Multi-Step Deductive Reasoning Over Natural Language: An Empirical Study on Out-of-Distribution Generalisation

Qiming Bao[1,2,*], Alex Yuxuan Peng[1,*], Tim Hartill[1], Neset Tan[1], Zhenyun Deng[1], Michael Witbrock[1,*] and Jiamou Liu[1,2,*]

[1]Strong AI Lab, School of Computer Science, The University of Auckland, Auckland, New Zealand
[2]LIU AI Lab, School of Computer Science, The University of Auckland, Auckland, New Zealand

Abstract

Combining deep learning with symbolic logic reasoning aims to capitalize on the success of both fields and is drawing increasing attention. Inspired by DeepLogic, an end-to-end model trained to perform inference on logic programs, we introduce IMA-GloVe-GA, an iterative neural inference network for multi-step reasoning expressed in natural language. In our model, reasoning is performed using an iterative memory neural network based on RNN with a gate attention mechanism. We evaluate IMA-GloVe-GA on three datasets: PARARULES, CONCEPTRULES V1 and CONCEPTRULES V2. Experimental results show DeepLogic with gate attention can achieve higher test accuracy than DeepLogic and other RNN baseline models. Our model achieves better out-of-distribution generalisation than RoBERTa-Large when the rules have been shuffled. Furthermore, to address the issue of unbalanced distribution of reasoning depths in the current multi-step reasoning datasets, we develop PARARULE-Plus, a large dataset with more examples that require deeper reasoning steps. Experimental results show that the addition of PARARULE-Plus can increase the model's performance on examples requiring deeper reasoning depths. The source code and data are available at https://github.com/Strong-AI-Lab/Multi-Step-Deductive-Reasoning-Over-Natural-Language.

Keywords

Gate attention, Out-of-distribution generalisation, Deductive reasoning, Multi-step reasoning

1. Introduction

Symbolic reasoning and deep learning remain two cornerstones in AI with profound yet divergent consequences. Indeed, symbolic approaches, equipped with various logic languages for knowledge representation and inference, have been the dominant paradigm in problem solving and reasoning. Deep learning approaches, through superior ability to capture rich semantic features from complex signals, triumph in tasks that usually require more intuitive and automatic judgements. A growing interest in AI amounts to harnessing the power from both schools, while mitigating each other's weaknesses. First, symbolic reasoning were suitable only when the task at hand, along with all contextual knowledge, can be encoded by rigorous and structured logic expressions, which is itself a formidable obstacle. Then, deep learning relies

NeSy 2022, 16th International Workshop on Neural-Symbolic Learning and Reasoning, Cumberland Lodge, Windsor, UK
*Corresponding author.
qbao775@aucklanduni.ac.nz (Q. Bao); ypen260@aucklanduni.ac.nz (A. Y. Peng); thar011@aucklanduni.ac.nz (T. Hartill); ntan607@aucklanduni.ac.nz (N. Tan); zden658@aucklanduni.ac.nz (Z. Deng); m.witbrock@auckland.ac.nz (M. Witbrock); jiamou.liu@auckland.ac.nz (J. Liu)

Table 1
Logic programs written in Prolog with different reasoning depths. These examples were sampled from [1].

1 Step	2 Steps	3 Steps
$g(L, S)$:- $x(S, L)$.	$x(G, B)$:- $k(G, B)$.	$p(P, R)$:- $b(R, P)$.
$x(a, m)$.	$k(Z, V)$:- $g(Z, V)$.	$b(A, L)$:- $a(A, L)$.
$y(X)$:- $r(X)$.	$g(k, k)$.	$a(W, F)$:- $v(F, W)$.
$p(h)$.	$e(k, s)$.	$v(t, i)$.
$s(t, v)$.	$p(L, G)$:- $v(G, L)$.	$l(D)$:- $t(D)$.
? $g(m, a)$. 1	? $x(k, k)$. 1	? $p(t, i)$. 1
? $g(a, m)$. 0	? $x(k, s)$. 0	? $p(i, t)$. 0

on neural networks which have not demonstrated the ability to perform iterative, multi-step reasoning, which has a gap in making them suitable tools for reasoning. Towards a trainable reasoner that is able to perform complex real-world reasoning tasks, it is important to (1) facilitate end-to-end reasoning by enabling multi-step reasoning and (2) bypass logic-based knowledge representation and make inferences directly from natural language inputs. The goal of this paper is to explore possibilities around these two objectives. We now present the research background in detail.

Logic programs: Reasoning with logic programs is one of the key questions in AI. Here a knowledge base consists of a number of *rules*, i.e., (universally quantified) implications where the antecedent is a conjunction of literals and the consequent is an atom, and observed *facts*, i.e., ground atoms. The task would specify a *question* which is another ground atom and asks if the question logically follows from the knowledge base. Table 1 illustrates several archetypal examples of reasoning tasks in logic programs (as in [1]). The rules, facts, and questions are expressed in predicate logic where variables are capitalised (such as L, S) and constants are in small case (such as a, m). The three columns show logic programs of different reasoning *depths*. For example, the first column contains rule "$g(L, S)$:- $x(S, L)$", fact "$x(a, m)$", and two questions (starting with ?) at the bottom rows. Semantically, the rule expresses that $g(L, S)$ holds whenever $x(S, L)$ holds for any constants L and S. From $x(a, m)$, a simple unification followed by a 1-step forward chaining inference derives $g(m, a)$, which answers the first question positively [2]. On the other hand, the second column contains rules "$x(G, B)$:- $k(G, B)$" "$k(Z, V)$:- $g(Z, V)$", and fact "$g(k, k)$". It takes two forward chaining steps to establish "$x(k, k)$" and thus has depth 2.

DeepLogic: The ability to conduct iterated inference for multi-step reasoning tasks such as the ones above is viewed as an unchallenged strength of rule-based inference algorithms. Yet recent advancements in deep learning techniques have challenged this view. DeepLogic, introduced in [1], is an RNN-based neural network for solving reasoning tasks of logic programs. The model encodes logic programs at the character level and is trained on 12 different types of logic programs, without explicitly applying any symbolic inference algorithm. Through a series of experiments, DeepLogic has demonstrated abilities to handle tasks that require multi-step reasoning (up to a certain small depth).

Reasoning in natural language: The abilities demonstrated by DeepLogic has given hope for similar neural networks to perform more general reasoning tasks. In particular, neural network's key strengths involve the ability to extract rich syntactical and semantic feature from

free-flowing texts, expressed in natural language. Indeed, PARARULES, introduced in [3], is a multi-step reasoning dataset expressed in natural language[1]. Each sample in PARARULES resembles a logic program in the style of Table 1, except that the knowledge base (rules and facts) and questions are expressed in natural language (See Figure 2). We thus aim to explore end-to-end neural-based multi-step reasoners over natural language using PARARULES as a testing platform, while addressing three issues:

(1) Existing models, including DeepLogic and other RNN-based baseline models, have room for improvement in terms of their reasoning abilities over natural language. The vanilla GRU/LSTM model might not handle well the multi-step reasoning tasks over logic programs from [1] and natural language from Table 4. DeepLogic shows that with the help of GRU and dot-product attention, the model can learn to reason over logic programs from [1] and natural language from Table 4. However, DeepLogic does not show the best performance on Table 4. Dynamic memory network with gate attention has shown remarkable performance on the bAbI deductive reasoning task [4]. Our first contribution is to introduce IMA-GloVe-GA, an iterative neural inference network that combined DeepLogic with gate attention, for multi-step reasoning tasks. Our model achieves the best test accuracy among the RNN-based models on the PARARULES dataset. The test accuracy of our model is on average 7.8 percentage points higher than that of DeepLogic (from Table 4. IMA-GloVe-GA is our model, and IMA-GloVe is from DeepLogic).

(2) *Out-of-distribution (OOD) generalisation* means the test set has a distribution that is unknown or different from the distribution of the training set. In multi-step reasoning tasks, OOD generalisation means that (1) the model is able to reason for cases that have a depth greater than the depths of the instances it was trained on, and (2) the model is able to handle samples with shuffled rules from the training instances. Shuffling here means permuting the rules in the knowledge base. Being able to handle OOD is a crucial indicator of a model's reasoning capabilities. In [3], pretrained RoBERTa-Large [5] achieves good performance on the PARARULES dataset (See Table 4). However, it is unclear whether the model indeed performs multi-step reasoning to the extend that it handles OOD test examples. Through a series of experiments, we show that RoBERTa-Large overfits and fails to generalise on examples with shuffled rules. This demonstrates that RoBERTa-Large over-utilises the ordering of rules. On the other hand, our IMA-GloVe-GA outperforms RoBERTa-Large and DeepLogic, when the models are trained on a dataset with fewer examples and unshuffled rules and are tested on a larger dataset with more relations and entities and shuffled rules (See Table 5).

(3) CONCEPTRULES V1 and V2 [6, 7] are natural-language-based multi-step reasoning datasets. Similar to PARARULES, the CONCEPTRULES datasets also contain samples that require deep reasoning steps (depth up to 3), and thus are suitable alternatives when evaluating models' abilities for multi-step reasoning. A common issue with all three existing datasets (PARARULES, CONCEPTRULES V1 & V2), however, lies in their unbalanced distributions over reasoning depths. They have much fewer examples that require deep reasoning (depth ≥ 2) than examples that require shallow reasoning (See Table 2). To address the issue of depth imbalance, we develop a large dataset on multi-step reasoning over natural language called PARARULE-Plus that has a balanced distribution over different reasoning depths. The test accuracy on deeper depths and extra out-of-distribution examples is greatly improved when we

[1] https://allenai.org/data/ruletaker

add PARARULE-Plus in the training process (Table 7 and 8). The experiment result also verifies the necessity of our dataset.

2. Related Work

Systems that integrate deep learning techniques with symbolic reasoning are called *neuro-symbolic systems* [8]. Many such neural reasoning models can be viewed as logical program interpreters. Neural-symbolic machines (NSM) [9] and neural program interpreters (NPI) [10] are such examples. NSM describes a framework that consists of a seq-to-seq neural programmer, a Lisp interpreter to execute the program, and iterative maximum likelihood to train the model; NPI presents an RNN to learn and represent logic programs. NPI is designed for compositional programs, including addition, sorting, and canonicalising 3D models. Reinforcement learning has also been applied to learn Prolog-like algorithms [11]. Distributed representations of predicates and constants for traditional symbolic reasoning engines can be learned by neural theorem provers [12]. In our method, end-to-end neural networks learn representations at word level and learn to reason with natural language.

Several reasoning datasets have been introduced for natural language-based reasoning tasks which can be used to evaluate and compare neural models' reasoning capabilities. Roughly speaking, the datasets can be categorised as "shallow reasoning" and "deep reasoning" tasks. The first category includes Task 15 in the bAbI dataset v1.0 [4], conditional probes in [13], and "multi-hop" reasoning dataset HotpotQA [14]. In these datasets, the reasoning instances usually do not go beyond 2-steps. A typical example in the bAbI dataset would be "Mouse is afraid of cats. Alice is a mouse. What is Alice afraid of? A: cats." and a typical example in the conditional probes of [13] is "If A has visited B, then C has visited D. A visited B. Has C visited D? A: Yes." The main difference of HotpotQA with the other two is that the rules in HotpotQA are embedded in sentences. In HotpotQA, a sentence contains both factual information and rule information. Figure 1 shows a 2-hop example from HotpotQA dataset.

The second category contains instances where the reasoning depth may go beyond 2. These datasets are PARARULES [3], CONCEPTRULES V1 [6] and CONCEPTRULES V2 [7]. Examples in PARARULES may require reasoning depth as deep as 5, while CONCEPTRULES V1 and V2 require depths as deep as 3. PARARULES differs from the three above in the following senses: First, solving the problems in bAbI Task 15 requires implicit rules. For example, "Alice goes to the park. Peter goes to the restaurant. Where is Alice? A: park" requires the rule "A moves to B $\rightarrow$ A at B". Contrary to the bAbI tasks, PARARULES requires reasoning with explicit rules that is more akin to logic programming. Contrary to the HotpotQA dataset, in PARARULES, the facts and rules are separate. One of the main issues of PARARULES, CONCEPTRULES V1 and CONCEPTRULES V2 is the unbalanced distribution over reasoning depths. The datasets have more examples of shallow reasoning (Depth$<$2) than that of deep reasoning (Depth $\geq$ 2).

Other work uses Transformer-based pretrained language models to perform tasks that require multi-step reasoning over natural language. One such work is [3] which demonstrated that the pretrained language models (RoBERTa [5], and BERT [15]) can be used to solve natural language-based reasoning tasks. However, it is unknown whether these pretrained language models would perform better than neural networks specifically designed for these tasks. One of our goals is to compare them against models based on iterative memory mechanism which have been shown to perform well on multi-step reasoning tasks over logic programs.

(*Paragraph A:*) ***LeBron James*** won ***the 2015-2016 NBA Championship***.
(*Paragraph B:*) ***LeBron James*** is a basketball player for ***Cleveland Cavaliers***.
(*Question:*) Which team did the players who won the 2015-2016 NBA Championship play for? (*Answer:*) Cleveland Cavaliers.

Figure 1: A depth-2 example from HotpotQA [14].

3. Problem Definition

We consider multi-step deductive reasoning over natural language. Each sample is a triple (*Context*, *Question*, *Answer*) where *Context* contains natural language implications (rules) and observations (facts) resembling a knowledge base in logic programs, *Question* is a natural language sentence expressing an atomic fact, and *Answer* $\in$ {true, false} tells whether *Question* naturally follows from *Context*. In this regard, (*Context*, *Question*, *Answer*) is a natural-language counterpart to a logic program. Figure 2 illustrates several examples in the PARARULES dataset [3]. The rules are expressed in phrases such as "If A, then B" and "All A are B", and facts are represented using propositions such as "A is happy" and "B is funny".

Multi-step reasoning requires multiple reasoning steps to answer a question. We define the reasoning *depths* (or *steps*) as the number of rules required to answer a question. For example, from "Bob is smart.", it takes only one rule "All smart people are talented." to answer "Bob is talented?". This example is therefore considered as depth-1 reasoning. PARARULES has seven sub-datasets, each is named by the greatest depth of reasoning required to deduce which related facts support its question: depths $D = 1$, $D = 2$, $D = 3$, $D \leq 3$, $D \leq 3$ + NatLang, $D \leq 5$, $D \leq 5$ + NatLang, respectively. Here, NatLang means the extra out-of-distribution examples, containing about 2,000 examples. These examples include questions of different depths. They were created by paraphrasing examples using crowdsourcing. Crowdworkers rewrite part of the synthetic dataset using phrases such as "often", "rather resembles", and "a bit". For instance, sentences like "Charlie is green, but ***often*** kind" and "Harry ***seems*** to be round" are more natural to a human reader.

(*Input Facts:*) Anne is rough. Anne is blue.
(*Input Rules:*) Rule 1: Cold people are rough.
Rule 2: Rough people are young.
Rule 3: If Anne is green then Anne is blue.
Rule 4: If someone is rough and nice then they are green.
Rule 5: If someone is rough and furry then they are blue.
Rule 6: All young people are cold.
Q1: Anne is cold. True/False? **[Answer: T]**
Q2: Anne is not young. True/False? **[Answer: F]**
Q3: Anne is not green. True/False? **[Answer: T]**

Figure 2: Examples from PARARULES [3]. The context (facts + rules) and the question are grouped as the input, and the output is a Boolean value indicating if the question is true or false, given the context.

While PARARULES separates fact and rule explicitly, CONCEPTRULES V1 and V2 [6, 7] put fact and rule together. Each of CONCEPTRULES V1 and V2 has a simplified and a full version. Both CONCEPTRULES V2 (simplified) and CONCEPTRULES V2 (full) include *negation as failure (NAF)* and derivable cases. Derivable means answers can be derived from context and question. Negation as failure means if we cannot find facts or rules to derive the answer,

we assume it is false. The different versions of CONCEPTULES are summarised in Table 3. In CONCEPTRULES V2 (full), the rulesets are randomly shuffled and random textual noise is added. Both CONCEPTRULES V1 and V2 contain examples with reasoning depths from 0 to 3. CONCEPTRULES V1 does not label the reasoning depth for each example, while CONCEPTRULES V2 contains the labels.

4. Method

This section describes a word-level RNN-based iterative neural network. The general idea of the model is borrowed from DeepLogic. DeepLogic is an end-to-end iterative memory attention network trained on symbolic logic programs. For details about DeepLogic, we refer the reader to the original paper [1]. The main differences in the work presented here are that we adapt the DeepLogic model to learning logic expressed in natural language, and the model operates at the word-level instead of character-level. The main architecture of the iteration framework is the same as DeepLogic (shown in Figure 3).

Word-level embedding. The input representation layer of the network takes a sequence of words concatenated from two sentences $w_0^C, \ldots, w_m^C$ and $w_0^S, \ldots, w_n^S$ for context and question respectively.

$$h_t = \mathrm{GRU}\left(\mathrm{GloVe}\left[\mathbf{I}^C_{:,t} + \mathbf{I}^S_{:,t}\right], h_{t-1}\right) \tag{1}$$

The context $\mathbf{I}^C$ and the question $\mathbf{I}^S$ at time step t are embedded by GloVe$[\mathbf{I}^C_{:,t} + \mathbf{I}^S_{:,t}]$, the GloVe [16] word vector representation. GloVe is a set of large-scale pretrained word vectors. We use GloVe instead of the character-level embedding in DeepLogic. In DeepLogic, the logic programs are expressed by symbols using English letters and other characters. However, the logic programs in our settings are expressed in natural language. Representing the programs using word embeddings can better capture the semantic information of natural language. From the other perspective, GloVe uses ratios of co-occurrence probabilities to enlarge or narrow the relationship between words of different or similar meanings [16]. At the same time, Word2Vec uses a local n-gram window to extract information. Furthermore, GloVe achieves better results faster than the other word-level embeddings like Word2Vec on word analogy task [16]. The dimension of the GloVe embedding we used is 1×100. The dimension of a sentence embedding concatenated with 5 words is 5×100. The sentence embedding is processed by the gated recurrent unit (GRU) [17]. Hidden state at time t is denoted as h_t. The context vector is $\mathbf{C} \in \mathbb{R}^{R \times L \times d}$, where R is the number of rules, L is the number of words in the rules and d is the dimension of the embedding.

Iteration. Each iteration step computes the new state based on the current state and rules. In our model, gate attention is trained to decide how much of the state information will be reserved from the current state and the previous state. The process is iterated for T steps (T is pre-determined). The initial state is denoted as $s^0 = q$, where q is the question vector and $q \in \mathbb{R}^d$. The i-th rule is denoted as r_i. W and U are the learnable weight matrices. b is the bias vector.

$$w_i^t = \left[s^t; q; r_i; (s^t - r_i)^2; s^t \odot r_i\right] \tag{2}$$

$$\alpha_i^t = \sigma(W\left(Uw_i^t + b\right) + b) \tag{3}$$

At time step t, we compute a feature vector w_i^t using the current state $s^t \in \mathbb{R}^d$, question vector q and a rule r_i. In (2), [;] is a concatenation operator. We use a feed-forward network to compute

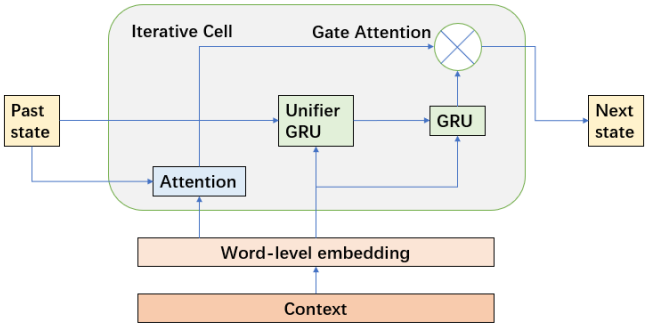

Figure 3: The iterative neural cell with word-level embeddings as input. Questions and contexts are represented as word embeddings, and then attentions are computed to pick up related rules. Gate attention is used to compute the weighted sum of the Unifier GRU outputs. Then the weighted sum updates the state for the next iteration.

the final attention vector α_i^t (3), where σ is a sigmoid function. For experimental comparison, softmax is used for our IMA-GloVe-GA and another baseline model IMASM-GloVe to compute the attention vector α_i^t, instead of the sigmoid function.

$$h_{ij}^t = \underset{R}{\text{GRU}}\left(\mathbf{C}_{ij}, h_{i(j-1)}^t\right) \tag{4}$$

$$s^{t+1} = \sum_i \alpha_i^t h_{iL}^t \tag{5}$$

We use another recurrent neural network to process the context embedding $\mathbf{C}_{ij}$. The initial hidden state $h_{i0}^t = s^t$, where s^t is the current state. For each rule, the new hidden state h_{ij}^t is computed by (4). In the end, the new state s^{t+1} is computed as a weighted sum of the final hidden states in (5). The Unifier GRU learns the unification between pronouns (variables) and nouns (constants).

Gate attention. Dynamic memory network+ [18] achieved 100% test accuracy by using gate attention on the bAbI deductive reasoning task (Task-15). However, Task-15 is not multi-step deductive reasoning as considered in our experiment. Dynamic memory network with gate attention was shown to perform better than the one with the traditional GRU on bAbI tasks [18]. Although gate attention can perform well on Task-15, it is worth exploring applying an iterative neural network with gate attention to multi-step deductive reasoning. This inspired us to integrate gate attention with DeepLogic which is an iterative neural inference network for multi-step reasoning. We use gate attention to replace dot-product attention, and then update the GRU utilizing the output of gate attention. Gate attention can be seen as one of variety solution from gate mechanism and attention. There are two main existing work related to gate attention. The first one is gate mechanism. The gate mechanism (forget gate in LSTM [19]) is designed to address the issue of forgetting in long and short term memory for recurrent neural network. The other one is gate mechanism and attention that DeepLogic shows that with the help of GRU and dot-product attention, the model's performance on the multi-step reasoning tasks can improve a lot from pure GRU without adding dot-product attention. However, in DeepLogic, the dot-product attention is added above on GRU. For each iteration, the attention is used to compute a new state based on the embeddings of the context and question with the previous time stamp state, while the gate in GRU is not specifically trained to keep the

useful information with multi-step reasoning. Then we want to use the gate attention to update the gate with attention-enhanced information from context and question in order to improve multi-step reasoning. By replacing the attention vector (4) with gate attention g_i^t (6), the gate attention is used to update the internal state for GRU as in (7).

$$g_i^t = \frac{\exp\left(W\left(Uw_i^t + b\right) + b\right)}{\sum_{k=1}^{R} \exp\left(W\left(Uw_k^t + b\right) + b\right)} \tag{6}$$

$$h_i = g_i^t \circ \tilde{h}_i + \left(1 - g_i^t\right) \circ h_{i-1} \tag{7}$$

5. The Datasets

To investigate whether end-to-end neural networks designed for symbolic logic reasoning can be adapted to do multi-step deductive reasoning in natural language, we evaluated the models on PARARULES [3], CONCEPTRULES V1 [6], and CONCEPTRULES V2 [7] datasets that require various depths of reasoning. The main difference between the existing datasets used to evaluate DeepLogic and the multi-step natural language reasoning datasets is that the former are based on logic programs like the examples in Table 1, each predicate is represented by a string of symbols and characters. However, natural language is much more diverse and expressive. Both PARARULES and CONCEPTRULES are synthetically generated datasets. PARARULES also includes examples with sentences paraphrased by humans. These examples are more diverse in language and more challenging for the model. Furthermore, to address the issue of depth imbalance in the current multi-step reasoning datasets, we develop a new dataset called PARARULE-Plus, a large multi-step reasoning dataset over natural language. The dataset includes examples of four reasoning depths, from 2 to 5. There are around 100,000 samples for each depth and nearly 400,000 samples in total. The detailed information about PARARULE-Plus can be found in Table 2 and Appendix.

6. Experiments

We experiment with three variants of DeepLogic. The iterative memory attention (IMA) model is adopted from DeepLogic. IMASM is similar to IMA, except that IMASM uses softmax rather than sigmoid when computing attention scores. We also test three baseline models from the bAbI leaderboard: Long short-term memory (LSTM) [20] (The baseline method on bAbI dataset), dynamic memory networks (DMN) [21] (100% test accuracy on bAbI), and memory attention control networks (MAC) [22] (A classical method using memory network). We use GloVe [16] as the word vector representation for the baseline methods, including IMA-GloVe, IMASM-GloVe, MAC-GloVe, DMN-GloVe, and LSTM-GloVe. Our IMA-GloVe-GA uses gate attention instead of the dot-product attention in the IMA-GloVe model.

We train the models on PARARULES, incorporating all reasoning depths (Depth=0, Depth=1, Depth= 2, Depth=3, Depth=5) and the examples paraphrased by humans (NatLang). We set the random seed to 0. We train the models using the Adam [23] optimiser for 30 epochs. After each epoch, the data is reshuffled, and the rules in the context are reshuffled for each mini-batch. The batch size is 32. The maximum iteration depth is set to 4. The initial learning rate for Adam[2] is 1e-02. The latent dimension d for GRU is set to 64. The loss function is binary cross entropy because the task is a binary classification problem. The evaluation measure is accuracy. It is computed as accuracy = $n_{\text{correct}}/n_{\text{total}}$, where n_{correct} is the number of correctly classified examples and n_{total} is the total number of examples. We evaluate the Transformer

[2]Keras Optimizers, https://keras.io/api/optimizers/

baseline model (RoBERTa-Large) from FAIRSEQ [24] that was included the original paper that introduced PARARULES [3]. We follow the official script[3] to fine-tune the model, we use the same hyperparameter in [24] to fine-tune RoBERTa-Large on PARARULES. We use the initial learning rate of 1e-05, and we use 16 as the batch size. We conduct all experiments using the NVIDIA 460.84 Linux Driver. The CPU version is Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz and 16 cores. The CUDA version is 11.2 and a Quadro RTX 8000 with 48 GB GPU memory is used for our experiments.

Experimental Results: Table 4 shows the results of the RNN-based models on test sets that require different reasoning steps. Unless otherwise stated models are trained on the entire training set with all reasoning depths. The first horizontal row denotes test sets with specific reasoning depth(s), and the second horizontal row denotes the number of test samples. We find that our IMA-GloVe-GA and RoBERTa-Large achieve the 2nd and 1st best results. These two models also achieve the 2nd and 1st best results in the test set with additional human-rewritten examples (+NatLang), showing a better generalisation performance and robustness. We find that by adding gate attention the test accuracy improved over IMA-GloVe in all cases. The results support that gate attention can be more effective than dot-product attention on examples that require multi-step reasoning. IMA-GloVe and IMASM-GloVe obtained better performance than the other RNN-based baseline models. We speculate that the dot-product attention enhances the learning and representation between context and question. In addition, the vanilla LSTM fails to converge in all those cases. We get similar results as reported in DeepLogic [1] that the vanilla GRU/LSTM failed on the multi-step reasoning over logic programs. A possible reason is that the vanilla LSTM is more sensitive to hyperparameter tuning. It is harder to train to converge than the other models.

The results shown in Table 4 compare the performances of the IMA models based on the GloVe word embeddings and the pretrained RoBERTa model on the same test sets of different reasoning depths. We find that by adding gate attention the test accuracy improved over IMA-GloVe in all cases. The results support that gate attention can be more effective than dot-product attention on examples that require multi-step reasoning. We also find that the pretrained RoBERTa model achieves better results in all of the cases. A possible reason is that our model is trained from scratch without any large-scale pretraining. However, our model can perform better than the other baselines without any pretraining.

Table 5 shows the results on CONCEPTRULES V1 (simplified) and CONCEPTRULES V1 (full). The second column is the training set CONCEPTRULES V1 (simplified or full). The last two columns show test accuracy on CONCEPTRULES V1 (simplified or full) test set. For example, in the first row, we train IMA-GloVe on the simplified version of the training sets, and then test on different test sets. We find that IMA-GloVe-GA (with gate attention) performs better than IMA-GloVe in all cases. In contrast, the pretrained RoBERTa model only achieves high performance in the simplified test set when it is trained on the simplified training set. It indicates that the pretrained RoBERTa overfits CONCEPTRULES V1 (simplified) since the rules are not shuffled. Most likely it learns spurious relations in the training data and fails to generalise on test examples with shuffled rules.

Table 6 shows results on CONCEPTRULES V2. We select the four models used in Table 5, and

[3]FAIRSEQ, https://github.com/pytorch/fairseq

respectively select data of different depths in CONCEPTRULES V2 (full) as the training data. For example, Mod3 represents a model trained on examples with depth of 3. Mod0123 represents a model trained on examples with depth of 3 and less. Each row of the table represents a test set of a specific reasoning depth. We find that the IMA-based model achieves better results in almost all cases. The pretrained RoBERTa model does not outperform the IMA-based models in all cases. We find that when the model is only trained on examples with reasoning step of 1, the test accuracy dropped as we increase the reasoning steps in the test set. However, the drop in performance is not as great as we expected for the IMA-based models. This shows that the IMA-based models generalise better on out-of-distribution test examples. A possible reason why models achieve higher test accuracies on CONCEPTRULES V2 (full) is that CONCEPTRULES V2 has much more training data compared to PARARULES, from Table 2.

Table 7 shows the results of fine-tuning RoBERTa-Large on datasets with different reasoning depths and testing on test sets of various reasoning depths. Depth≤2 means the model is trained on the dataset with depths no larger than 2. Depth≤3+NatLang represents the model that is trained on the dataset with depths no larger than 3 and extra examples paraphrase by humans. The models trained on the datasets with shallow depths have lower test accuracies on test sets that require deeper reasoning depths. We find that adding training examples that require deeper reasoning steps for Mod0123 improves the results on the test sets with deeper reasoning steps (e.g. Mod0123Nat vs. Mod0123, and Mod012345 vs. Mod0123). Additionally, we find that adding the examples paraphrased by humans improves the performance on test examples with deep reasoning steps and human-paraphrased test examples.

Table 8 shows the results of fine-tuning RoBERTa-Large on the datasets found in Table 7 and our PARARULE-Plus. The addition of PARARULE-Plus during the fine-tuning improves the performance on the examples that require more reasoning steps. The yellow background shows the improvement over the test accuracy reported in Table 7, and the bracket contains the magnitude of the improvement. The bold numbers indicate the highest test accuracy on corresponding test sets. The results support that our PARARULE-Plus addresses the depth imbalance issue of the current datasets and the addition of it during training improves the model's generalisation on the examples that require deeper reasoning steps.

7. Conclusion

We provide insights into an RNN-based iterative memory model that incorporates gate attention on multi-step reasoning over natural language. Instead of using the original GRU and dot-product attention, we integrate gate attention to update hidden states. The experiment results show the model with gate attention achieves generally better performance than the original RNN-based iterative-memory model with dot-product attention and other RNN-based models. The performance of our model is comparable or better than the much larger and pretrained RoBERTa-Large in some scenarios. Furthermore, our model shows better out-of-distribution generalisation performance than the pretained RoBERTa. To address the issue of depth-imbalance in the existing datasets on multi-step reasoning over natural language, we develop a large-scale multi-step reasoning dataset called PARARULE-Plus, with more examples of deep reasoning depths than previous datasets. We find that the performance of the models in our experiments improves when we add PARARULE-Plus in the training, especially on examples that require deeper reasoning depths and extra out-of-distribution examples.

References

[1] N. Cingillioglu, A. Russo, Deeplogic: Towards end-to-end differentiable logical reasoning, in: AAAI-MAKE, 2019.

[2] S. Russell, P. Norvig, E. Davis, Artificial intelligence: A modern approach (3rd (global) ed.), 2016.

[3] P. Clark, O. Tafjord, K. Richardson, Transformers as soft reasoners over language, in: IJCAI, 2020, pp. 3882–3890.

[4] J. Weston, A. Bordes, S. Chopra, T. Mikolov, Towards ai-complete question answering: A set of prerequisite toy tasks, in: ICLR, 2016.

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

[6] T. Hartill, Conceptrules V1 dataset, https://bit.ly/3uVemXG, 2020.

[7] T. Hartill, Conceptrules V2 dataset, https://bit.ly/3PApIIB, 2020.

[8] A. S. d. Garcez, K. B. Broda, D. M. Gabbay, Neural-symbolic learning systems: foundations and applications, 2012.

[9] C. Liang, J. Berant, Q. Le, K. D. Forbus, N. Lao, Neural symbolic machines: Learning semantic parsers on Freebase with weak supervision, in: ACL, 2017, pp. 23–33.

[10] S. Reed, N. De Freitas, Neural programmer-interpreters, arXiv preprint arXiv:1511.06279 (2015).

[11] Z. Jiang, S. Luo, Neural logic reinforcement learning, in: ICML, PMLR, 2019, pp. 3110–3119.

[12] T. Rocktäschel, S. Riedel, End-to-end differentiable proving, in: NIPS, 2017, pp. 3788–3800.

[13] K. Richardson, H. Hu, L. S. Moss, A. Sabharwal, Probing natural language inference models through semantic fragments., in: AAAI, 2020, pp. 8713–8721.

[14] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, C. D. Manning, Hotpotqa: A dataset for diverse, explainable multi-hop question answering, arXiv preprint arXiv:1809.09600 (2018).

[15] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: NAACL-HLT, 2019.

[16] J. Pennington, R. Socher, C. Manning, GloVe: Global vectors for word representation, in: EMNLP, Doha, Qatar, 2014, pp. 1532–1543.

[17] K. Cho, B. van Merriënboer, D. Bahdanau, Y. Bengio, On the properties of neural machine translation: Encoder–decoder approaches, in: SSST-8, 2014, pp. 103–111.

[18] C. Xiong, S. Merity, R. Socher, Dynamic memory networks for visual and textual question answering, in: ICML, PMLR, 2016, pp. 2397–2406.

[19] F. A. Gers, J. Schmidhuber, F. Cummins, Learning to forget: Continual prediction with lstm, Neural computation 12 (2000) 2451–2471.

[20] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (1997) 1735–1780.

[21] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, R. Socher, Ask me anything: Dynamic memory networks for natural language processing, in: ICML, volume 48, 2016, pp. 1378–1387.

[22] D. A. Hudson, C. D. Manning, Compositional attention networks for machine reasoning,

in: ICLR, 2018.
[23] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: ICLR, 2015.
[24] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, M. Auli, fairseq: A fast, extensible toolkit for sequence modeling, in: NAACL-HLT, 2019.

8. Appendix

Following the closed-world assumption, we use the entities from the PARARULES which includes mainly PEOPLE and ANIMALS. We also consider negation, so there are 4 different scenarios in each depth for each category: PEOPLE with negation in the rules, PEOPLE without negation in the rules, ANIMALS with negation in the rules and ANIMALS without negation in the rules. All of the questions can be derived from the contexts and rules. If a question cannot be matched directly from the context, then it can be derived using rules. For the ANIMALS, we consider 14 different animal entities ("the bald eagle", "the tiger", "the bear", "the lion", "the wolf", "the crocodile", "the dinosaur", "the snake", "the leopard", "the cat", "the dog", "the mouse", "the rabbit", "the squirrel"), 7 different animals relationships ("is", "likes", "chases", "needs", "visits", "attacks", "sees"), and 28 different animals attributes ("big", "strong", "awful", "fierce", "heavy", "horrible", "powerful", "angry", "furry", "small", "cute", "lovely", "beautiful", "funny", "dull", "rough", "lazy", "slow", "sleepy", "boring", "tired", "reckless", "kind", "quiet", "round", "nice", "smart", "clever"). For the PEOPLE, we consider 9 different entities ("Anne", "Alan", "Bob", "Charlie", "Dave", "Erin", "Harry", "Gary", "Fiona"), 1 relationship ("is") and 20 people attributes ("wealthy", "smart", "nice", "quiet", "kind", "poor", "dull", "rough", "bad", "sad", "short", "thin", "small", "little", "big", "strong", "high", "old", "young", and "huge").

Require: Type of dataset: animal/people, add negation rule or not: Yes/No,
reasoning depth $d \in 2, 3, 4, 5$, animal name list = ['the bald eagle', 'the tiger', 'the bear', 'the lion', 'the wolf', 'the crocodile', 'the dinosaur', 'the snake', 'the leopard', 'the cat', 'the dog', 'the mouse', 'the rabbit', 'the squirrel'],
people name list = ['Anne', 'Alan', 'Bob', 'Charlie', 'Dave', 'Erin', 'Harry', 'Gary', 'Fiona'],
animal relation list = ['is', 'is not', 'likes', 'chases', 'needs', 'visits', 'attacks', 'sees'],
people relation list = ['is', 'is not'], animal attribute list = ['kind', 'quiet', 'round', 'nice', 'smart', 'dull', 'rough', 'lazy', 'slow', 'sleepy', 'furry', 'small', 'cute', 'lovely', 'beautiful', 'big', 'strong', 'awful', 'fierce', 'heavy'], people attribute list = ['big', 'strong', 'high', 'huge', 'short', 'thin', 'small', 'little', 'wealthy', 'smart', 'nice', 'quiet', 'kind', 'poor', 'dull', 'rough', 'bad', 'sad', 'old', 'young'] total_list = []
for reasoning depth d in 2,3,4,5 **do**
item_list = randomly select 4 animals/people name from animal/people name list

for index in $range(0, len(item\_list))$ **do**
item, item_1, item_2, item_3 = item_list[0], item_list[1, item_list[2], item_list[3]
random shuffle animal/people relation list
if add negation rules == "No" **then**
context, question, label = load the template and fill item, item_1, item_2, item_3 to the subject or object in the template, and we select one of the elements from the animal/people relation list as the verb. For some facts, we add an item, item_1, item_2, item_3 as the subject, and one of the elements from the animal/people attribute list as the object to generate a dataset that only includes the depth = d, which means all questions have the same number of rules to derive the answer.
else
context, question, label = load the template, the template is similar to the above, but add a negation(not) in any rule to generate a dataset which only includes the depth = d
end if
total_list = total_list.append(context, question, label, depth = d)
end for
end for
return total_list

Algorithm 1: PARARULE-Plus data generation

Table 2
Information about the datasets used in this paper. PARARULES has less number of examples that require deep reasoning steps. CONCEPTRULES V2 does not consider reasoning depths greater than 3. The train, dev and test set are already splitted by the author of each dataset.

Dataset		Depth=0	Depth=1	Depth=2	Depth=3	Depth=4	Depth=5
PARARULES	Train	290435	157440	75131	48010	9443	7325
	Dev	41559	22276	10833	6959	1334	1038
	Test	83119	45067	21496	13741	2691	2086
PARARULE-Plus	Train	-	-	89952	90016	90010	90022
	Dev	-	-	16204	16154	16150	16150
	Test	-	-	2708	2694	2704	2692
CONCEPTRULES V2 (full)	Train	2074360	1310622	873748	436874	-	-
	Dev	115148	72810	48540	24270	-	-
	Test	115468	72810	48540	24270	-	-
CONCEPTRULES V2 (simplified)	Train	131646	74136	49424	24712	-	-
	Dev	7166	4116	2744	1372	-	-
	Test	7362	4116	2744	1372	-	-

Table 3
The entity types and relation types for CONCEPTRULES V1 (simplified/full), CONCEPTRULES V2 (simplified/full), PARARULES, and our PARARULE-Plus.

Dataset	#Entity	#Relation	Shuffled Rules	Depth Tag	Derivable	NAF
CONCEPTRULES V1 (simplified)	385	7	No	No	Yes	Yes
CONCEPTRULES V1 (full)	4048	24	Yes	No	Yes	No
CONCEPTRULES V2 (simplified)	385	7	No	Yes	Yes	Yes
CONCEPTRULES V2 (full)	4048	24	Yes	Yes	Yes	Yes
PARARULES	19	4	No	Yes	Yes	Yes
PARARULE-Plus	71	8	No	Yes	Yes	Yes

Table 4
We use GloVe [16] as the word vector representation. We use PARARULES with all depths as the training set for all models and then test them on examples with different reasoning depths (D). Comparison among our IMA-GloVe-GA, IMA-GloVe, MAC-GloVe, DMN-GloVe, IMASM-GloVe, LSTM-GloVe, and RoBERTa-Large on PARARULES test sets with different reasoning depths.

Train ↓; Test →	D=1	D=2	D=3	D≤3	D≤3+NatLang	D≤5	D≤5+NatLang
IMA-GloVe	0.861	0.853	0.830	0.842	0.810	0.792	0.705
MAC-GloVe	0.792	0.776	0.750	0.763	0.737	0.701	0.652
DMN-GloVe	0.846	0.843	0.817	0.827	0.789	0.779	0.666
IMASM-GloVe	0.864	0.855	0.824	0.838	0.801	0.782	0.608
LSTM-GloVe	0.500	0.500	0.500	0.499	0.499	0.500	0.500
IMA-GloVe-GA	**0.950**	**0.943**	**0.919**	**0.927**	**0.883**	**0.879**	**0.741**
RoBERTa-Large	**0.986**	**0.985**	**0.977**	**0.979**	**0.972**	**0.967**	**0.949**

Table 5
IMA-GloVe, IMA-GloVe-GA, and RoBERTa-Large trained on CONCEPTRULES V1 (simplified / full) and tested on different test sets. Rules in CONCEPTRULES V1 Simplified are not shuffled, while CONCEPTRULES V1 full contains randomly shuffled rules. CONCEPTRULES V1 full has larger number of relations and entities than CONCEPTRULES V1 simplified.

Model	Train set	Test accuracy (Simplified Test set)	Test accuracy (Full Test set)
IMA-GloVe	Simplified	0.994	0.729
	Full	0.844	**0.997**
IMA-GloVe-GA	Simplified	**0.998**	**0.747**
	Full	0.851	**0.999**
RoBERTa-Large	Simplified	**0.997**	0.503
	Full	**0.927**	0.995

Table 6
IMA-GloVe, IMA-GloVe-GA, and RoBERTa-Large trained on CONCEPTRULES V2 (full) and tested on test sets that require different depths of reasoning.

Model	Test set	Mod1 Depth=1	Mod2 Depth=2	Mod3 Depth=3	Mod01 Depth≤1	Mod012 Depth≤2	Mod0123 Depth≤3
IMA-GloVe	Depth=1	**0.999**	**0.998**	**0.990**	**0.997**	**0.998**	**0.997**
	Depth=2	**0.998**	**0.999**	**0.988**	**0.995**	**0.998**	**0.997**
	Depth=3	**0.997**	0.998	0.981	**0.991**	0.996	**0.997**
IMA-GloVe-GA	Depth=1	0.993	0.996	0.987	0.987	0.991	**0.997**
	Depth=2	0.993	**0.999**	0.974	0.986	0.991	0.995
	Depth=3	0.988	**1**	**0.994**	0.989	**0.997**	0.994
RoBERTa-Large	Depth=1	0.998	0.975	0.831	0.995	0.975	0.971
	Depth=2	0.997	0.972	0.885	0.993	0.972	0.965
	Depth=3	0.987	0.951	0.984	0.988	0.951	0.936

Table 7
RoBERTa-Large trained on PARARULES with different reasoning depths and tested on test sets that require different depths of reasoning. A bold number indicates the highest accuracy in a test set.

Model	Test set	Mod012 (Depth≤2)	Mod0123 (Depth≤3)	Mod0123Nat (Depth≤3+NatLang)	Mod012345 (Depth≤5)
RoBERTa-Large	Depth=0	**0.971**	0.946	0.968	0.953
	Depth=1	**0.943**	0.907	0.933	0.909
	Depth=2	**0.933**	0.902	0.932	0.902
	Depth=3	0.562	0.902	**0.926**	0.907
	Depth=4	0.481	0.863	**0.904**	0.888
	Depth=5	0.452	0.856	0.916	**0.933**
	NatLang	0.573	0.579	**0.962**	0.594

Table 8
RoBERTa-Large is fine-tuned on examples with different depths from PARARULES and also the entire PARARULE-Plus(PPT), and then is evaluated on test sets that require different depths of reasoning. The yellow background indicates improvement on accuracy after adding our PARARULE-Plus in the training process.

Model	Test set	Mod012 (Depth≤2+PPT)	Mod0123 (Depth≤3+PPT)	Mod0123Nat (Depth≤3+NatLang+PPT)	Mod012345 (Depth≤5+PPT)
RoBERTa-Large	Depth=0	0.946	0.901	0.965	**0.963 (+0.010)**
	Depth=1	0.877	0.847	**0.937 (+0.004)**	0.881
	Depth=2	0.868	0.873	**0.927**	0.839
	Depth=3	0.771 (+0.209)	0.862	**0.904**	0.826
	Depth=4	0.675 (+0.194)	0.852	**0.897**	0.832
	Depth=5	0.661 (+0.209)	0.888 (+0.032)	0.923 (+0.007)	**0.934 (+0.001)**
	NatLang	0.557	0.593 (+0.014)	**0.970 (+0.008)**	0.649 (+0.055)