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

Code & data: https://github.com/nju-websoft/AdaLoGN

2022.4.3 • ChongQing

—— ACL2022

Reported by Sijin Liu

1.Introduction

2.Method

3.Experiments

Introduction

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:

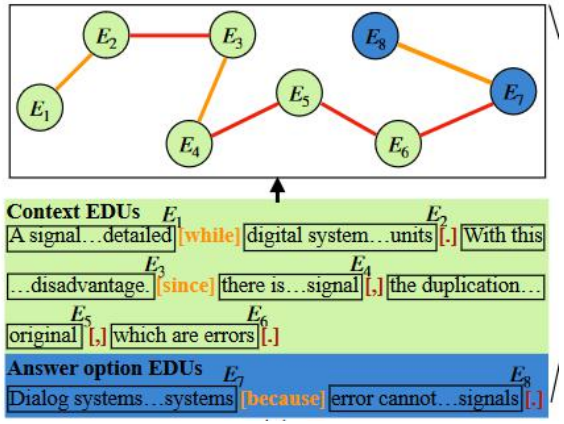
A. The company gets project A and product B is put on the market on schedule.

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.

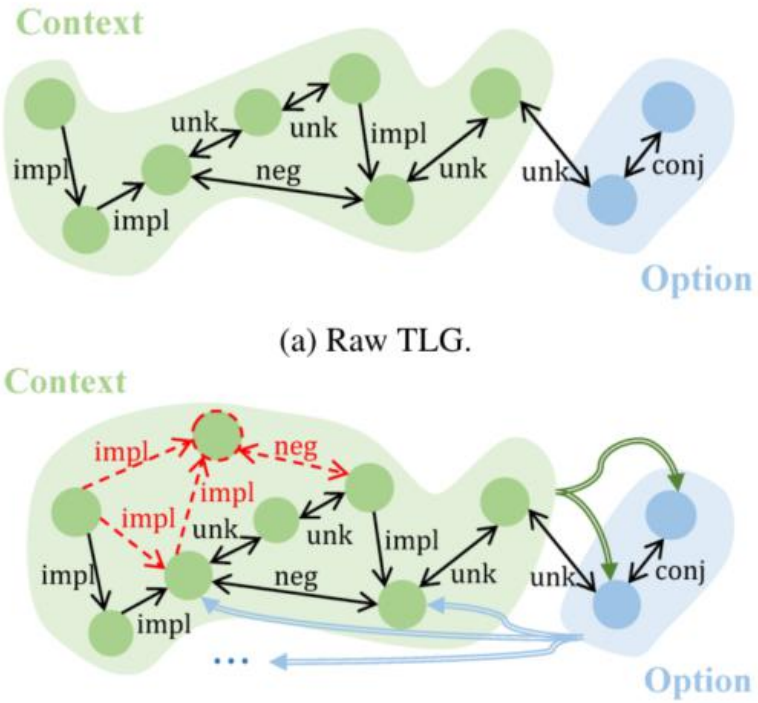


DAGN method (Huang et al., 2021a). It breaks down the context and each option into a set of EDUs and connects them with discourse relations as a graph.

Two limitations:

- Despite the graph representation, it is predominantly a neural method over discourse relations.
- The graph is often loosely connected and composed of long paths. Node-to-node message passing implemented in existing GNN models is prone to provide insufficient interaction between the context and the option.

Method



(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.

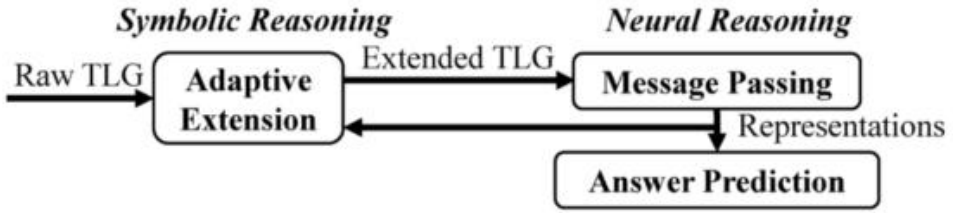


Figure 3: Our main idea: mutual and iterative reinforcement between symbolic and neural reasoning.

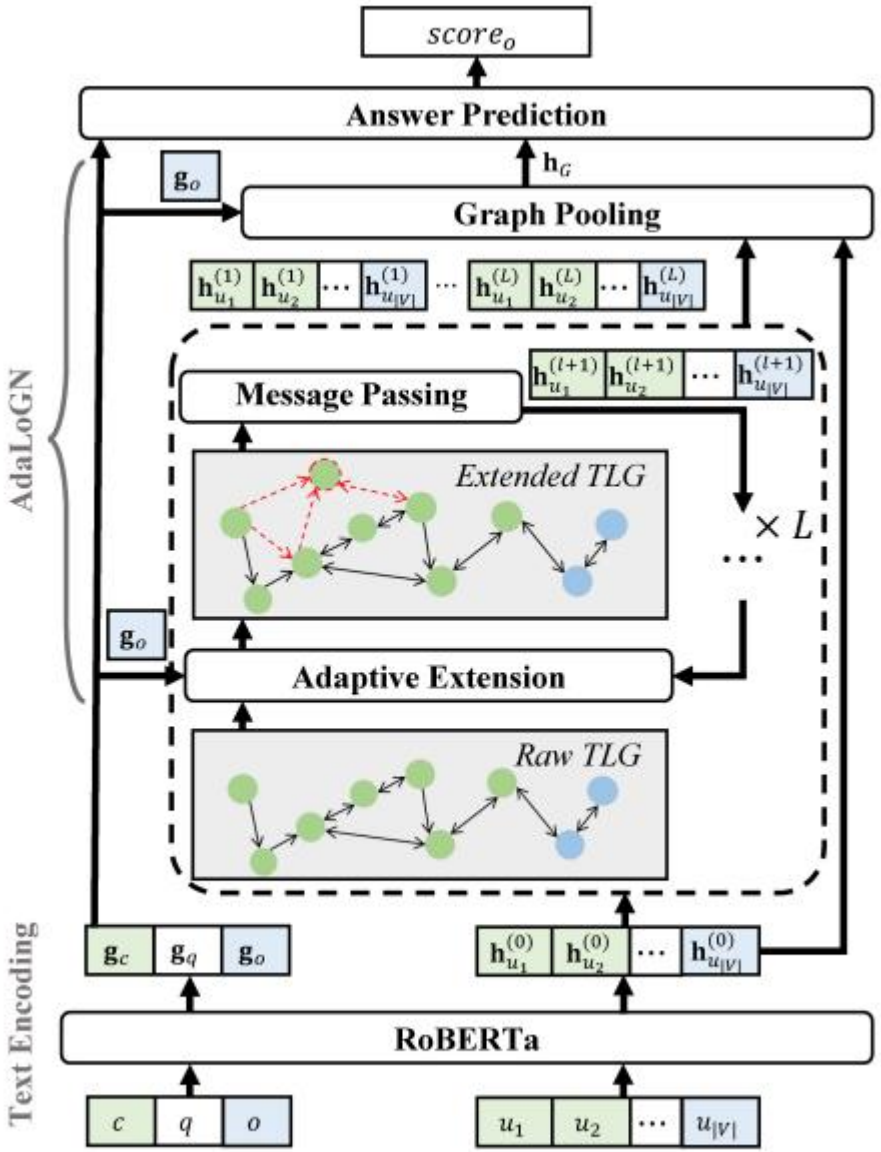


Figure 4: Overview of our approach.

Method

Problem Definition

A MRC task $<c, q, O>$ 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.

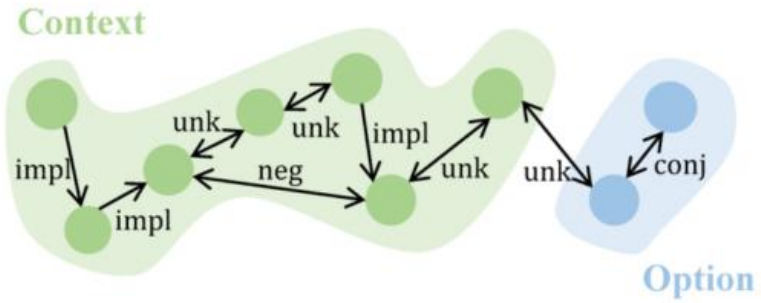
Text Encoding

$$c = c_1 \cdots c_{|c|},\ q = q_1 \cdots q_{|q|},\ \text{and}\ o = o_1 \cdots o_{|o|}$$

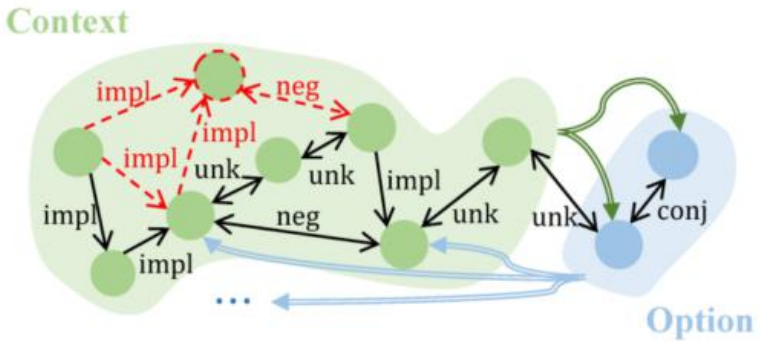
$$\begin{aligned}&[\mathbf{g}_{<s>}; \mathbf{g}_{c_1}; \ldots; \mathbf{g}_{</s>}; \mathbf{g}_{q_1}; \ldots; \mathbf{g}_{o_1}; \ldots; \mathbf{g}_{</s>}] \\ &= \text{RoBERTa}(<s>\ c_1 \cdots\ </s>\ q_1 \cdots\ o_1 \cdots\ </s>)\,.\end{aligned} \tag{1}$$

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

Method



(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.

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.

- **Definition of TLG (Text Logic Graph)**

$$G = < V, E >$$

V is a set of nodes representing EDUs of the text
$E \subseteq V \times R \times V$ is a set of labeled directed edges representing logical relations between EDUs described in the text.

$R = \{conj, disj, impl, neg, rev, unk\}$:
- conjunction ($conj$), disjunction ($disj$), implication ($impl$), and negation (neg)
- reversed implication (rev) is introduced to represent the inverse relation of $impl$
- unk represents an unknown relation

Method

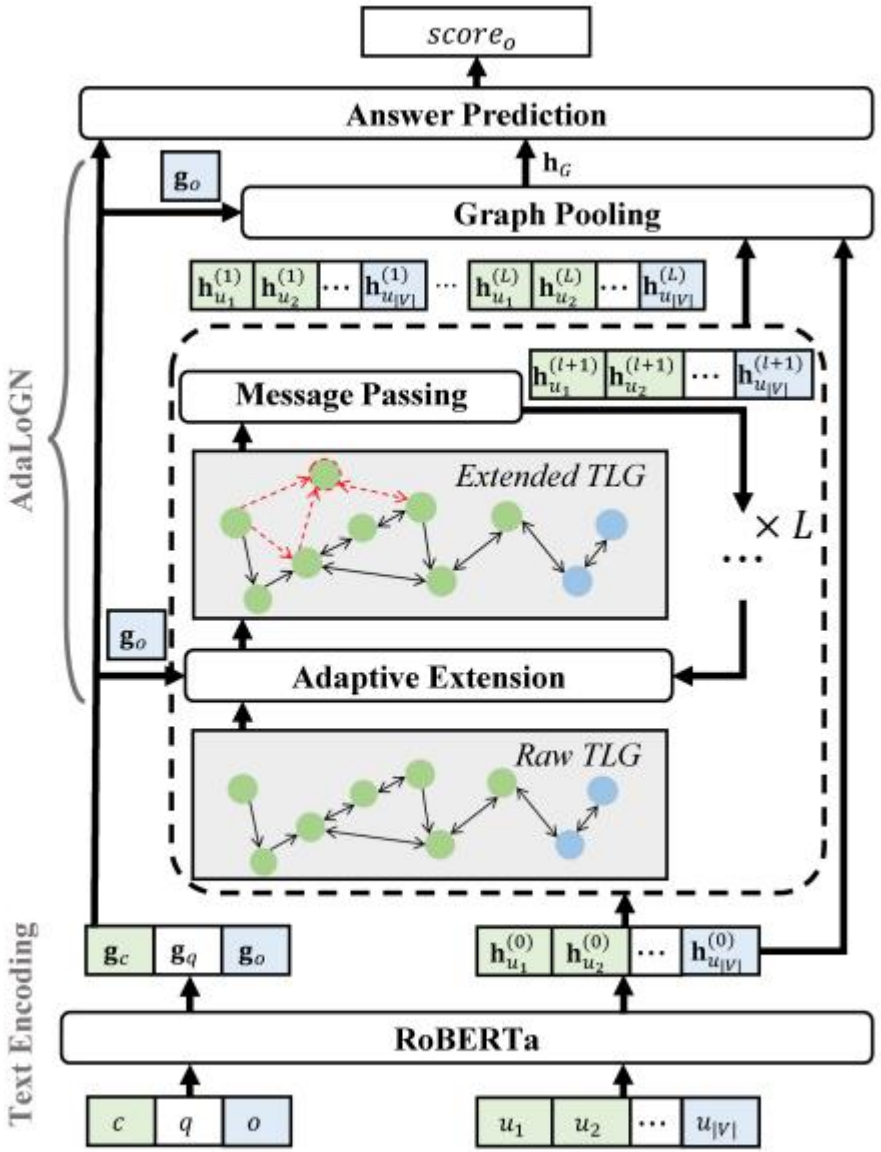
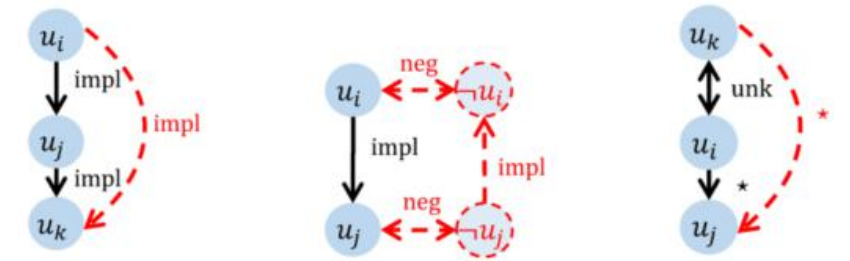


Figure 4: Overview of our approach.

Adaptive Logic Graph Network (AdaLoGN)

Inference Rules



(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 $\texttt{rev}$ edges.

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

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

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

Method

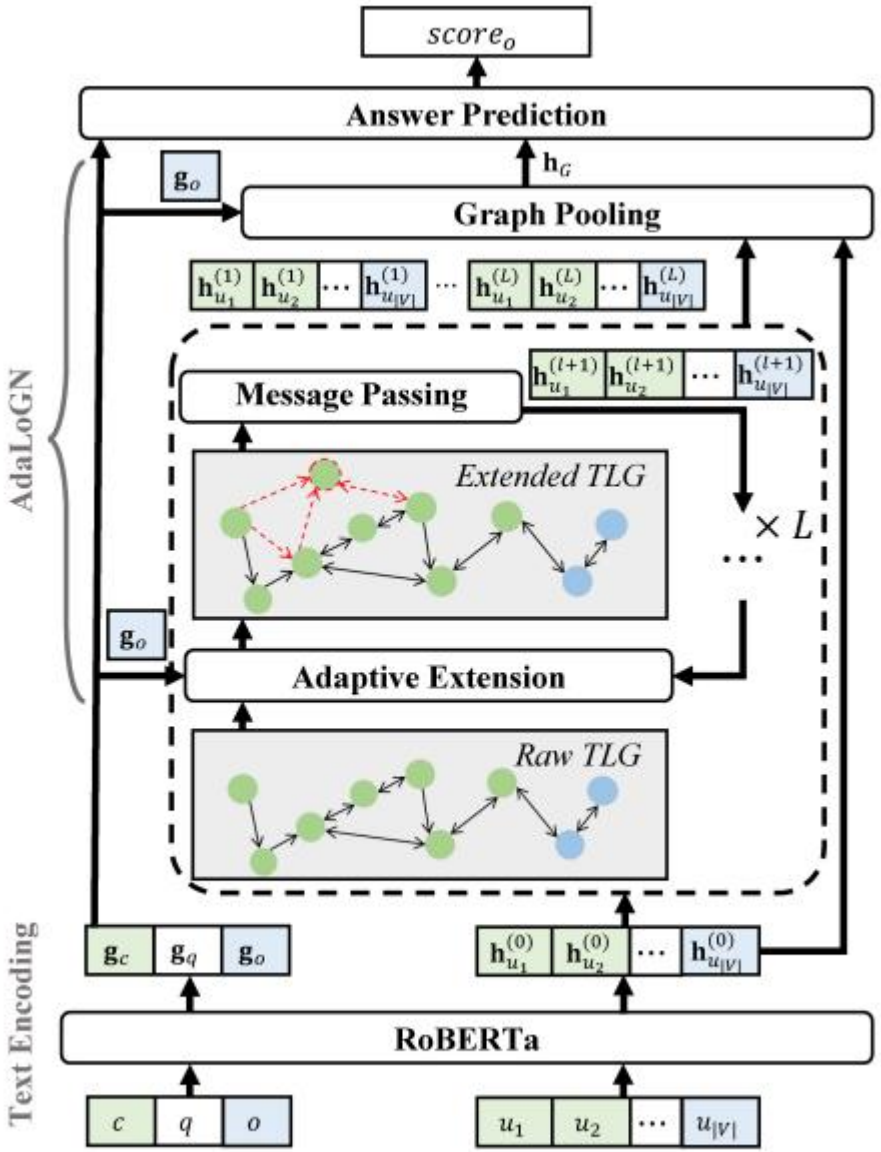


Figure 4: Overview of our approach.

Adaptive Logic Graph Network (AdaLoGN)

Adaptive Extension of TLG

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

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

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

$$V_c = \{u_1, \ldots, u_{|V_c|}\} \quad V_o = \{u_{|V_c|+1}, \ldots, u_{|V|}\} \quad u_i = u_{i_1} \cdots u_{i_{|u_i|}}$$

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

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

Method

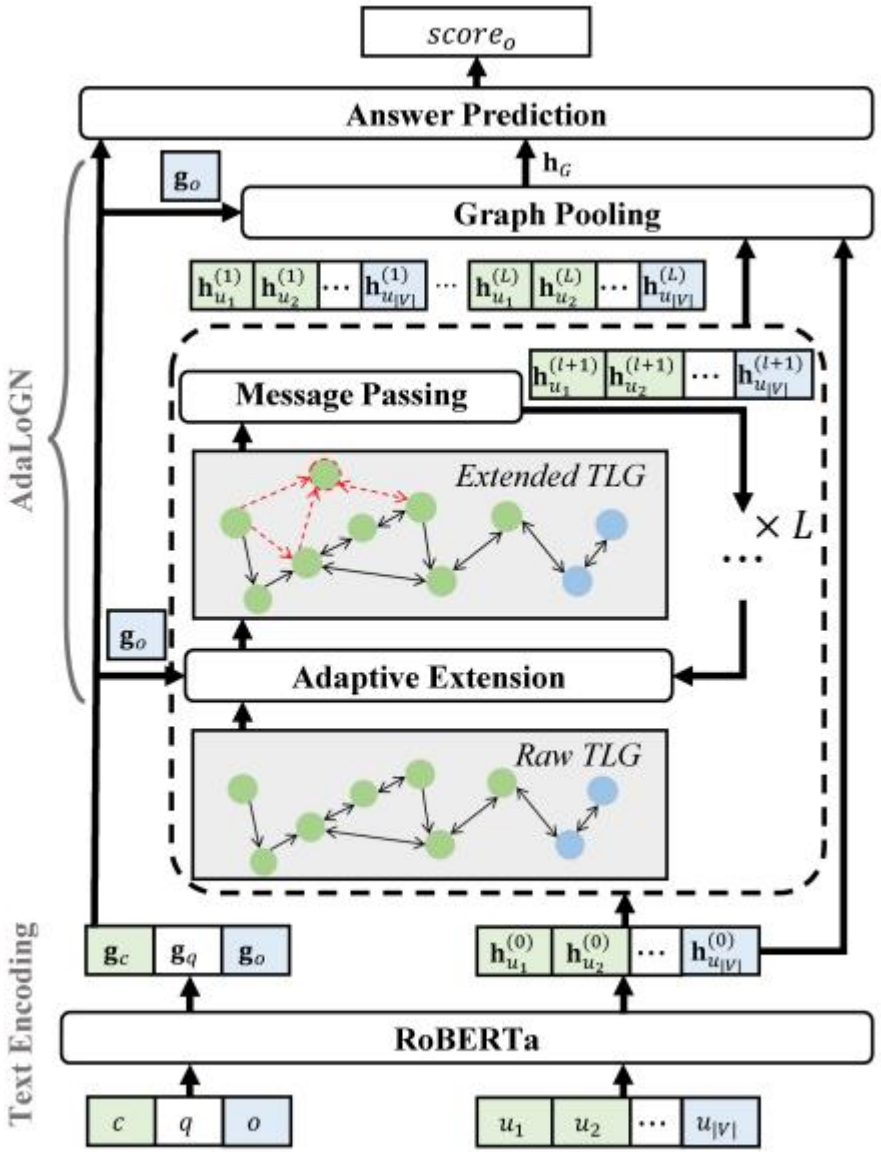


Figure 4: Overview of our approach.

Adaptive Logic Graph Network (AdaLoGN)

Message Passing

for each $u_i \in V_o$

$$\mathbf{h}_{V_c,u_i}^{(l)} = \sum_{u_j \in V_c} \alpha_{i,j} \mathbf{h}_{u_j}^{(l)}, \text{ where}$$

$$\alpha_{i,j} = \texttt{softmax}_j([a_{i,1}; \ldots; a_{i,|V_c|}]^\intercal),$$

$$a_{i,j} = \texttt{LeakyReLU}(\texttt{linear}(\mathbf{h}_{u_i}^{(l)} \parallel \mathbf{h}_{u_j}^{(l)})). \quad (10)$$

$$\mathbf{h}_{u_i}^{(l+1)} = \texttt{ReLU}(\sum_{r \in R} \sum_{u_j \in N_r^i} \frac{\alpha_{i,j}}{|N_r^i|} \mathbf{W}_r^{(l)} \mathbf{h}_{u_j}^{(l)} + \mathbf{W}_0^{(l)} \mathbf{h}_{u_i}^{(l)} + \beta_i \mathbf{W}_{\text{subgraph}}^{(l)} \mathbf{h}_{V_c,u_i}^{(l)}), \text{ where}$$

$$\alpha_{i,j} = \texttt{softmax}_{\texttt{idx}(a_{i,j})}([\ldots; a_{i,j}; \ldots]^\intercal) \text{ for all } u_j \in N^i,$$

$$a_{i,j} = \texttt{LeakyReLU}(\texttt{linear}(\mathbf{h}_{u_i}^{(l)} \parallel \mathbf{h}_{u_j}^{(l)})),$$

$$\beta_i = \texttt{sigmoid}(\texttt{linear}(\mathbf{h}_{u_i}^{(l)} \parallel \mathbf{h}_{V_c,u_i}^{(l)})), \quad (11)$$

Method

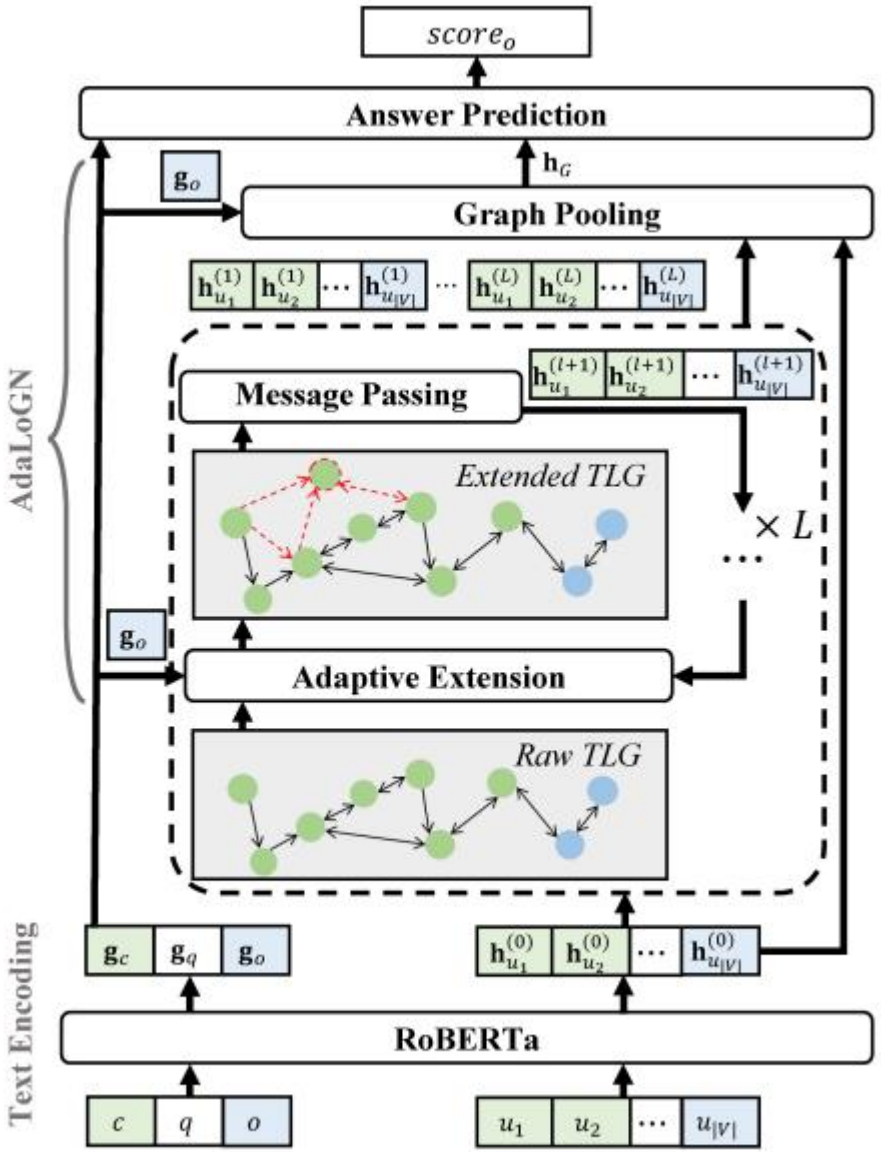


Figure 4: Overview of our approach.

Adaptive Logic Graph Network (AdaLoGN)

Graph Pooling

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

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

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

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

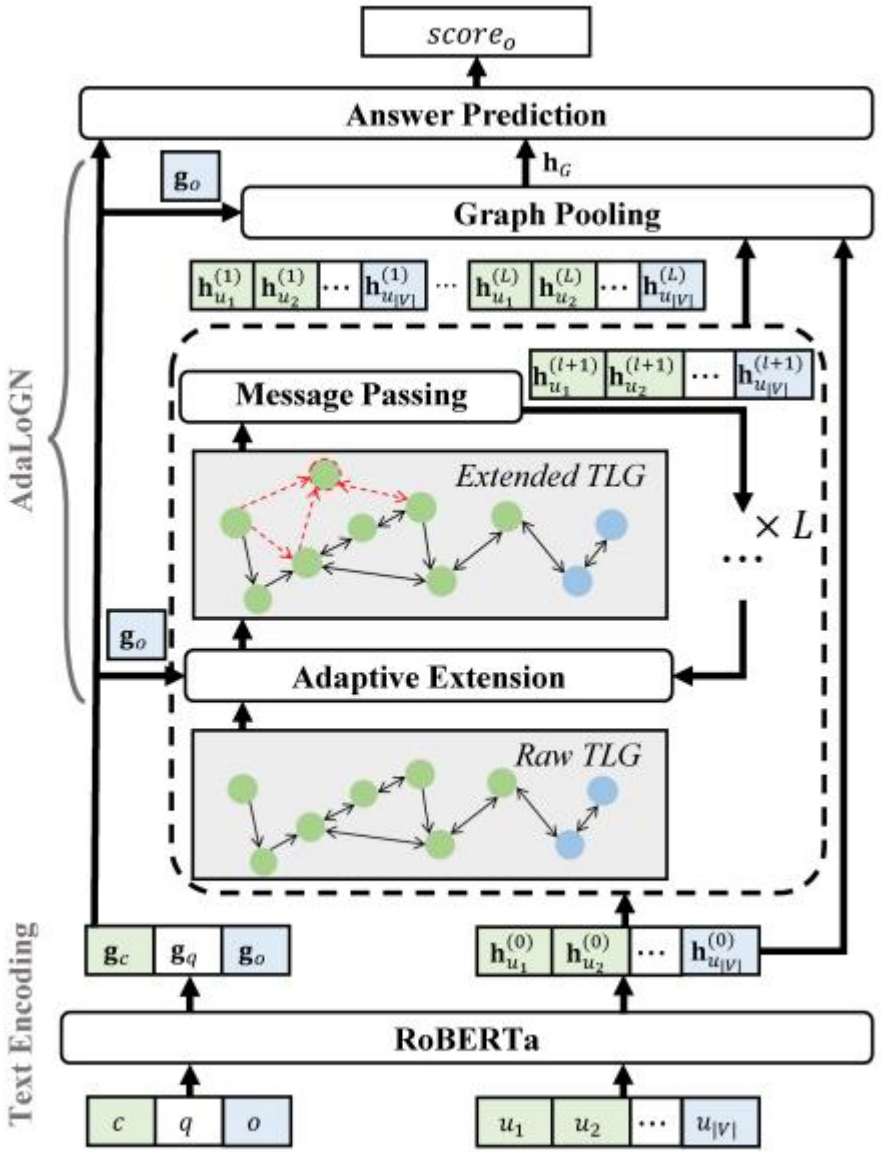


Figure 4: Overview of our approach.

Method

Answer Prediction

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

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

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

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

and γ is a predefined smoothing factor.

Experiments

Method	Dev	Test	Test-E	Test-H
BERT$_{\text{LARGE}}$	53.80	49.80	72.00	32.30
RoBERTa$_{\text{LARGE}}$	62.60	55.60	75.50	40.00
XLNet$_{\text{LARGE}}$	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
BERT$_{\text{LARGE}}$	34.10	31.03
RoBERTa$_{\text{LARGE}}$	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.

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

Table 4: Ablation study on ReClor.

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

Table 5: Ablation study on LogiQA.

Experiments

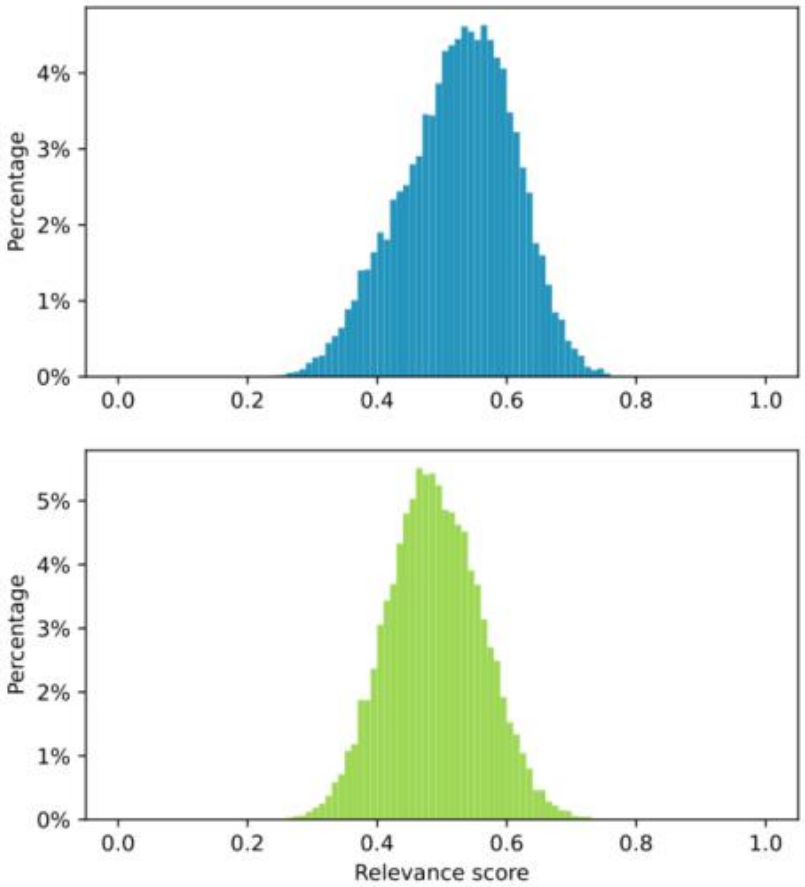


Figure 6: Distributions of relevance scores of candidate extensions. Top: on the development set of Reclor; Bottom: on the development set of LogiQA.

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.