System 1 + System 2 = Better World: Neural-Symbolic Chain of Logic Reasoning

Wenyue Hua
Department of Computer Science
Rutgers University
New Brunswick, NJ, US
wenyue.hua@rutgers.edu

Yongfeng Zhang
Department of Computer Science
Rutgers University
New Brunswick, NJ, US
yongfeng.zhang@rutgers.edu

Abstract

Logical reasoning is a challenge for many current NLP neural network models since it requires more than the ability of learning informative representations from data. Inspired by the Dual Process Theory in cognitive science — which proposes that human cognition process involves two stages: an intuitive, unconscious and fast process relying on perception called System 1, and a logical, conscious and slow process performing complex reasoning called System 2 — we leverage neural logic reasoning (System 2) on top of the representation learning models (System 1), which conducts explicit neural-based differentiable logical reasoning on top of the representations learned by the base neural models. Based on experiments on the commonsense knowledge graph completion task, we show that the two-system architecture always improves from its System 1 model alone. Experiments also show that both the rule-driven logical regularizer and the data-driven value regularizer are important and the performance improvement is marginal without the two regularizers, which indicates that learning from both logical prior and training data is important for reasoning tasks.

1 Introduction

Current NLP neural network models such as BERT (Devlin et al., 2018), RoBerta (Liu et al., 2019) and many more recent language models have revolutionized how representations and semantic information can be extracted from language, which have led to large improvements on various tasks. One challenge, however, is how to perform logical reasoning, which relies on the informative representations learned from data but also requires reasoning abilities on top of it.

Our paper builds an architecture that explicitly conducts neural logical reasoning inspired by cognitive science theory of the human mind. According to the Dual Process Theory (Sloman, 1996; Gilovich et al., 2002), humans' cognition processes involve two systems: an intuitive, unconscious and fast process called System 1, and a logical, conscious and slow process called System 2. System 1 relies mainly on perception, which is employed to get an intuitive judgment of the current situation and reach an experience-based conclusion. Meanwhile, System 2 is used for more complex logical reasoning processes such as solving numerical problems, logical deduction problems, and even logical and coherent everyday communications. System 1 is unconsciously deployed which generates representations and helps make intuitive judgments. When complex reasoning is needed, System 2 is used to work together with System 1 and reason over the representations from System 1 for decision making. Most existing models mainly focus on the System 1 stage, leveraging gigantic deep neural networks to learn high-quality representations such as language models (Devlin et al., 2018; Brown et al., 2020), vision representations (LeCun et al., 1995; He et al., 2016; Radford et al., 2021) and graph embeddings (Bordes et al., 2013; Grover and Leskovec, 2016; Kipf and Welling, 2017). Based on the Dual Process Theory,

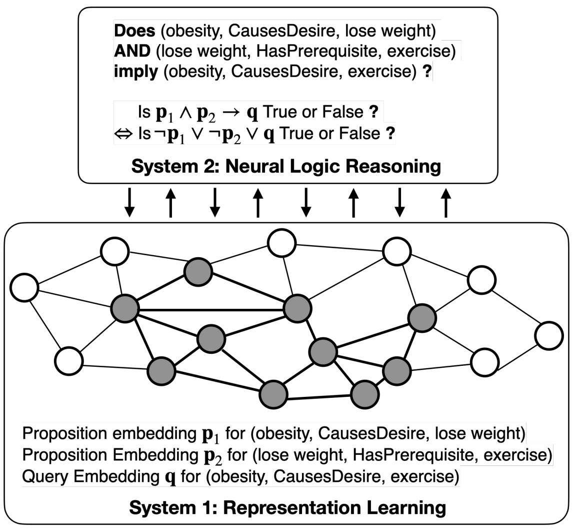

Does (obesity, CausesDesire, lose weight)
AND (lose weight, HasPrerequisite, exercise)
imply (obesity, CausesDesire, exercise) **?**
Is $\mathbf{p}_1 \wedge \mathbf{p}_2 \rightarrow \mathbf{q}$ True or False **?**
$\Leftrightarrow$ Is $\neg\mathbf{p}_1 \vee \neg\mathbf{p}_2 \vee \mathbf{q}$ True or False **?**
System 2: Neural Logic Reasoning
Proposition embedding $\mathbf{p}_1$ for (obesity, CausesDesire, lose weight)
Proposition Embedding $\mathbf{p}_2$ for (lose weight, HasPrerequisite, exercise)
Query Embedding $\mathbf{q}$ for (obesity, CausesDesire, exercise)
System 1: Representation Learning

Figure 1: Overview of the two-system architecture

equipping a reasoning layer that works as the System 2 on top of the representations from System 1 would work better at complex reasoning tasks.

In this paper, we use the Common Sense Knowledge Graph (CSKG) link prediction task to illustrate that models integrating the representation learning ability and the logical reasoning ability perform better than models that only perform representation learning. Technically, our model incorporates logical reasoning with representation learning to enhance the link prediction task based on the observation that a correct proposition (triple) usually has a relevant chain of propositions (triples) that helps validate it. For example, suppose there is a candidate proposition (obesity, CauseDesire, exercise), from these two known propositions (lose weight, HasPrerequisite, exercise) and (obesity, CausesDesire, lose weight), we can infer that (obesity, CauseDesire, exercise) is likely to be valid. If the model can pay attention to the chain of propositions in the knowledge graph to determine, it would be able to improve the graph completion task. A similar idea is introduced by Wei et al. (2022), where the model is trained to generate a chain of thoughts that leads to the final answer in a numerical question answering task. The training of generating explicit reasoning process facilitates the acquisition of reasoning ability in language models.

Our work does not abandon the existing successful representation learning models, instead, we leverage the representations and develop a general architecture that infuses logical reasoning ability on top of the representations for improved performance. The reasoning layer is flexible enough to be plugged into any representation learning model as long as outputs of these models are embeddings. Here are our main contributions: (1) Conceptually, we demonstrate the advantage of applying Dual Process Theory to facilitate the logical reasoning ability and (2) Technically, we develop a neural-symbolic two-system architecture for chain of logic reasoning based on both rule-driven logical regularizers and the data-driven value regularizers.

2 Related Work

2.1 Common Sense Knowledge Graphs

Common Sense Knowledge Graph (CSKG) is a subcategory of knowledge graph where each node is free text and each triple represents a common-sense relation. Extensive efforts have been made in CSKG (Carlson et al., 2010; Sap et al., 2019; Hwang et al., 2021; Bosselut et al., 2019; Nguyen et al., 2021; Mishra et al., 2017; Tandon et al., 2014, 2017; Romero et al., 2019; Romero and Razniewski, 2020), and two representative CSKGs are ConceptNet-100k (Speer et al., 2017) and WebChild (Tandon et al., 2014, 2017). Many efforts have been devoted to building models for knowledge graph link prediction, such as TransE (Bordes et al., 2013), DistMult (Yang et al., 2014), SimplE (Kazemi and Poole, 2018), Complex (Trouillon et al., 2017), HGN (Yan et al., 2021), ConvE (Dettmers et al., 2018), ConvTransE (Shang et al., 2019), NSKGE (Li et al., 2021), NTN (Socher et al., 2013), HypER (Balažević et al., 2019), PGPR (Xian et al., 2019), AttnIO (Jung et al., 2020), CAFE (Xian et al., 2020), DPMPN (Xu et al., 2019) and PLM (Geng et al., 2022). Besides, GCN (Malaviya et al., 2020) leverages a much larger graph neural network to learn representations with the help of transfer learning from pretrained models to enrich representations. These models focus on representation learning and use semantic matching to retrieve the correct tail out of a pool of tails. Our work leverages the representation and infuses logical reasoning ability on top of it for improved performance than representation learning alone.

2.2 Neural Symbolic Reasoning

Neural-symbolic system leverages the representation learning ability of connectionism (the neural system) and the reasoning ability of symbolism (the symbolic system) to effectively integrate learning and reasoning. Various designs of neural-symbolic reasoning models have been presented by researchers (Garcez et al., 2022; Zhang et al., 2021; Moghimifar et al., 2021; Yang et al., 2017). This work adopts the neural logic reasoning (NLR) paradigm, where logical operators such as AND, OR, NOT are learned as neural modules based on self-supervised logic regularization, while inputs to the operators are representation vectors (Shi et al., 2020). The advantage of NLR paradigm is that it can be easily infused into any established representation learning model. NLR helps various tasks such as solving logical equations (Shi et al., 2020), recommender systems (Chen et al., 2021), graph neural networks (Chen et al., 2022a), compositional reasoning (Chen et al., 2022b), analogy learning (Fan and Zhang, 2022), and visual reasoning (Li et al., 2022). Instead of the neighbor-based neural logic reasoning in previous works, we propose a chain of logic reasoning model which provides a clearer reasoning path.

Proposition Triple	The Corresponding Chain of Propositions
(ice cream, AtLocation, freezer)	(ice cream, IsA, food), (food, AtLocation, freezer)
(play lacrosse, HasSubevent, run)	(play lacrosse, SIM, play hockey), (play hockey, MotivatedByGoal, exercise), (exercise, HasSubevent, run)
(bathe, HasPrerequisite, water)	(bathe, UsedFor, get clean), (get clean, HasSubevent, take shower), (take shower, HasPrerequisite, water)
(orange, IsA, colour)	(orange, HasProperty, yellow), (yellow, IsA, colour)
(bath, UsedFor, relaxation)	(bath, AtLocation, your house), (bed, AtLocation, your house), (bed, UsedFor, relaxation)
(classroom, AtLocation, university)	(classroom, UsedFor, education), (education, AtLocation, university)

Table 1: Examples of proposition triples and their corresponding chains

3 Preliminaries: Chain of Propositions

This section introduces the definition of chain used in the model and its coverage in the dataset.

Definition 3.1 (Chain)**.** Let $q = (e_{hq}, rel_q, e_{tq})$ be a query proposition triple of CSKG $G = (E, R)$, where E is the set of entities and R is the set of relations. Notice that q could be an existing triple in G (for training) or a non-existing one that needs to be predicted in inference. An ordered list of propositions $p_1, p_2, \cdots, p_n$ where $p_i = (e_{hi}, rel_i, e_{ti})$ is a chain for q if and only if: (1) for each p_i, $e_{hi}, e_{ti} \in E$ and $rel_i \in R$, (2) $p_1 = (e_{hq}, rel_1, e_{t1})$ and $p_n = (e_{hn}, rel_n, e_{tq})$ for some $e_{t1}, e_{hn} \in E$, and (3) for each p_i, $e_{hi} = e_{t(i-1)}$.

The above is a very strict definition for propositional reasoning chain. To increase coverage, we apply two extensions to the above definition.

3.1 Extension 1: Reversed Relations

To cover propositions such as (ferret, AtLocation, pet store) which is supported by a potential list of triples (ferret, IsA, mammal), (cat, IsA, mammal), (cat, AtLocation, pet store), we extend the definition to allow reversed links. We can rewrite the reversed link (e_t, rel, e_h) as (e_h, rel^{-1}, e_t). Based on this, the definition of chain remains unchanged except that for each triple (e_{hi}, rel$_i$, e_{ti}) in the chain, rel$_i \in R \cup R^{-1}$.

3.2 Extension 2: Graph Densification

CSKG is notorious for the sparsity, for example, 81% of the entities occur only once in ConceptNet-100k. One of the reasons is that free-formed text can differ even though they refer to the same entity or event. For example, “watch movie” and “watch film” are two separate entities in the dataset while they refer to the same event. Such sparsity imposes difficulty on generating chains, which requires densification of the graph. One method is to find similar pairs of entities and generate new triples using a newly created relation “SIM”, so that we can add two new triples (watch movie, SIM, watch film) and (watch film, SIM, watch movie) to the dataset.

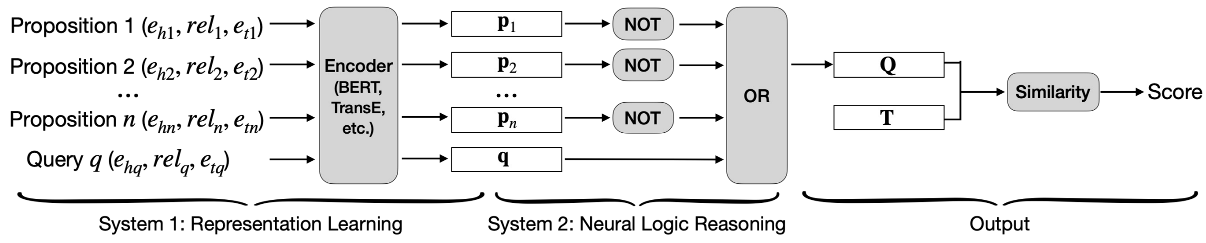
Dataset	Train	Validation	Test
ConceptNet-100k	22.5%	70.42%	78.83%
WebChild-comparative	46.88%	41.58 %	43.75%

Table 2: Statistics of coverage on two datasets

We extend the knowledge graph with these triples by computing the similarity between entity embeddings as in Malaviya et al. (2020). To form these edges, we fine-tune a BERT model on sentences transformed from each triple using simple heuristic rules. Then we extract node representations and use these representations to compute the cosine similarity between all pairs of nodes in the graph: the format of the input to the model is [CLS] + e_phrase + [SEP], where e_phrase is the free-formed text of an entity node. The embedding of each node is the embedding of the [CLS] token. Upon computing the pairwise cosine similarities, we use a threshold τ to filter the pairs of nodes that are most similar. We use $\tau = 0.955$ and create 87,292 more triples on the dataset ConceptNet-100k. This extension is only applied on ConceptNet-100k due to its sparsity.

3.3 Coverage of Chains

For each proposition in the training, validation and testing dataset, we exhaustively compute all of its chains with length within four. Table 2 shows the coverage of chains on the ConceptNet-100k (Speer et al., 2017) and WebChild (Tandon et al., 2014, 2017) datasets, where coverage is the percentage of known propositions in the graph that are accompanied by at least one chain. Each proposition may have multiple chains and the model selects the shortest chain during training and evaluation. Table 1 lists some example chains. We will see in the experiments that chain-based neural logical reasoning significantly helps link prediction.



Proposition 1 (e_{h1}, rel_1, e_{t1})
Proposition 2 (e_{h2}, rel_2, e_{t2})
...
Proposition n (e_{hn}, rel_n, e_{tn})
Query q (e_{hq}, rel_q, e_{tq})
Encoder (BERT, TransE, etc.)
$\mathbf{p}_1$
$\mathbf{p}_2$
...
$\mathbf{p}_n$
$\mathbf{q}$
NOT
NOT
NOT
OR
$\mathbf{Q}$
$\mathbf{T}$
Similarity
Score
System 1: Representation Learning
System 2: Neural Logic Reasoning
Output

Figure 2: Overview of the model structure, where System 1 learns representations and System 2 conducts logical reasoning. The final output vector $\mathbf{Q}$ is compared with the constant true vector $\mathbf{T}$ to decide the final output.

4 The Two-System Architecture

Our architecture consists of two components: a representation learning component and a neural logic reasoning component on top of it, as shown in Figure 2. The representation learning component can be any model that encodes proposition triples into vector embeddings, and the neural logic reasoning component conducts reasoning in a latent logical space based on neural logical operators.

Given a query proposition q and its chain $\{p_1, p_2, \cdots, p_n\}$, the model determines whether q is valid by determining whether the following logical expression is True or False:

$$p_1 \wedge p_2 \wedge p_3 \cdots \wedge p_n \rightarrow q \qquad (1)$$

According to the definition of material implication $(\rightarrow)$[1], the above expression can be transformed to:

$$\neg p_1 \vee \neg p_2 \vee \neg p_3 \cdots \vee \neg p_n \vee q \qquad (2)$$

where each variable is represented as an embedding in $\in \mathbb{R}^d$. Since the above expression only involves negation and disjunction, the neural logic component involves only these two logical operators. We use a multi-layer feed-forward neural network with GELU activation function to instantiate both logical operators as neural modules. The negation operator $\mathbf{N}(\cdot)$ is a unary operator: it takes in an embedding and outputs an embedding of the same dimension. The disjunction operator $\mathbf{D}(\cdot, \cdot)$ is a binary operator: it takes in a concatenation of two embeddings and outputs one embedding in $\mathbb{R}^d$. The overall embedding $\mathbf{Q}$ of the whole expression is computed by applying the two neural logic operators over the proposition embeddings and the query embedding, as shown in Figure 2. Notice that the model architecture is dynamic which builds different computational graphs for different input logical expressions since the number of prerequisite propositions could be different. To evaluate whether the whole expression is true, the model computes the similarity between the final expression embedding $\mathbf{Q}$ and a predefined constant true vector $\mathbf{T}$. $\mathbf{T}$ is defined as an all-ones vector, functioning as the anchor vector of the logical space.

[1]Material implication $(\rightarrow)$ can be represented by basic operations: $x \rightarrow y \Leftrightarrow \neg x \vee y$

4.1 Logical and Value Regularizers

For now the disjunction and negation modules are just multi-layer neural networks. To ensure that they are indeed performing the expected logical reasoning operations, and that the known expressions are correctly embedded into the logical reasoning space, we employ two regularizers:
(1) The self-supervised logical regularizers for the disjunction and negation modules. They make sure that the modules satisfy a collection of basic logical rules shown in Table 3. The regularizers are applied over all expression embeddings $\mathbf{v} \in V$ (including the intermediate expressions) appeared during the process of computing the score for a triple, which force all the logical rules to be satisfied.
(2) The supervised value regularizers for expressions of known values, which guarantee that the embeddings of the intermediate expressions—whose true/false value is known—are close to their deserved true ($\mathbf{T}$) or false ($\mathbf{F} = \mathbf{N}(\mathbf{T})$) embeddings. The value regularizer enriches the model training by fusing the ground-truth true/false information into the intermediate expression embeddings.

Here is an example: Given a query triple q (obesity, CausesDesire, exercise) which has a chain of length two: proposition p_1 (obesity, CausesDesire, lose weight) and p_2 (lose weight, HasPrerequisite, exercise), the model evaluates whether $p_1 \wedge p_2 \rightarrow q$ is valid by computing the similarity score between $\mathbf{T}$ and $\mathbf{Q} = \mathbf{D}(\mathbf{D}(\mathbf{N}(\mathbf{p}_1), \mathbf{N}(\mathbf{p}_2)), \mathbf{q})$, where the latter computes the final embedding for this expression. The logical regularizer requires that all expression embeddings $\mathbf{v} \in V$ that appeared in the calculation process (i.e., $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{q}$, $\mathbf{N}(\mathbf{p}_1)$, $\mathbf{N}(\mathbf{p}_2)$, $\mathbf{D}(\mathbf{N}(\mathbf{p}_1), \mathbf{N}(\mathbf{p}_2))$ and $\mathbf{D}(\mathbf{D}(\mathbf{N}(\mathbf{p}_1), \mathbf{N}(\mathbf{p}_2)), \mathbf{q})$) satisfy the basic logical requirements in Table 3,

NOT	Negation	$\neg v \neq v$	$r_1 = \sum_{\mathbf{v} \in V \cup \{\mathbf{T}\}} Sim(\mathbf{N}(\mathbf{v}), \mathbf{v})$
	Double Negation	$\neg(\neg v) = v$	$r_2 = \sum_{\mathbf{v} \in V \cup \{\mathbf{T}\}} 1 - Sim(\mathbf{N}(\mathbf{N}(\mathbf{v})), \mathbf{v})$
OR	Identity	$v \vee \mathbf{F} = v$	$r_3 = \sum_{\mathbf{v} \in V \cup \{\mathbf{T}\}} 1 - Sim(\mathbf{D}(\mathbf{v}, \mathbf{N}(\mathbf{T})), \mathbf{v})$
	Annihilator	$v \vee \mathbf{T} = \mathbf{T}$	$r_4 = \sum_{\mathbf{v} \in V \cup \{\mathbf{T}\}} 1 - Sim(\mathbf{D}(\mathbf{v}, \mathbf{T}), \mathbf{T})$
	Idempotence	$v \vee v = v$	$r_5 = \sum_{\mathbf{v} \in V \cup \{\mathbf{T}\}} 1 - Sim(\mathbf{D}(\mathbf{v}, \mathbf{v}), \mathbf{v})$
	Complementation	$v \vee \neg v = \mathbf{T}$	$r_6 = \sum_{\mathbf{v} \in V \cup \{\mathbf{T}\}} 1 - Sim(\mathbf{D}(\mathbf{v}, \mathbf{N}(\mathbf{v})), \mathbf{T})$

Table 3: Self-supervised logical regularizers over the logical modules

while the value regularizer requires the intermediate expressions to be close to their deserved true or false embeddings, e.g., $\mathbf{p}_1$ should be close to $\mathbf{T}$ since it is a known triple in the graph, while $\mathbf{N}(\mathbf{p}_1)$ should be close to $\mathbf{F}$ due to the negation, and similar for other intermediate expressions.

If a query triple q has no corresponding chain, then we evaluate the true/false value of the simple expression $\mathbf{T} \rightarrow q$ to decide if q is true or false.

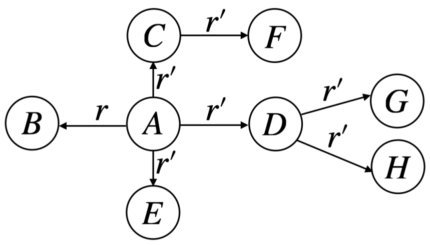
4.2 Learning Proposition Embeddings

Following common practice (Bosselut et al., 2019; Malaviya et al., 2020), we use transfer learning to enhance representation learning from language models. Transfer learning from language model to knowledge graph has been shown effective for commonsense knowledge graph completion (Bosselut et al., 2019). Same as (Malaviya et al., 2020), we fine-tune a BERT-large (Devlin et al., 2018) model on phrases of all nodes in the knowledge graph using masked language modeling. The rich semantics from the language model enhances the node representations. We initialize the node embeddings in ConceptNet-100k, and both node embeddings and relation embeddings in WebChild (since there are more than 6k relations) using fine-tuned models.

4.3 Training by Negative Sampling

The model is trained using negative sampling where each negative example consists of a negative triple and its chain. For each head entity e_h and a relation rel, we sample k tails e_t' from the set of all entities to construct k triples (e_h, rel, e_t') that do not belong to the knowledge graph. However, computation of chains for randomly selected negative triples takes a long time. Thus, instead of randomly sampling tail entities, we sample negative tails from those entities that have a chain from the head entity.

Figure 3 is an example. This small graph G has nodes $\{A, B, C, D, E, F, G, H\}$ and relations r, r'. Suppose (A, r, B) is the gold triple. Our goal is to create negative triples (A, r, e_t') that do not exist



$$\begin{tikzcd} & C \arrow[r, "r'"] & F & \\ B & A \arrow[l, "r"'] \arrow[u, "r'"] \arrow[r, "r'"] \arrow[d, "r'"] & D \arrow[r, "r'"] \arrow[dr, "r'"] & G \\ & E & & H \end{tikzcd}$$

Figure 3: Example graph to illustrate negative sampling

in the graph. We first create a negative tail entity set E_t' which is initialized as $\emptyset$. Then, randomly sample k entities e_t' that is connected to the head entity by a chain of length one and that (A, r, e_t') does not belong to the graph. In this example, nodes $\{C, D, E\}$ are added to the set E_t'. Then for each sampled entity, we randomly sample another k entities e_t' that is connected to it by a chain of length one and that (A, r, e_t') does not belong to the graph. Again, nodes $\{F, G, H\}$ are added to the set E_t'. We sample iteratively four times which gives us negative tail entities within four hops of the head entity and they are added into the set E_t'. We then randomly sample another k entities from the rest of the entities without a chain and add them to set E_t'. Finally, we randomly sample k entities from the set E_t' and k negative triples (A, r, e_t') are created for the positive triple (A, r, B). We refer to the new sampling method as chain sampling.

4.4 Model Optimization

The loss consists of three parts. The first is a contrastive loss: for each triple q in the training dataset, we sample k negative tails to create k negative triples q'. The model computes the logical expression embedding $\mathbf{Q}$ for q and $\mathbf{Q}'$ for all q's based on their chains. The contrastive loss maximizes the difference between the score s of $\mathbf{Q}$ and $\mathbf{Q}'$:

$$L_c = \sum_{q \in G} \sum_{q' \in neg(q)} \log \sigma(\alpha \cdot (s_{\mathbf{Q}} - s_{\mathbf{Q}'})) \quad (3)$$

where α is the amplifying parameter, $\sigma(\cdot)$ is the sigmoid function, s is the cosine similarity score between $\mathbf{Q}$ (or $\mathbf{Q}'$) and $\mathbf{T}$, and neg(q) is the set of negative triples for the positive triple q.

The second part minimizes the distance between $\mathbf{Q}$ and $\mathbf{T}$ as well as the distance between $\mathbf{Q}'$ and $\mathbf{F}$, which encodes the ground-truth supervision signal into the learned proposition embeddings:

$$L_d = \sum_{q \in G} \sum_{q' \in neg(q)} \|\mathbf{Q} - \mathbf{T}\|_2^2 + \|\mathbf{Q}' - \mathbf{F}\|_2^2 \quad (4)$$

The third part consists of two regularizers: logical regularizer L_l and value regularizer L_v:

$$\begin{aligned} L_l &= r_1 + r_2 + r_3 + r_4 + r_5 + r_6 \\ L_v &= \sum_{v \in V} \mathbb{I}_{v=T}\|\mathbf{v} - \mathbf{T}\|_2^2 + \mathbb{I}_{v=F}\|\mathbf{v} - \mathbf{F}\|_2^2 \end{aligned} \quad (5)$$

where V is the set of all intermediate expressions during the model calculation (see Section 4.1) and $\mathbb{I}$ is an indicator function whose value is 1 if the condition holds and 0 otherwise.

The final loss function is:

$$L = L_c + L_d + \lambda_l \cdot L_l + \lambda_v \cdot L_v \quad (6)$$

where λ_l is the weight for logical regularization and λ_v is the weight for value regularization.

4.5 Inference

Each model is evaluated in two inference settings: (1) retrieval out of 1,000 randomly sampled candidates and (2) retrieval of 1,000 candidates based on chain sampling. We explore the two inference settings because we noticed that the distribution of negative triples in random sampling (which is used in previous works) and in chain sampling are different: randomly sampled negative triples are in general much easier to be distinguished than negative triples based on chain sampling, since the chain-sampled negatives are closer to the corresponding positive triple. As a result, chain-sampling presents a more challenging task that is worth study. For random-sampling setting, we first compute the chains for each triple q by Breadth First Search; if no chain is found, $\mathbf{T}$ is used as the null chain. In chain-sampling setting, each triple already has its chain which is used to inference the final expression score. We rank the triples based on the scores and select the top-ranked triples for evaluation.

5 Experiments

5.1 Datasets

We experiment with two prominent commonsense knowledge graphs: ConceptNet-100k (Speer et al., 2017) and WebChild (Tandon et al., 2014, 2017).
ConceptNet-100K contains general commonsense facts of 78,093 entities and 34 relations. The entity has 2.85 words on average. We use the original splits of the dataset and combine the two provided validation sets to create a larger validation set. The validation and test sets has 1,200 triples each.
WebChild is a large collection of commonsense knowledge from the Web. We take the Webchild-comparative dataset which contains 800k comparisons among 576k entities. We randomly select 1,200 triples for validation and another 1,200 for testing and use the remaining for training.

5.2 Baselines and Evaluation Metrics

We take fine-tuned BERT-large (Devlin et al., 2018) to initialize the node embeddings (see Section 4.2). Since our method is a general framework that can be applied on existing CSKG completion methods, we apply our Neural Logic (NL) layer (System 2) on top of the following methods (System 1) to see if adding the System 2 reasoning layer can help to improve the System 1 performance:

Neural Tensor Network (NTN) (Socher et al., 2013), which uses a bilinear tensor layer to learn how head, relation and tail embeddings interact across multiple dimensions. **DistMult** (Yang et al., 2014), which is an embedding-based bilinear diagonal model to learn entity and relation embeddings. **SimplE** (Kazemi and Poole, 2018), which decomposes an entity's embedding by two vectors, each capturing the entity's behaviour as the head or as the tail of a relation. A relation's embedding is decomposed by two vectors: itself and its reverse. **ConvE** (Dettmers et al., 2018), which uses 2D convolution over entity and relation embeddings and multiple layers of nonlinear features to model knowledge graphs. **ConvTransE** (Shang et al., 2019), which is a model built upon the ConvE model but additionally models the translational properties of TransE (Bordes et al., 2013). **HypER** (Balažević et al., 2019), which uses a hypernetwork (one network generates weights for another network) to generate convolutional filter weights based on each relation to process the input entities.

Since the neural logic layer works on the triple embeddings while the outputs of the above baselines are usually scores, we minimally modify the baseline models such that its output is an embedding for each triple. The triple embedding is either directly used to calculate its score by computing

Dataset	ConceptNet-100k						WebChild-comparative					
Metric	**MRR↑**	**Hits@1↑**	**@3↑**	**@10↑**	**@100↑**	**MR↓**	**MRR↑**	**Hits@1↑**	**@3↑**	**@10↑**	**@100↑**	**MR↓**
Unit	%	%	%	%	%	1	%	%	%	%	%	1
Bert+DistMult	38.90	29.92	44.52	55.50	71.00	146.48	21.98	15.25	24.08	34.50	59.25	178.10
Bert+DistMult*	53.90	42.50	61.83	73.67	**91.33**	**39.56**	47.68	37.50	53.08	67.33	88.08	46.61
Bert+DistMult*+NL	**66.22**	**59.08**	**71.58**	**77.05**	87.31	71.72	**61.00***	**50.75***	**69.28***	**81.52***	**94.04***	**24.20***
Bert+ConvE	59.00	45.25	65.83	85.50	**98.58**	10.51	9.54	8.42	9.50	10.00	22.25	417.03
Bert+ConvE*	58.15	48.33	54.38	70.78	87.99	58.59	11.79	10.58	11.83	12.97	23.08	427.45
Bert+ConvE*+NL	**76.23**	**54.73**	**84.63**	**91.73**	97.19	**8.01**	**20.45**	**25.07**	**38.40**	**49.43**	**61.54**	**216.22**
Bert+ConvTransE	71.57	62.25	79.17	86.25	94.08	12.73	21.39	12.58	24.67	38.58	69.67	118.21
Bert+ConvtransE*	71.72	63.42	77.42	79.57	91.58	26.19	21.92	13.42	24.33	37.92	69.00	120.00
Bert+ConvTransE*+NL	**83.00***	**75.88***	**89.47***	**94.42***	**99.12***	**4.98***	**24.21**	**25.28**	**36.45**	**51.82**	**77.36**	**82.42**
Bert+HypER	65.26	51.75	**75.25**	**90.33**	**97.83**	**8.77**	32.32	22.92	35.08	51.25	81.92	72.48
Bert+HypER*	55.21	41.58	63.17	82.75	97.08	13.30	36.45	26.00	41.67	55.17	84.58	61.24
Bert+HypER*+NL	**65.78**	**56.73**	72.00	81.84	92.92	33.38	**50.56**	**37.56**	**60.78**	**74.04**	**92.29**	**31.64**
Bert+SimplE	25.15	12.16	28.83	53.58	91.00	35.10	10.66	6.08	11.50	19.17	46.17	254.80
Bert+SimplE*	24.64	12.00	28.58	51.08	90.42	38.22	13.77	7.50	14.50	25.58	58.42	166.88
Bert+SimplE*+NL	**38.34**	**27.12**	**40.93**	**61.88**	**94.54**	**22.70**	**31.65**	**23.75**	**33.25**	**48.79**	**75.21**	**97.56**
Bert+NTN	18.49	7.08	20.50	42.33	89.08	40.09	13.74	11.16	14.50	17.50	30.00	415.58
Bert+NTN*	26.98	12.25	30.42	59.25	**94.58**	**26.81**	4.02	1.50	4.08	9.50	30.17	436.56
Bert+NTN*+NL	**28.99**	**14.25**	**38.19**	**65.65**	92.33	28.45	**22.51**	**15.04**	**26.33**	**36.18**	**45.52**	**228.24**

Table 4: Evaluation using random sampling on ConceptNet-100k and WebChild-comparative

its similarity with the constant $\mathbf{T}$ vector (denoted as baseline*) or routed through the Neural Logic layer (System 2) to get the final expression embedding and score (denoted as baseline*+NL). We experiment the baseline models with open-source implementations.[2]

We evaluate the performance of retrieving the correct tail given a head and a relation. The evaluation metrics include Mean Reciprocal Rank (MRR), Mean Rank (MR) and Hits@k among 999 negative tails plus the gold tail for k in $\{1, 3, 10, 100\}$. The implementation details are provided in Appendix.

5.3 Main Results

Table 4 and Table 5 show the results on random sampling setting and chain sampling setting, respectively. The best number in each base model is bolded, and the overall best value in each column is starred. By comparing Bert+baseline with Bert+baseline*+NL in Table 4 and 5, we can see that adding System 2 reasoning layer on top of System 1 representation learning layer almost always improves the performance on both datasets, especially on WebChild-comparative which has no exception at all. Besides, in most cases, the global best performance on each metric is achieved by the System 2 enhanced model which has the neural logic reasoning layer.

By comparing results in Table 4 and Table 5, we can see that the performance of all models dramatically decreases under chain sampling based evaluation. This indicates that the negative triples with tail entity close to the head entity are more difficult to distinguish than random negative triples. Retrieval on chain sampling poses a much more challenging problem than retrieval on random sampling. However, Bert+baseline*+NL still achieves improvements in both datasets under most scenarios except for ConvtransE and HypER on ConceptNet-100k.

5.4 Ablation Study

The model performs neural logical reasoning by learning the logical operators NOT and OR using neural modules. The model uses logical regularizer to help the learning of logical operators, and uses value regularizer to facilitate the learning of whether an expression is true or false. To study the effectiveness of the two regularizers, we present the Bert+ConvTransE*+NL model as an example for the ablation study and other models have similar results. We compare the following four versions of the model: model without either regularizer (no reg.), model with only logical regularizer (logical reg.), model with only value regularizer (value reg.), and model with both regularizers (both reg.).

Table 6 reports the results. The model without any regularizer already performs better than the baseline model, showing that simply providing extra triple information is helpful. The performance is better with either the logical regularizer or the value regularizer, with the model value reg. performing slightly better than the model logical reg., showing that explicitly infusing the logical reasoning ability during the model computation for every step helps the prediction, either by directly enforcing the true/false value of each intermediate expression or enforcing the negation and disjunction operator to function logically. The two regularizers

[2]https://github.com/TimDettmers/ConvE, https://github.com/ibalazevic/HypER

Dataset	**ConceptNet-100k**						**WebChild-comparative**					
Metric	**MRR↑**	**Hits@1↑**	**@3↑**	**@10↑**	**@100↑**	**MR↓**	**MRR↑**	**Hits@1↑**	**@3↑**	**@10↑**	**@100↑**	**MR↓**
Unit	%	%	%	%	%	1	%	%	%	%	%	1
Bert+DistMult	15.42	7.23	16.93	32.36	75.06	95.83	0.96	0.00	0.42	1.75	14.08	437.66
Bert+DistMult*	21.58	9.41	25.08	47.83	82.50	93.77	8.61	4.25	8.25	16.42	51.00	197.72
Bert+DistMult*+NL	**31.88**	**20.01***	**38.39**	**60.08**	**92.15**	**42.18**	**31.13**	**17.61**	**42.96***	**52.82***	**78.87***	**69.85***
Bert+ConvE	27.03	12.58	32.58	54.92	93.83	29.78	2.54	0.75	2.50	5.17	17.58	451.19
Bert+ConvE*	**29.28**	12.25	**33.00**	54.67	**96.83***	29.55	2.12	0.58	1.92	4.30	16.08	475.91
Bert+ConvE*+NL	26.11	**14.12**	32.20	**55.51**	96.15	**23.08**	**34.47***	**27.63***	**39.47**	**47.37**	**68.42**	**107.34**
Bert+ConvtransE	**27.65**	15.43	**35.36**	**58.55**	93.49	25.07	8.77	4.25	8.33	17.33	46.58	245.41
Bert+ConvtransE*	26.36	11.33	31.92	58.42	**95.08**	**24.82**	6.85	3.17	6.42	13.58	41.75	276.07
Bert+ConvtransE*+NL	26.96	**16.85**	32.83	51.62	91.14	29.02	**24.90**	**17.52**	**27.43**	**39.24**	**57.90**	**145.95**
Bert+HypER	**33.63***	17.58	**40.58***	**67.25***	96.00	**20.16***	6.90	2.58	6.75	15.33	45.08	220.93
Bert+HypER*	30.54	14.16	38.08	63.50	**96.13**	21.36	6.26	2.17	6.00	13.08	42.25	221.28
Bert+HypER*+NL	30.07	**18.00**	40.30	53.82	91.94	28.36	**17.18**	**9.79**	**20.98**	**29.37**	**59.44**	**151.28**
Bert+SimplE	8.53	3.58	9.17	17.33	48.67	180.68	8.93	4.83	9.42	15.33	42.42	272.08
Bert+SimplE*	6.03	2.08	5.58	12.92	44.41	230.13	11.16	5.92	11.33	20.58	51.83	217.49
Bert+SimplE*+NL	**15.18**	**7.58**	**15.67**	**30.58**	**75.08**	**87.69**	**26.62**	**11.44**	**22.72**	**33.28**	**59.25**	**172.10**
Bert+NTN	0.00	0.00	0.00	0.00	0.00	501.10	3.25	1.42	2.92	7.17	16.42	562.61
Bert+NTN*	2.31	0.50	1.42	4.33	30.08	285.50	0.81	0.08	0.42	1.58	9.42	609.86
Bert+NTN*+NL	**10.98**	**5.75**	**11.42**	**21.83**	**51.33**	**171.94**	**27.20**	**10.11**	**15.37**	**20.93**	**35.04**	**385.14**

Table 5: Evaluation using chain sampling on ConceptNet-100k and WebChild-comparative

together improves the model even more, showing that regularizing the logical operators and regularizing the true/false value of intermediate expressions are both important and helpful.

Model	**MRR↑**	**Hits@1↑**	**@3↑**	**@10↑**	**@100↑**	**MR↓**
baseline	71.57	62.25	79.17	86.25	94.08	12.73
no reg.	74.40	66.21	86.27	92.57	97.26	7.76
logical reg.	79.21	70.98	86.64	92.23	98.14	7.14
value reg.	80.24	70.61	88.85	93.82	98.14	7.40
both reg.	**83.00**	**75.88**	**89.47**	**94.42**	**99.12**	**4.98**

Table 6: Ablation study result on ConceptNet-100k

5.5 Parameter Sensitivity for Regularization

The default values for the regularizer weights λ_l and λ_v in Eq.(6) are both 0.5. In this section, we try multiple values $\lambda_l, \lambda_v \in \{0.01, 0.05, 0.1, 0.5, 1, 5\}$ for each of the two regularizer weights respectively while holding the other weight as the default value. The MRR of each experiment is shown in Figure 4. We can see that (1) too small λ and too large λ both show negative effect while the latter decreases performance much more, and (2) too large λ of the logical regularizer shows worse performance than that of the value regularizer.

5.6 Qualitative Analysis

We analyze some error cases and find three error types: (1) Wrong reasoning conducted on the chain of propositions; (2) Correct reasoning conducted on the chain but the chain contains a mistaken triple which leads to a wrong answer. For example, (fish, AtLocation, at beach) is invalid, but the provided chain (fish, HasPrerequisite, water), (water, AtLocation, at beach) validates it. This mistake is made due to the wrong triple (water, AtLocation,

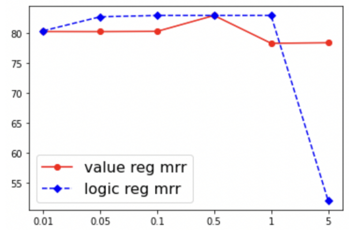


Figure 4: Parameter Sensitivity on ConceptNet-100k

at beach); (3) Correct reasoning conducted on the chain containing an imprecise triple which leads to a wrong or unnatural triple. For example, it is questionable whether (door, AtLocation, street) is correct, but given the chain (door, PartOf, car), (car, AtLocation, street), it is a correct conclusion. A more natural triple would be considered correct if we replace "door" by "car door" here. This shows that some ground-truth triples are not precise enough to conduct valid logical reasoning. Some examples of the three error types and false negative cases on ConceptNet-100k are presented in Table 7 and Table 8 of the Appendix, respectively.

6 Conclusions and Future Work

As a simple instantiation of the Dual Process Theory, this paper demonstrates the advantage of incorporating a System 2 reasoning model on top of a System 1 representation learning model on CSKG link prediction tasks. The positive results verify the potential of our method. As a flexible reasoning model that is differentiable and that can be easily infused with any representation learning model, we intend to incorporate our method with larger pretrained language models and extend to more NLP tasks in the future.

Error Type 1	
(go to bed, HasPrerequisite, take book off shelf)	(go to bed, HasLastSubevent, read book), (read book, HasPrerequisite, take book off shelf)
(book, UsedFor, print)	(book, HasA, paper page), (paper page, UsedFor, print)
(glass, UsedFor, hang clothe)	(glass, AtLocation, closet), (closet, UsedFor, hang clothe)
Error Type 2	
(fish, AtLocation, at beach)	(fish, HasPrerequisite, water), (water, AtLocation, at beach)
(think, Causes, feel sad)	(think, UsedFor, discover truth), (discover truth, HasSubevent, feel hurt), (feel hurt, SIM, feel sad)
(chat with friend, HasSubevent, argue)	(chat with friend, SIM, talk to friend), (talk to friend, SIM, talk to someone), (talk to someone, HasSubevent, argue)
(relax, HasSubevent, stop breathe)	(relax, HasPrerequisite, listen to music), (listen to music, HasSubevent, die), (die, Causes, stop breathe)
Error Type 3	
(read newspaper, HasPrerequisite, buy food)	(read newspaper, HasSubevent, eat breakfast), (eat breakfast, HasPrerequisite, buy food)
(door, AtLocation, street)	(door, PartOf, car), (car, AtLocation, street)
(seat, AtLocation, repair shop)	[seat, PartOf, bicycle), (bicycle, AtLocation, repair shop)

Table 7: Examples of three error types

False Negatives	
(do housework, Causes, muscle growth)	(UsedFor, do housework, exercise), (Causes, exercise, muscle growth]
(boat, UsedFor, mobility]	(boat, IsA, vehicle), (vehicle, UsedFor, mobility)
(clock, CapableOf, indicate passage of time)	(clock, UsedFor, indicate passage of time)
(leaf, AtLocation, garden)	(leaf, PartOf, plant), (plant, AtLocation, garden)
(chat with friend, HasSubevent, breath)	(chat with friend, SIM, talk to friend), (talk to friend, SIM, talk to someone), (talk to someone, HasSubevent, breath)

Table 8: Examples of false negatives

7 Limitations

The current model requires a pre-computed chain between a head entity and a tail entity which has no guarantee that it helps validate the target triple. Although empirical experiments demonstrated that most of the chains are helpful to determine the validity of the target triple, noise could be introduced. It is also difficult to improve the accuracy of the chains through which we may further improve the performance of the neural logic reasoning model. In future work, we aim to build trainable models that are able to learn to find or generate chains to help validate the target triple, aiming at both comprehensiveness and accuracy.

In the current version of the model, each training datapoint has k negatives with different length of chains, and thus it is difficult to batch the training data. For example, expressions $p_1 \wedge p_2 \rightarrow q_1$ and $p_3 \wedge p_4 \wedge p_5 \rightarrow q_2$ cannot be put into the same batch since the different length leads to different neural logic network structures. One method we will try in the future to solve the problem is to append the constant true embedding $\mathbf{T}$ to the proposition chain so as to align the length. For example, expression $p_1 \wedge p_2 \rightarrow q_1$ can be rewritten as $\mathbf{T} \wedge p_1 \wedge p_2 \rightarrow q_1$ and thus it can be batched with $p_3 \wedge p_4 \wedge p_5 \rightarrow q_2$ since they are same-length expressions which share the same System 2 network structure.

Appendix

1.1 Implementation Details

The embedding size for expression embedding is 1024 unless otherwise specified. The embedding size is 100 for NTN on WebChild-Comparative since this embedding size gives NTN its best performance. The number of layers of the logical modules is 5. The weights for the logical regularizer and value regularizer are both 0.5, i.e., $\lambda_l = \lambda_v = 0.5$. For the number of negative samples k, we use $k = 10$ to train the model on ConceptNet-100k and $k = 100$ to train model on WebChild-comparative because WebChild-comparative is a much larger dataset. The α amplifying parameter is set to be 10. The optimization function is AdamW with learning rate 1e-5. The scheduler is linear warm-up with 500 steps. The batch size is 1 with 5 accumulate gradient descent steps. Each baseline model is trained for at most 200 epochs and each neural logic model is trained for 10 epochs where the System 1 module is initialized by a trained checkpoint.

1.2 Case Study

Some examples of the three error types and false negative cases on ConceptNet-100k are presented in Table 7 and Table 8, respectively.

Acknowledgement. This work was supported in part by NSF IIS 1910154, 2007907, and 2046457.

References

Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019. Hypernetwork knowledge graph embeddings. In *International Conference on Artificial Neural Networks*, pages 553–565. Springer.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. Comet: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *Twenty-Fourth AAAI conference on artificial intelligence*.

Hanxiong Chen, Yunqi Li, Shaoyun Shi, Shuchang Liu, He Zhu, and Yongfeng Zhang. 2022a. Graph collaborative reasoning. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 75–84.

Hanxiong Chen, Yunqi Li, He Zhu, and Yongfeng Zhang. 2022b. Learn basic skills and reuse: Modularized adaptive neural architecture search (manas). In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*.

Hanxiong Chen, Shaoyun Shi, Yunqi Li, and Yongfeng Zhang. 2021. Neural collaborative reasoning. In *Proceedings of the Web Conference 2021*, pages 1516–1527.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Yujia Fan and Yongfeng Zhang. 2022. Neural logic analogy learning. *In Proceedings of the ICLR 2022 PAIR2Struct Workshop*.

Artur d'Avila Garcez, Sebastian Bader, Howard Bowman, Luis C Lamb, Leo de Penning, BV Illuminoo, Hoifung Poon, and COPPE Gerson Zaverucha. 2022. Neural-symbolic learning and reasoning: A survey and interpretation. *Neuro-Symbolic Artificial Intelligence: The State of the Art*, 342:1.

Shijie Geng, Zuohui Fu, Juntao Tan, Yingqiang Ge, Gerard De Melo, and Yongfeng Zhang. 2022. Path language modeling over knowledge graphsfor explainable recommendation. In *Proceedings of the ACM Web Conference 2022*, pages 946–955.

Thomas Gilovich, Dale Griffin, and Daniel Kahneman. 2002. *Heuristics and biases: The psychology of intuitive judgment*. Cambridge university press.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Jena D. Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. 2021. Comet-atomic 2020: On symbolic and neural commonsense knowledge graphs. In *AAAI*.

Jaehun Jung, Bokyung Son, and Sungwon Lyu. 2020. Attnio: Knowledge graph exploration with in-and-out attention flow for knowledge-grounded dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3484–3497.

Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. *Advances in neural information processing systems*, 31.

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

Yann LeCun, Yoshua Bengio, et al. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

Yong-Lu Li, Xinpeng Liu, Xiaoqian Wu, Yizhuo Li, Zuoyu Qiu, Liang Xu, Yue Xu, Hao-Shu Fang, and Cewu Lu. 2022. Hake: A knowledge engine foundation for human activity understanding. *arXiv preprint arXiv:2202.06851*.

Zelong Li, Jianchao Ji, Zuohui Fu, Yingqiang Ge, Shuyuan Xu, Chong Chen, and Yongfeng Zhang. 2021. Efficient non-sampling knowledge graph embedding. In *Proceedings of the Web Conference 2021*, pages 1727–1736.

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. *arXiv preprint arXiv:1907.11692*.

Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. 2020. Commonsense knowledge base completion with structural and semantic context. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 2925–2933.

Bhavana Dalvi Mishra, Niket Tandon, and Peter Clark. 2017. Domain-targeted, high precision knowledge extraction. *Transactions of the Association for Computational Linguistics*, 5:233–246.

Farhad Moghimifar, Lizhen Qu, Terry Yue Zhuo, Gholamreza Haffari, and Mahsa Baktashmotlagh. 2021. Neural-symbolic commonsense reasoner with relation predictors. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 797–802.

Tuan-Phong Nguyen, Simon Razniewski, and Gerhard Weikum. 2021. Advanced semantics for commonsense knowledge extraction. In *Proceedings of the Web Conference 2021*, pages 2636–2647.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763.

Julien Romero and Simon Razniewski. 2020. Inside quasimodo: Exploring construction and usage of commonsense knowledge. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 3445–3448.

Julien Romero, Simon Razniewski, Koninika Pal, Jeff Z. Pan, Archit Sakhadeo, and Gerhard Weikum. 2019. Commonsense properties from query logs and question answering forums. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1411–1420.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. 2019. Atomic: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3027–3035.

Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. 2019. End-to-end structure-aware convolutional networks for knowledge base completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3060–3067.

Shaoyun Shi, Hanxiong Chen, Weizhi Ma, Jiaxin Mao, Min Zhang, and Yongfeng Zhang. 2020. Neural logic reasoning. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1365–1374.

Steven A Sloman. 1996. The empirical case for two systems of reasoning. *Psychological bulletin*, 119(1):3.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. *Advances in neural information processing systems*, 26.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence*.

Niket Tandon, Gerard De Melo, Fabian Suchanek, and Gerhard Weikum. 2014. Webchild: Harvesting and organizing commonsense knowledge from the web. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 523–532.

Niket Tandon, Gerard De Melo, and Gerhard Weikum. 2017. Webchild 2.0: Fine-grained commonsense knowledge distillation. In *Proceedings of ACL 2017, System Demonstrations*, pages 115–120.

Théo Trouillon, Christopher R Dance, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2017. Knowledge graph completion via complex tensor factorization. *arXiv preprint arXiv:1702.06879*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Yikun Xian, Zuohui Fu, Shan Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, pages 285–294.

Yikun Xian, Zuohui Fu, Handong Zhao, Yingqiang Ge, Xu Chen, Qiaoying Huang, Shijie Geng, Zhou Qin, Gerard De Melo, Shan Muthukrishnan, et al. 2020. Cafe: Coarse-to-fine neural symbolic reasoning for explainable recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1645–1654.

Xiaoran Xu, Wei Feng, Yunsheng Jiang, Xiaohui Xie, Zhiqing Sun, and Zhi-Hong Deng. 2019. Dynamically pruned message passing networks for large-scale knowledge graph reasoning. In *International Conference on Learning Representations*.

Jun Yan, Mrigank Raman, Aaron Chan, Tianyu Zhang, Ryan Rossi, Handong Zhao, Sungchul Kim, Nedim

Lipka, and Xiang Ren. 2021. Learning contextualized knowledge structures for commonsense reasoning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4038–4051.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Fan Yang, Zhilin Yang, and William W Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning. *Advances in neural information processing systems*, 30.

Jing Zhang, Bo Chen, Lingxi Zhang, Xirui Ke, and Haipeng Ding. 2021. Neural, symbolic and neural-symbolic reasoning on knowledge graphs. *AI Open*, 2:14–35.