of their results but mainly compared with the former because our approach and other baseline methods would also benefit if data augmentation were incorporated.

Method	Dev	Test	Test-E	Test-H
$\text{BERT}_{\texttt{LARGE}}$	53.80	49.80	72.00	32.30
$\text{RoBERTa}_{\texttt{LARGE}}$	62.60	55.60	75.50	40.00
$\text{XLNet}_{\texttt{LARGE}}$	62.00	56.00	75.70	40.50
DAGN	65.80	58.30	75.91	44.46
Focal Reasoner	66.80	58.90	77.05	44.64
LReasoner (w/o DA)	65.20	58.30	78.60	42.30
LReasoner (w/ DA)	66.20	62.40	81.40	47.50
AdaLoGN	65.20	60.20	79.32	45.18
Human	–	63.00	57.10	67.20

Table 2: Comparison with baselines on ReClor.

Method	Dev	Test
$\text{BERT}_{\texttt{LARGE}}$	34.10	31.03
$\text{RoBERTa}_{\texttt{LARGE}}$	35.02	35.33
DAGN	36.87	39.32
Focal Reasoner	41.01	40.25
LReasoner (w/ DA)	38.10	40.60
AdaLoGN	39.94	40.71
Human	–	86.00

Table 3: Comparison with baselines on LogiQA.

3.4 Evaluation Metric

Following the literature, we reported accuracy, i.e., the proportion of correctly answered questions. For our approach we reported the max across 3 runs on the development set of each dataset.

3.5 Comparison with Baselines

On ReClor, as shown in Table 2, AdaLoGN outperformed all the baseline methods on the test set by at least 1.30%, except for LReasoner (w/ DA) which performed data augmentation so that the comparison might be unfair. AdaLoGN and LReasoner (w/ DA) both exceeded 60%, being comparable with human-level performance (63%).

On LogiQA, as shown in Table 3, AdaLoGN outperformed all the baseline methods on the test set, including LReasoner (w/ DA). Still, our result (40.71%) was not comparable with human-level performance (86%).

In particular, on both ReClor and LogiQA, AdaLoGN exceeded DAGN on the test set by 1.39%–1.90%, which demonstrated the effectiveness of our approach in addressing the limitations of DAGN mentioned in Section 1.

3.6 Ablation Study

We conducted an ablation study to evaluate the effectiveness of the two main technical contributions in our approach: adaptive extension of TLG and subgraph-to-node message passing.

Method	Dev	Test	Test-E	Test-H
AdaLoGN	65.20	60.20	79.32	45.18
$\text{AdaLoGN}_{\texttt{no-ext}}$	65.80	59.50	77.27	45.54
$\text{AdaLoGN}_{\texttt{full-ext}}$	65.00	58.80	78.19	43.57
$\text{AdaLoGN}_{\texttt{no-at}}$	64.80	59.40	79.77	43.39
$\text{AdaLoGN}_{\texttt{n2n}}$	65.20	57.60	77.95	41.61
$\text{AdaLoGN}_{\texttt{n2n+}}$	65.00	58.60	78.64	42.86

Table 4: Ablation study on ReClor.

Method	Dev	Test
AdaLoGN	39.94	40.71
$\text{AdaLoGN}_{\texttt{no-ext}}$	37.94	39.02
$\text{AdaLoGN}_{\texttt{full-ext}}$	39.63	39.02
$\text{AdaLoGN}_{\texttt{no-at}}$	38.56	39.94
$\text{AdaLoGN}_{\texttt{n2n}}$	38.40	39.02
$\text{AdaLoGN}_{\texttt{n2n+}}$	38.40	38.86

Table 5: Ablation study on LogiQA.

3.6.1 Effectiveness of Adaptive Extension

We compared the standard version of AdaLoGN with two variants removing adaptive extension.

- $\text{AdaLoGN}_{\texttt{no-ext}}$ performs no extension.
- $\text{AdaLoGN}_{\texttt{full-ext}}$ performs full extension by computing and admitting the deductive closure.

On ReClor, as shown in Table 4, both variants exhibited a fair decrease in accuracy on the test set by 0.70%–1.40%. On LogiQA, as shown in Table 5, the decreases were larger, 1.69% on the test set, possibly because the questions in LogiQA were harder so that the effectiveness of our adaptive extension became more noticeable. Interestingly, on both datasets, $\text{AdaLoGN}_{\texttt{full-ext}}$ was not better than $\text{AdaLoGN}_{\texttt{no-ext}}$ on the test set, indicating that a naive injection of logical reasoning into neural reasoning might not have positive effects.

We analyzed the distributions of relevance scores of candidate extensions, i.e., rel_ϵ in Equation (7). As shown in Figure 6, they approximated a normal distribution on both datasets. By setting the threshold $\tau = 0.6$, we admitted 19.57% and 4.86% of the extensions on ReClor and LogiQA, respectively.

We also compared with a variant of AdaLoGN using a subset of inference rules.

- $\text{AdaLoGN}_{\texttt{no-at}}$ ignores the adjacency-transmission rule.

By ignoring the adjacency-transmission rule, $\text{AdaLoGN}_{\texttt{no-at}}$ showed a decrease in accuracy on the test sets by 0.77%–0.80%, suggesting the usefulness of this rule despite its heuristic nature.

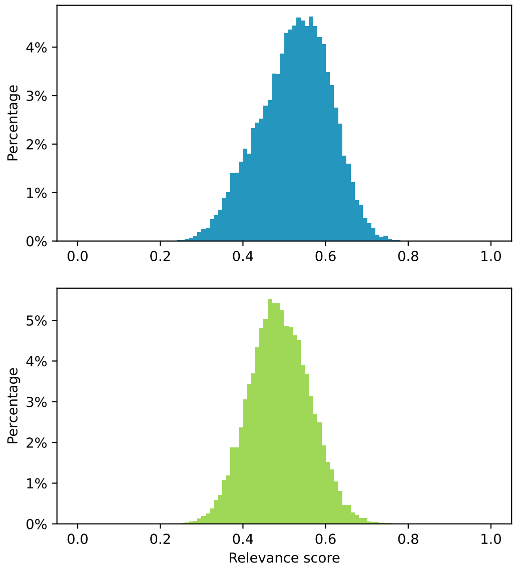


Figure 6: Distributions of relevance scores of candidate extensions. Top: on the development set of Reclor; Bottom: on the development set of LogiQA.

3.6.2 Effectiveness of Subgraph-to-Node Message Passing

We compared the standard version of AdaLoGN with two variants removing subgraph-to-node message passing or implementing it in a different way.

- AdaLoGN$_{\texttt{n2n}}$ only performs node-to-node message passing in a standard way.
- AdaLoGN$_{\texttt{n2n+}}$ only performs node-to-node message passing but, as an alternative to our holistic subgraph-to-node message passing, it adds a bidirectional unk edge between each node in the context subgraph and each node in the option subgraph to enhance context-option interaction.

On ReClor, as shown in Table 4, both variants exhibited a large decrease in accuracy on the test set by 1.60%–2.60%. On LogiQA, as shown in Table 5, the decreases were also large, 1.69%–1.85% on the test set. The results demonstrated the effectiveness of our subgraph-to-node message passing.

Compared with AdaLoGN$_{\texttt{n2n}}$, AdaLoGN$_{\texttt{n2n+}}$ achieved better results on ReClor but worse results on LogiQA on the test set, indicating that a naive enhancement of context-option interaction could have negative effects.

3.7 Error Analysis

From the development set of each dataset, we randomly sampled fifty questions to which our approach outputted an incorrect answer. We analyzed the sources of these errors. Note that an error could have a mixture of multiple sources.

Source of Error	ReClor	LogiQA
Construction of raw TLG	38%	36%
Adaptive extension of TLG	18%	22%
Expressivity of symbolic reasoning	20%	18%
Others (about neural reasoning)	46%	40%

Table 6: Error analysis of AdaLoGN.

As shown in Table 6, we mainly relied on Graphene to extract a raw TLG from text based on syntactic analysis, which accounted for about one third of the errors (36%–38%). Our adaptive extension of TLG constituted about one fifth of the errors (18%–22%), e.g., some excessive extensions produced irrelevant logical relations which might mislead message passing. One fifth of the errors (18%–20%) were due to the limited expressivity of our symbolic reasoning, i.e., a subset of propositional logic, while some questions required quantifiers. Other errors might be related to neural reasoning such as message passing or answer prediction (40%–46%).

3.8 Run Time

On both ReClor and LogiQA, our approach used about 0.8 second for answering a question.

4 Related Work

4.1 Reasoning-Based MRC

While simple MRC tasks have been well studied, complex MRC tasks requiring various reasoning capabilities are receiving increasing research attention. Among others, multi-hop MRC tasks in HotpotQA (Yang et al., 2018) and WikiHop (Welbl et al., 2018) require retrieving and reading multiple supporting passages to answer a question. They can be solved by constructing and reasoning over a graph connecting passages that overlap or co-occur with each other (Qiu et al., 2019; Tu et al., 2020), by implicitly supervising a retriever via word weighting (Huang et al., 2021b), or by iteratively applying dense retrieval (Xiong et al., 2021). MRC tasks in DROP (Dua et al., 2019) require discrete reasoning such as addition, counting, and sorting. Neural networks have been extended to incorporate modules that can perform such reasoning over numbers and dates mentioned in a given context (Gupta et al., 2020). For MRC tasks in CommonsenseQA (Talmor et al., 2019) which are targeted at commonsense knowledge and reasoning, recent methods fuse external commonsense knowledge with pre-trained language models for reasoning (Yan et al., 2021; Xu et al., 2021). There are also studies on MRC tasks requiring spatial/geographical reasoning (Huang et al., 2019; Li et al., 2021) and temporal/causal reasoning (Sun et al., 2018).

Different from the above reasoning capabilities, the MRC tasks considered in this paper require *logical reasoning*, such as reasoning about sufficient and necessary conditions, categorization, conjunctions and disjunctions. Pre-trained language models alone struggled and were far behind human-level performance on such tasks in ReClor (Yu et al., 2020) and LogiQA (Liu et al., 2020) due to their weakness in logical reasoning.

Among existing methods for solving such tasks, DAGN (Huang et al., 2021a) and Focal Reasoner (Ouyang et al., 2021) extract discourse or coreference relations from text and represent as a graph of text units. Then they employ GNN to pass messages and update representations for predicting an answer. Different from their neural nature, our approach *symbolically performs logical reasoning* as required by such tasks, by applying inference rules over extracted logical relations to extend the graph. This feature resembles LReasoner (Wang et al., 2021a,b) which extends the context with inferred logical relations to benefit the subsequent neural reasoning. However, different from LReasoner which computes the deductive closure and identifies relevant extensions by text overlapping with the options in an unsupervised manner, our approach predicts relevance based on signals from neural reasoning *in a supervised manner*, and our prediction *evolves over iterations* after sufficient interaction between symbolic and neural reasoning. All these features helped our approach achieve better performance in the experiments.

4.2 Neural-Symbolic Reasoning

Our approach represents a novel implementation of neural-symbolic reasoning (Raedt et al., 2020), and it differs from the following existing methods.

One paradigm of neural-symbolic reasoning is logic-driven neural reasoning. For example, logical constraints can be compiled into a neural network by augmenting the loss function (Xu et al., 2018) or the network structure (Li and Srikumar, 2019). Logical connectives, quantifiers, and consistency checking can also be approximated by neural networks (Dong et al., 2019; Ren et al., 2020; Gu et al., 2019). While these methods incorporate logical reasoning into neural reasoning via emulation, our approach *explicitly performs logical reasoning* by applying inference rules over logical relations. Such exact inference is more accurate than emulation-based approximation.

Another paradigm is neural-driven logical reasoning. For example, neural networks have been employed to predict the truth of an atom in answering first-order logic queries (Arakelyan et al., 2021), and to implement predicates in probabilistic logic programming (Manhaeve et al., 2021). These methods and our approach cope with different problems, thus using different techniques. Specifically, while these methods complement logical reasoning with extra facts generated by neural reasoning, our approach *filters inferred logical relations* based on signals from neural reasoning.

Moreover, observe that the neural-symbolic interaction in the above methods are unidirectional, i.e., they leverage either symbolic or neural reasoning to reinforce the other. By contrast, we allow *bidirectional neural-symbolic interaction* where neural and symbolic reasoning mutually and iteratively reinforce each other for better performance.

5 Conclusion

To meet the challenge of reasoning-based MRC, we presented a neural-symbolic approach where neural and symbolic reasoning mutually and iteratively reinforce each other via our new AdaLoGN model. We also enhanced graph-based neural reasoning with a novel subgraph-to-node message passing mechanism. Since these ideas are quite general, we believe they have great potential for a variety of applications beyond MRC, e.g., link prediction.

Error analysis has revealed some shortcomings of our approach. Currently we rely on syntactic tools to extract a raw TLG from text. We will explore other extraction methods to achieve a higher quality. We also plan to apply more inference rules and incorporate quantifiers to improve the expressivity of our symbolic reasoning.

Acknowledgements

This work was supported in part by the NSFC (62072224) and in part by the Beijing Academy of Artificial Intelligence (BAAI).

References

Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. 2021. Complex query answering with neural link predictors. In *ICLR 2021*.

Matthias Cetto, Christina Niklaus, André Freitas, and Siegfried Handschuh. 2018. Graphene: semantically-linked propositions in open information extraction. In *COLING 2018*, pages 2300–2311.

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *AAAI-IAAI-EAAI 2020*, pages 3438–3445.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP 2014*, pages 1724–1734.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT 2019*, pages 4171–4186.

Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou. 2019. Neural logic machines. In *ICLR 2019*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *NAACL-HLT 2019*, pages 2368–2378.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural message passing for quantum chemistry. In *ICML 2017*, pages 1263–1272.

Yu Gu, Jeff Z. Pan, Gong Cheng, Heiko Paulheim, and Giorgos Stoilos. 2019. Local ABox consistency prediction with transparent TBoxes using gated graph neural networks. In *NeSy 2019*.

Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. 2020. Neural module networks for reasoning over text. In *ICLR 2020*.

Yinya Huang, Meng Fang, Yu Cao, Liwei Wang, and Xiaodan Liang. 2021a. DAGN: discourse-aware graph network for logical reasoning. In *NAACL-HLT 2021*, pages 5848–5855.

Zixian Huang, Yulin Shen, Xiao Li, Yuang Wei, Gong Cheng, Lin Zhou, Xinyu Dai, and Yuzhong Qu. 2019. GeoSQA: A benchmark for scenario-based question answering in the geography domain at high school level. In *EMNLP-IJCNLP 2019*, pages 5865–5870.

Zixian Huang, Ao Wu, Yulin Shen, Gong Cheng, and Yuzhong Qu. 2021b. When retriever-reader meets scenario-based multiple-choice questions. In *Findings of EMNLP 2021*, pages 985–994.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR 2017*.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI-IAAI-EAAI 2018*, pages 3538–3545.

Tao Li and Vivek Srikumar. 2019. Augmenting neural networks with first-order logic. In *ACL 2019*, pages 292–302.

Xiao Li, Yawei Sun, and Gong Cheng. 2021. TSQA: tabular scenario based question answering. In *AAAI-IAAI-EAAI 2021*, pages 13297–13305.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. LogiQA: A challenge dataset for machine reading comprehension with logical reasoning. In *IJCAI 2020*, pages 3622–3628.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. 2021. Neural probabilistic logic programming in DeepProbLog. *Artif. Intell.*, 298:103504.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Siru Ouyang, Zhuosheng Zhang, and Hai Zhao. 2021. Fact-driven logical reasoning. *CoRR*, abs/2105.10334.

Lin Qiu, Yunxuan Xiao, Yanru Qu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. 2019. Dynamically fused graph network for multi-hop reasoning. In *ACL 2019*, pages 6140–6150.

Luc De Raedt, Sebastijan Dumancic, Robin Manhaeve, and Giuseppe Marra. 2020. From statistical relational to neuro-symbolic artificial intelligence. In *IJCAI 2020*, pages 4943–4950.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP 2016*, pages 2383–2392.

Hongyu Ren, Weihua Hu, and Jure Leskovec. 2020. Query2box: reasoning over knowledge graphs in vector space using box embeddings. In *ICLR 2020*.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC 2018*, pages 593–607.

Yawei Sun, Gong Cheng, and Yuzhong Qu. 2018. Reading comprehension with graph-based temporal-casual reasoning. In *COLING 2018*, pages 806–817.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *NAACL-HLT 2019*, pages 4149–4158.

Ming Tu, Kevin Huang, Guangtao Wang, Jing Huang, Xiaodong He, and Bowen Zhou. 2020. Select, answer and explain: Interpretable multi-hop reading comprehension over multiple documents. In *AAAI-IAAI-EAAI 2020*, pages 9073–9080.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR 2018*.

Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2019. Deep Graph Library: a graph-centric, highly-performant package for graph neural networks. *CoRR*, abs/1909.01315.

Siyuan Wang, Zhongkun Liu, Wanjun Zhong, Ming Zhou, Zhongyu Wei, Zhumin Chen, and Nan Duan. 2021a. From LSAT: The progress and challenges of complex reasoning. *arXiv preprint arXiv:2108.00648*.

Siyuan Wang, Wanjun Zhong, Duyu Tang, Zhongyu Wei, Zhihao Fan, Daxin Jiang, Ming Zhou, and Nan Duan. 2021b. Logic-driven context extension and data augmentation for logical reasoning of text. *arXiv preprint arXiv:2105.03659*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Trans. Assoc. Comput. Linguistics*, 6:287–302.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: state-of-the-art natural language processing. In *EMNLP 2020*, pages 38–45.

Wenhan Xiong, Xiang Lorraine Li, Srini Iyer, Jingfei Du, Patrick S. H. Lewis, William Yang Wang, Yashar Mehdad, Scott Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oguz. 2021. Answering complex open-domain questions with multi-hop dense retrieval. In *ICLR 2021*.

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. 2018. A semantic loss function for deep learning with symbolic knowledge. In *ICML 2018*, pages 5498–5507.

Yichong Xu, Chenguang Zhu, Ruochen Xu, Yang Liu, Michael Zeng, and Xuedong Huang. 2021. Fusing context into knowledge graph for commonsense question answering. In *Findings of ACL-IJCNLP 2021*, pages 1201–1207.

Jun Yan, Mrigank Raman, Aaron Chan, Tianyu Zhang, Ryan A. Rossi, Handong Zhao, Sungchul Kim, Nedim Lipka, and Xiang Ren. 2021. Learning contextualized knowledge structures for commonsense reasoning. In *Findings of ACL-IJCNLP 2021*, pages 4038–4051.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP 2018*, pages 2369–2380.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. 2020. ReClor: A reading comprehension dataset requiring logical reasoning. In *ICLR 2020*.

Responsible NLP Research Checklist

Members of the ACL are responsible for adhering to the ACL code of ethics. The ARR Responsible NLP Research checklist is designed to encourage best practices for responsible research, addressing issues of research ethics, societal impact and reproducibility.

Please read the Responsible NLP Research checklist guidelines for information on how to answer these questions. Note that not answering positively to a question is not grounds for rejection.

All supporting evidence can appear either in the main paper or the supplemental material. For each question, if you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Please do not modify, reorder, delete or add questions, question options or other wording of this document.

A For every submission

A1 Did you discuss the *limitations* of your work?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification Section 3.7

A2 Did you discuss any potential *risks* of your work?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

N/A

Section or justification Click or tap here to enter text.

A3 Do the abstract and introduction summarize the paper's main claims?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification Abstract, Section 1

B Did you use or create *scientific artifacts*?

If you answer **Yes**, provide the section number; if you answer **No**, you can skip the rest of this section.

Yes

If yes:

B1 Did you cite the creators of artifacts you used?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification   Sections 2.1, Section 2.2.2, Section 3.1, Section 3.2

B2 Did you discuss the *license or terms* for use and/or distribution of any artifacts?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification   The license for our code is available on GitHub.

B3 Did you discuss if your use of existing artifact(s) was consistent with their *intended use*, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

N/A

Section or justification   Click or tap here to enter text.

B4 Did you discuss the steps taken to check whether the data that was collected/used contains any *information that names or uniquely identifies individual people* or *offensive content*, and the steps taken to protect / anonymize it?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

N/A

Section or justification   Click or tap here to enter text.

B5 Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification   Section 3.1

B6 Did you report relevant statistics like the number of examples, details of train/test/dev splits, etc. for the data that you used/created?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification   Section 3.1

C Did you run *computational experiments*?

If you answer **Yes**, provide the section number; if you answer **No**, you can skip the rest of this section.

Yes

If yes:

C1 Did you report the *number of parameters* in the models used, the *total computational budget* (e.g., GPU hours), and *computing infrastructure* used?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification   Section 3.2, Section 3.8

C2 Did you discuss the experimental setup, including *hyperparameter search* and *best-found hyperparameter* values?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification   Section 3.2

C3 Did you report *descriptive statistics* about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification   Section 3.2, Section 3.4, Section 3.5

C4 If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification   Section 3.2

D Did you use *human annotators* (e.g., crowdworkers) or *research with human subjects*?

If you answer **Yes**, provide the section number; if you answer **No**, you can skip the rest of this section.

No

If yes:

D1 Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Choose an item.

Section or justification   Click or tap here to enter text.

D2 Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such *payment is adequate* given the participants' demographic (e.g., country of residence)?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Choose an item.

Section or justification   Click or tap here to enter text.

D3 Did you discuss whether and how consent was obtained from people whose data you're using/curating (e.g., did your instructions explain how the data would be used)?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Choose an item.

Section or justification   Click or tap here to enter text.

D4 Was the data collection protocol *approved* (or *determined exempt*) by an ethics review board?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Choose an item.

Section or justification   Click or tap here to enter text.

D5 Did you report the basic demographic and geographic characteristics of the *annotator* population that is the source of the data?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Choose an item.

Section or justification   Click or tap here to enter text.