IMPLICIT REASONET: IMPLICITLY MODELING LARGE-SCALE STRUCTURAL RELATIONSHIPS WITH SHARED MEMORY

Yelong Shen*, Po-Sen Huang*, Ming-Wei Chang , Jianfeng Gao
Microsoft Research, Redmond, WA, USA
{yeshen,pshuang,minchang,jfgao}@microsoft.com

ABSTRACT

The knowledge base completion task (KBC) has emerged as an important open research problem. Existing neural models for KBC typically use linear prediction functions. However, a much more involved inference or search procedure is needed to make a correct prediction. Unfortunately, due to the size of knowledge bases, performing search with neural models on top of all observed triples can be very costly. In this paper, we propose performing large-scale inference through the design of *shared memory*. Our inference procedure is controlled by the *search controller*, which only operates on the shared memory instead of on the set of observed triples. We propose Implicit ReasoNet (IRN) that uses a shared memory and an attention mechanism to model large scale relationships implicitly. Without using any explicit knowledge base information, our proposed model outperforms all of previous approaches significantly on the popular FBK15K and WN18 benchmarks.

1 INTRODUCTION

Knowledge bases such as WordNet (Fellbaum, 1998), Freebase (Bollacker et al., 2008), or Yago (Suchanek et al., 2007) contain many real-world facts expressed as triples, e.g. (*Bill Gates*, FounderOf, *Microsoft*). These knowledge bases are useful for many downstream applications such as question answering (Berant et al., 2013; Yih et al., 2015) and information extraction (Mintz et al., 2009). However, despite their formidable size, many important facts are still missing in the knowledge bases. For example, West et al. (2014) showed that 21% of the 100K most frequent PERSON entities have no known nationality in a recent version of Freebase. Thus, the link prediction or knowledge base completion task (KBC) has emerged as an important open research problem. Formally, we seek to infer unknown relations based on the observed triples.

Neural-network based methods have been very popular for solving the KBC task. Following (Bordes et al., 2013), one of the most popular approaches for the KBC task is to learn a vector-space representation of entities and relations during training, and then apply simple operations to infer the missing relations at test time. However, several recent papers demonstrate limitations of approaches relying upon vector-space models alone. By themselves, these models cannot capture the structural relationships between multiple triples adequately (Guu et al., 2015; Toutanova et al., 2016; Lin et al., 2015a). These papers each propose new ways to inject structural information at training time.

Existing neural KBC models, despite their strong empirical performance, often use very simple prediction functions. Typically, the probability of two entities that have a previously unknown relationship is proportional to a linear or bi-linear function of their corresponding vector or matrix representations. However, intuitively, a much more involved inference or search procedure is needed to make the correct prediction. For example, assume that we want to fill in the missing relation for the triple (Obama, NATIONALITY, ?), a multi-step search procedure can make use of the evidence in the observed triples such as (Obama, BORNIN, Hawaii) and (Hawaii, PARTOF, U.S.A). Unfortunately, due to the size of knowledge bases, performing search with neural models on top of all observed triples can be very costly, or heavy constraints needed to be applied to inference procedures and models.

*These authors contributed equally to this work.

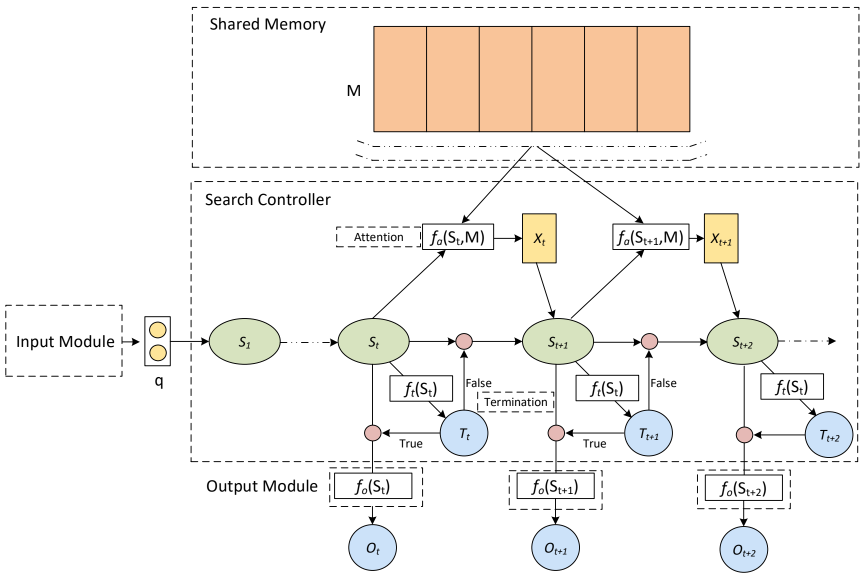


Figure 1: An IRN Architecture.

In this paper, we take a different approach from prior work for the KBC task by addressing the challenges of performing large-scale inference through the design of *search controller* and *shared memories*. Our inference procedure is controlled by the *search controller*, which only operates on the shared memories instead of on the set of observed triples. After receiving the representation from the *input module*, the search controller will interact with the *shared memory* and call the *output module* several times until it feels confident enough to output the prediction. Intuitively, the shared memory is designed to store key information about the overall structures it learned during training, and hence the search controller only needs to access the shared memory instead of the observed triples.

There are several advantages of using IRNs. First, the cost of inference can be controlled because the search controller only needs to access the shared memory. Second, all the modules are jointly trained, and hence the model does not require extra inference procedures designed by human. Finally, it is easy to apply IRNs to other tasks by switching the input and output modules.

The contributions of our paper are as follows:

- We propose Implicit ReasoNet (IRN), which uses a shared memory and an attention mechanism to model large scale relationships implicitly.
- IRNs surpass all of prior approaches significantly on the popular FBK15K and WN18 benchmarks.
- We analyze the behavior of IRNs with a newly proposed generation task, shortest path synthesis. We show that IRNs outperform a standard sequence-to-sequence model, and are capable of performing meaningful multi-step inference.

2 REASONET FOR IMPLICIT INFERENCE

In this section, we first describe the general architecture of IRNs, and our description is agnostic to the KBC task. There are four main components in an IRN, including an input component, an output component, a shared memory, and a search controller, as shown in Figure 1. The descriptions of these modules are as follow:

Input/Output Module: These two modules are task-dependent ones. The input module takes a query and converts the query into vector representations q. The output module is a function f_o which converts the hidden state receives from the search controller (s) into the output O. We optimize the whole model using the output prediction O with respect to a ground-truth target using a task-specified loss function.

Shared Memory: The shared memory is denoted as M. It consists of a list of memory vectors, $M = \{m_i\}_{i=1...D}$, where m_i is a fixed dimensional vector. The memory vectors are randomly initialized and automatically updated through back-propagation. The shared memory component is shared across all of the instances.

Search Controller: The search controller is a recurrent neural work and controls the search process by keeping an internal state sequences to track the current search process and history, using an attention mechanism to fetch information from relevant memory vectors in M, and deciding if the model should output the prediction or continue to generate the next possible output.

- *Internal State*: The internal state is denoted as S, which is a vector representation of the input state. The initial state s_1 is usually the vector representation of the input vector q. The t-th time step of the internal state is represented by s_t. The sequence of internal states is modeled by an RNN: $s_{t+1} = \text{RNN}(s_t, x_t; \theta_s)$.
- *Attention to memory*: The attention vector x_t at t-th time step is generated based on the current internal state s_t and the shared memory M: $x_t = f_{att}(s_t, M; \theta_x)$. Specifically, the attention score $a_{t,i}$ on a memory vector m_i given a state s_t is computed as $a_{t,i} = \text{softmax}_{i=1,...,|M|}\lambda \cos(W_1 m_i, W_2 s_t)$, where λ is set to 10 in our experiments and the weight matrices W_1 and W_2 are learned in the training process. The attention vector x_t can be written as $x_t = \sum_i^{|M|} a_{t,i} m_i$.
- *Termination Control*: The terminate gate produces a stochastic random variable according to the current internal state, $t_t \sim p(\cdot|f_t(s_t; \theta_t)))$. t_t is a binary random variable. If t_t is true, the `IRN` will finish the search process, and the output module will execute at time step t; otherwise the `IRN` will generate the next attention vector x_{t+1}, and feed into the state network to update the next internal state s_{t+1}. In our experiments, the termination variable is modeled by a logistical regression approach: $f_t(s_t; \theta_t) = \text{sigmoid}(W_t s_t + b_t)$, where the weight matrix W_t and bias vector b_t are learned during training.

Compared `IRNs` to Memory Network Weston et al. (2014), the biggest difference between our model and Memory Network is the use of the search controller. The search controller allows us to perform a multi-step inference even for just making a single prediction. The differences between our model and the existing frameworks such as (Graves et al., 2014; 2016; Shen et al., 2016) is on the use of shared memory, where the key information of the *whole* training data will be stored in the shared memory.

2.1 STOCHASTIC INFERENCE PROCESS

The inference process of an `IRN` is as follows. First, the model converts a task- dependent input to a vector representation through the input module. Then, the model uses the input representation to initialize the search controller. In every time step, the search controller determines whether the process is finished by checking its terminate gate, which generate a binary random variable bases on the search controller states. If the termination gate determines to continue, the search controller will move on to the next time step, and create an attention vector based on the current search controller state and a shared memory. Finally, once the terminate gate decides to finish the current process, the output module will generate a task-dependent prediction given the search controller states. The whole process can be thought of as the model iteratively searches for its target through a shared memory and output its prediction when a satisfied answer is found. The detailed inference process is described in Algorithm 1.

The process of `IRN` is considered as a Partially Observable Markov Decision Process (POMDP) Kaelbling et al. (1998) in the reinforcement learning (RL) literature. `IRN` produces the output vector o_T at the T-th step, which implies termination gate variables $t_{1:T} = (t_1 = 0, t_2 = 0, ..., t_{T-1} = 0, t_T = 1)$, and then takes prediction action p_T according to the probability distribution given o_T. Therefore, `IRN` learns a stochastic policy $\pi((t_{1:T}, p_T)|q; \theta)$ with parameters θ to get a distribution over termination actions, and over prediction actions. The termination step T varies from instance to instance.

Algorithm 1: Stochastic Inference Process in an IRN

Input : Shared memory M; Input vector q; Maximum step T_{max}
Output : Output vector o

1 Define $s_1 = q$; $t = 1$;
2 Generate a binary termination variable $t_t \sim p(\cdot|f_t(s_t;\theta_t))$;
3 if t_t is false, go to Step 4; otherwise Step 7;
4 Generate an attention vector $x_t = f_{att}(s_t, M; \theta_x)$;
5 Update the internal state $s_{t+1} = \text{RNN}(s_t, x_t; \theta_s)$;
6 Set $t = t + 1$; if $t < T_{\text{max}}$ go to Step 2; otherwise Step 7;
7 Generate output $o_t = f_o(s_t; \theta_o)$;
8 Return $o = o_t$;

The parameters of the IRN θ are given by the parameters of shared memory M, the attention network θ_x, the search controller RNN network θ_s, the output generation network θ_o, and the termination gate network θ_t. The parameters $\theta = \{M, \theta_x, \theta_s, \theta_o, \theta_t\}$ are trained by maximizing the total reward that the IRN could expect when interacting with the environment. The expected reward for an instance is defined as:

$$J(\theta) = \mathbb{E}_{\pi(t_{1:T}, p_T; \theta)} \left[\sum_{t=1}^{T} r_t \right]$$

The reward can only be received at the final termination step when a prediction action p_T is performed. The rewards on intermediate steps are zeros, $\{r_t = 0\}_{t=1...T-1}$.

We employ the approach from our previous work (Shen et al., 2016), REINFORCE Williams (1992) based Contrastive Reward method, to maximize the expected reward. The gradient of J can be written as:

$$\nabla_\theta J(\theta) = \sum_{(t_{1:T}, a_T) \in \mathbb{A}^\dagger} \pi(t_{1:T}, p_T; \theta) \left[\nabla_\theta \log \pi(t_{1:T}, p_T; \theta) (\frac{r_T}{b^i} - 1) \right]$$

where $\mathbb{A}^\dagger$ is all the possible episodes, the baseline $b^i = \sum_{(t_{1:T}, a_T) \in \mathbb{A}^\dagger} \pi(t_{1:T}, a_T; \theta) r_T$ is the expected reward on the $|\mathbb{A}^\dagger|$ episodes for the i-th training instance.

3 Applying IRNs to Knowledge Base Completion

The goal of KBC tasks Bordes et al. (2013) is to predict a head or a tail entity given the relation type and the other entity, i.e. predicting h given $(?, r, t)$ or predicting t given $(h, r, ?)$ where $?$ denotes the missing entity.

For a KBC task, the input to our model is a head or tail entity and a relation. The task-dependent input module first extracts the embedding vectors for the entity and relation from an embedding matrix. We then represent the query vector q for the IRN as the concatenation of the two vectors. For the task dependent output module, we use a nonlinear projection to project the search controller state into an output vector o: $f_o(s_t; \theta_o) = \tanh(W_o s_t + b_o)$, where the W_o and b_o are the weight matrix and bias vector, respectively. Define the ground truth target entity embedding as y, and a L_1 distance measure between o and y, namely $d(o, y) = |o - y|_1$. We sample a set of incorrect entity embeddings $N = \{y_i^-\}_{i=1}^{|N|}$ as negative examples and the probability of selecting the ground-truth entity can be approximated as

$$p(y|o) = \frac{\exp(-\gamma d(o, y))}{\sum_{y_k \in D} \exp(-\gamma d(o, y_k))}$$

where $D = N \cup \{y\}$. We set $|N|$ and γ to 20 and 5, respectively, in the experiments of the IRN on FB15K and WN18 data sets. The IRN performs a prediction action p_T on selecting one entity from

D with probability $p(\hat{y}|o)$ where $\hat{y} \in D$. Hence, we define the reward on the prediction action as $r = 1$ if the ground truth entity is selected, and $r = 0$ otherwise.

4 EXPERIMENTAL RESULTS

In this section, we evaluate the performance of our model on the benchmark FB15k and WN18 datasets for KBC tasks Bordes et al. (2013). These datasets contain multi-relations between head and tail entities. Given a head entity and a relation, the model produces a ranked list of the entities according to the score of the entity being the tail entity of this triple. To evaluate the ranking, we report **mean rank (MR)**, the mean of rank of the correct entity across the test examples, and **hits@10**, the proportion of correct entities ranked in the top-10 predictions. Lower MR or higher hits@10 indicates a better prediction performance.

We use the same hyper-parameters of our model for both FB15k and WN18 data sets. Entity embeddings (which are not shared between input and output modules) and relation embedding are both 100-dimensions. There are 64 memory vectors with 200 dimensions each, initialized by random vectors with unit L_2-norm. We use single-layer GRU with 200 cells as the search controller. We set the maximum number of reasoning steps of the `IRN` to 5. We randomly initialize all model parameters, and use SGD as the training algorithm with mini-batch size of 64. We set the learning rate to a constant number, 0.01. To prevent the model from learning a trivial solution by increasing entity embeddings norms, we follow Bordes et al. (2013) to enforce the L_2-norm of the entity embeddings as 1. We use hits@10 as the validation metric for the `IRN`. Following the work (Lin et al., 2015a), we add reverse relations into the training triplet set, i.e., for each triple (h, r, t), we build two training instances, $(h, r) \rightarrow t$ and $(t, r^{-1}) \rightarrow h$.

Following (Nguyen et al., 2016), we divide the results of previous work into two groups. The first group contains models which directly optimizes a scoring function for the triples in a knowledge base without using extra information. The second group of models make uses of additional structural or textual information. For example, RTransE (García-Durán et al., 2015) and PTransE (Lin et al., 2015a) models are extensions of the TransE (Bordes et al., 2013) model by explicitly exploring path information in the knowledge base to regularize the trained embeddings. The NLFeat model (Toutanova et al., 2015) is a simple log-linear model that makes use of textual mentions from ClueWeb-12 corpus.

The results are in Table 1. According to the table, our model significantly outperforms previous baselines, despite if the usage of additional information or not. Specifically, on FB15k, the MR of our model surpass all previous results by 20, and our hit@10 leads others by 5%. When comparing to WN18, the `IRN` obtains the highest hit@10 while maintaining similar MR results compared to previous work.[1]

We analyze hits@10 results on FB15k with respect to the relation categories. We evaluate the performance in four types of relation categories: `1-1`, `Many-1`, `Many-1`, and `Many-Many`, where the former and the latter indicate the number of average head and tail entities given the corresponding relation, tail/head pair, respectively. `Many` is defined as the average number of entities is greater than 1.5. In Bordes et al. (2013), it gives more details about relation category analysis. The detailed results in Table 2 allow for a precise evaluation and understanding of the behavior of different approaches. The `IRN` significantly improves the $hits$@10 results in `Many-1` category on predicting head entity (18%), `1-Many` category on predicting tail (16%), and `Many-Many` category (over 10% in average).

To analyze the behavior of `IRNs`, we pick some examples for the tail entity prediction in Table 3. Interestingly, we observed that the model could gradually push the correct tail entity with a higher rank score during the search process.

[1]Nguyen et al. (2016) reported two results on WN18, where the first one is obtained by choosing to optimize hits@10 on the validation set, and second one is obtained by choosing to optimize MR on the validation set. We list both of them in Table 1.

Table 1: The knowledge base completion (link prediction) results on WN18 and FB15K.

Model	Additional Information	WN18		FBK15K	
		MR	Hits@10 (%)	MR	Hits@10 (%)
SE (Bordes et al., 2011)	NO	985	80.5	162	39.8
Unstructured (Bordes et al., 2014)	NO	304	38.2	979	6.3
TransE (Bordes et al., 2013)	NO	251	89.2	125	47.1
TransH (Wang et al., 2014)	NO	303	86.7	87	64.4
TransR (Lin et al., 2015b)	NO	225	92.0	77	68.7
CTransR (Lin et al., 2015b)	NO	218	92.3	75	70.2
KG2E (He et al., 2015)	NO	348	93.2	59	74.0
TransD (Ji et al., 2015)	NO	212	92.2	91	77.3
TATEC (García-Durán et al., 2015)	NO	-	-	58	76.7
NTN (Socher et al., 2013)	NO	-	66.1	-	41.4
DISTMULT (Yang et al., 2014)	NO	-	94.2	-	57.7
STransE (Nguyen et al., 2016)	NO	244 (**206**)	94.7 (93)	69	79.7
RTransE (García-Durán et al., 2015)	Path	-	-	50	76.2
PTransE (Lin et al., 2015a)	Path	-	-	58	84.6
NLFeat (Toutanova et al., 2015)	Textual Mention	-	94.3	-	87.0
IRN		249	**95.3**	**38**	**92.7**

Table 2: Hits@10 (%) in the relation category on FB15k. (M stands for Many)

Model	Predicting head h				Predicting tail t			
	1-1	1-M	M-1	M-M	1-1	1-M	M-1	M-M
SE (Bordes et al., 2011)	35.6	62.6	17.2	37.5	34.9	14.6	68.3	41.3
Unstructured (Bordes et al., 2014)	34.5	2.5	6.1	6.6	34.3	4.2	1.9	6.6
TransE (Bordes et al., 2013)	43.7	65.7	18.2	47.2	43.7	19.7	66.7	50.0
TransH (Wang et al., 2014)	66.8	87.6	28.7	64.5	65.5	39.8	83.3	67.2
TransR (Lin et al., 2015b)	78.8	89.2	34.1	69.2	79.2	37.4	90.4	72.1
CTransR (Lin et al., 2015b)	81.5	89.0	34.7	71.2	80.8	38.6	90.1	73.8
KG2E (He et al., 2015)	**92.3**	94.6	66.0	69.6	**92.6**	67.9	94.4	73.4
TransD (Ji et al., 2015)	86.1	95.5	39.8	78.5	85.4	50.6	94.4	81.2
TATEC (García-Durán et al., 2015)	79.3	93.2	42.3	77.2	78.5	51.5	92.7	80.7
STransE (Nguyen et al., 2016)	82.8	94.2	50.4	80.1	82.4	56.9	93.4	83.1
PTransE (Lin et al., 2015a)	91.0	92.8	60.9	83.8	91.2	74.0	88.9	86.4
IRN	87.2	**96.1**	**84.8**	**92.9**	86.9	**90.5**	**95.3**	**94.1**

5 Analysis: Applying IRN to a Shortest Path Synthesis Task

We construct a synthetic task, shortest path synthesis, to evaluate the inference capability over a shared memory. The motivations of applying our model to this task are as follows. First, we want to test if IRNs can work on another task which also requires performing inference to get the correct answer. Second, we choose to make this task to be a *sequence generation* task so that we are able to analyze the capability of IRNs for performing inference in greater details.

In the shortest path synthesis task, a training example consists of a start node and an end node ($76 \rightsquigarrow 493$) of an underlying weighted directed graph that is unknown to models. The output of each example is one of the shortest paths between the given start and the end nodes of the underlying graph ($76 \to 308 \to 101 \to 493$). At test time, a path sequence is considered correct if it connects the start node and the end node of the underlying graph, and the cost of the predicted path is the same as the optimal path.

Note that the task is very difficult and *cannot* be solved by dynamic programming algorithms since the weights on the edges are not revealed to the algorithms or the models. To recover some of the shortest paths at the test time, the model needs to learn to compose the correct path from the observed examples. In order to answer the shortest path between A and E, and assume that we have observe two examples in the training data, "$A \rightsquigarrow D$: $A \to B \to G \to D$" and "$B \rightsquigarrow E$: $B \to C \to E$". The model can observe that "$A \to B \to C \to E$" is a possible path between A and E. If there are

Table 3: Test examples in FB15k dataset, given a head entity and a relation, the `IRN` predicts the tail entity with multiple search steps.

Input: (`Dean Koontz`, /PEOPLE/PERSON/PROFESSION)
Target: `Film Producer`

Step	Termination Prob.	Rank	Predict top-3 entities		
1	0.018	9	`Author`	`TV. Director`	`Songwriter`
2	0.052	7	`Actor`	`Singer`	`Songwriter`
3	0.095	4	`Actor`	`Singer`	`Songwriter`
4	0.132	4	`Actor`	`Singer`	`Songwriter`
5	0.702	3	`Actor`	`Singer`	**`Film Producer`**

Input: (`1964 Summer Olympics`, /OLYMPICS/OLYMPIC_GAMES/SPORTS)
Target: `Judo`

Step	Termination Prob.	Rank	Predict top-3 entities		
1	0.018	498	`Rugby union`	`Golf`	`Curling`
2	0.079	4	`Alpine skiing`	`Tennis`	`Archery`
3	0.164	3	`Alpine skiing`	`Tennis`	**`Judo`**
4	0.215	1	**`Judo`**	`Alpine skiing`	`Archery`
5	0.524	1	**`Judo`**	`Alpine skiing`	`Archery`

multiple possible paths, the model has to decide which one is the shortest one using the statistical information.

In the experiments, we construct a graph with 500 nodes, and for each examples we randomly assign two paths to form a path. We split 20,000 examples for training, 10,000 examples for validation, and 10,000 examples for testing. We create the training and testing instances very carefully so that the model needs to perform inference to recover the correct path. The details of the graph and the data construction is in the appendix. A sub-graph of the data is shown in Figure 2.

For the settings of the `IRN`, we switch the output module by a GRU decoder to perform as a sequence generation task. We assign reward $r_T = 1$ if all the prediction symbols are correct and 0 otherwise. We use a 64-dimensional embedding vector for input symbols, a GRU controller with 128 cells, and a GRU decoder with 128 cells. The maximum reasoning step T_{max} is set to 5.

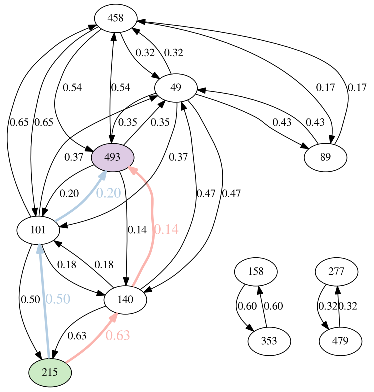
We compare the `IRN` with two baseline approaches: dynamic programming without edge weights information and a standard sequence-to-sequence model Sutskever et al. (2014). Without knowing the weights from the data, dynamic programming cannot infer the correct path and recovers only 589 correct paths during test time. The sequence-to-sequence model with similar size of the parameters recovers 904 correct paths. The `IRN` beats both baselines and recovered 1,319 paths. The `IRN` has the 76.9% of producing a *valid* path, which represents that the predicted path connects the start and end node nodes of the underlying graph, compared to 69.1% produced by the sequence to sequence model.

To further understand the inference process in the `IRN`, Figure 2 shows the inference process of a test example of the shortest path synthesis dataset. Interestingly, to make the correct prediction on this example, the model has to perform a fairly complicated inference.[2] We observe that the model does not make the correct prediction at the first step. In fact, for the first three steps that the model is not even able to find a connected path. Finally, it corrected the sequence it found at the forth step and produced the correct shortest path sequence at the fifth step.

6 Related Work

Knowledge base completion In the knowledge base completion (KBC) task, recent work suggest that training vector space models with explicit path information can improve the performance of

[2] In the example, to find the right path, the model needs to search over observed instances "$215 \rightsquigarrow 448$: $215 \to 101 \to 448$" and "$76 \rightsquigarrow 493$: $76 \to 308 \to 101 \to 493$", and figure out the distance of "$140 \to 493$" is longer than "$101 \to 493$" (there are four shortest paths between $101 \to 493$ and three shortest paths between $140 \to 493$).



Step	Termination Probability	Distance	Predictions
1	0.001	N/A	$215 \rightarrow 158 \rightarrow 89 \rightarrow 458 \rightarrow 493$
2	$\sim$0	N/A	$215 \rightarrow 479 \rightarrow 277 \rightarrow 353 \rightarrow 493$
3	$\sim$0	N/A	$215 \rightarrow 49 \rightarrow 493$
4	$\sim$0	0.77	$215 \rightarrow 140 \rightarrow 493$
5	0.999	0.70	$215 \rightarrow 101 \rightarrow 493$

Figure 2: An example of the shortest path synthesis dataset, given an input "$215 \rightsquigarrow 493$" (Answer: $215 \rightarrow 101 \rightarrow 493$). The corresponding termination probability and prediction results are shown in the table. The model terminates at step 5.

the KBC task significantly (García-Durán et al., 2015; Lin et al., 2015a; Toutanova et al., 2015) Nevertheless, these models have some limitations, as most paths are not informative for inferring missing relations, and it is prohibitive to consider all possible paths during the training time with expressive models. Prior approaches address this issue by designing a path-mining algorithm (Lin et al., 2015a) or considering all possible paths using a linear model (Toutanova et al., 2015).Instead of asking the question: "what extra information can we incorporate to improve the performance for the KBC task?", we ask the following research question: "can we design a model that can capture structural relationships between instances automatically?" The proposed `IRNs` learn to perform search over a shared memory without using explicitly structures (path information) and outperform prior approaches regardless of the use of explicit path information.

Sequence-to-sequence models Sequence-to-sequence models (Sutskever et al., 2014; Cho et al., 2014) have shown a great success in many applications such as machine translation. The models learn the mapping from an input sequence to an output sequence within an instance. In scenarios like shortest path synthesis tasks where an input sequence is simply a start-end node pair, sequence-to-sequence models can only use the decoder to memorize previously seen information. To solve the problem, it requires performing search over the underlying structures from observed instances. Our proposed `IRNs` iteratively search through a shared memory module that implicitly stores the structural information across instances. `IRNs` have shown its ability to outperform sequence to sequence models in the shortest path synthesis task by performing iterative search over a shared memory.

ReasoNet We are inspired by our previous work (Shen et al., 2016) for using a search controller module to perform iteratively search over a given paragraph and stop until a satisfied answer is found. In contrast, in this work, we propose `IRNs`, which use a shared memory component along with the search controller to address the issue of searching over large-scale structural relationships.

7 Conclusion

In this paper, we propose Implicit ReasoNet (`IRN`), which uses a shared memory and an attention mechanism to model large scale relationships implicitly. We demonstrate and analyze the multi-step inference capability of `IRN` in the knowledge base completion tasks and a shortest path synthesis task. Our model, without using any explicit knowledge base information, outperforms all of prior approaches significantly on the popular FBK15K and WN18 benchmarks. For future work, we aim to further develop `IRN` for modeling the relationships from unstructured data such as in natural language processing.

Acknowledgments

We thank Scott Wen-Tau Yih and Zachary Lipton for their thoughtful feedback and discussions.

REFERENCES

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of EMNLP*, 2013.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of SIGMOD-08*, pp. 1247–1250, 2008.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pp. 301–306, 2011.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pp. 2787–2795, 2013.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259, 2014.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

Alberto García-Durán, Antoine Bordes, and Nicolas Usunier. Composing relationships with translations. In *EMNLP*, pp. 286–290, 2015.

Alberto García-Durán, Antoine Bordes, Nicolas Usunier, and Yves Grandvalet. Combining two and three-way embeddings models for link prediction in knowledge bases. *CoRR*, abs/1506.00999, 2015.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 2016.

Kelvin Guu, John Miller, and Percy Liang. Traversing knowledge graphs in vector space. *arXiv preprint arXiv:1506.01094*, 2015.

Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. Learning to represent knowledge graphs with gaussian embedding. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pp. 623–632, 2015.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. In *ACL*, pp. 687–696, 2015.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2015a.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pp. 2181–2187, 2015b.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL-IJCNLP-09*, pp. 1003–1011, 2009.

Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. STransE: a novel embedding model of entities and relationships in knowledge bases. In *NAACL*, pp. 460–466, 2016.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. Reasonet: Learning to stop reading in machine comprehension. *CoRR*, abs/1609.05284, 2016.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. Reasoning With Neural Tensor Networks For Knowledge Base Completion. In *Advances in Neural Information Processing Systems*, 2013.

F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A Core of Semantic Knowledge. In *WWW*, 2007.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pp. 3104–3112, 2014.

Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pp. 57–66, 2015.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In *EMNLP*, 2015.

Kristina Toutanova, Xi Victoria Lin, Scott Wen tau Yih, Hoifung Poon, and Chris Quirk. Compositional learning of embeddings for relation paths in knowledge bases and text. In *ACL*, 2016.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pp. 1112–1119, 2014.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd international conference on World Wide Web*, pp. 515–526. ACM, 2014.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

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

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proc. of ACL*, 2015.

A Details of the graph construction for the shortest path synthesis task

We construct the underlying graph as follows: on a three-dimensional unit-sphere, we randomly generate a set of nodes. For each node, we connect its K-nearest neighbors and use the euclidean distance between two nodes to construct a graph. We randomly sample two nodes and compute its shortest path if it is connected between these two nodes. Given the fact that all the sub-paths within a shortest path are shortest paths, we incrementally create the dataset and remove the instances which are a sub-path of previously selected paths or are super-set of previous selected paths. In this case, all the shortest paths can not be answered through directly copying from another instance. In addition, all the weights in the graph are hidden and not shown in the training data, which increases the difficulty of the tasks. We set $k = 50$ as a default value.