SCALABLE NEURAL METHODS FOR REASONING WITH A SYMBOLIC KNOWLEDGE BASE

Anonymous authors
Paper under double-blind review

ABSTRACT

We describe a novel way of representing a symbolic knowledge base (KB) called a *sparse-matrix reified KB*. This representation enables neural modules that are fully differentiable, faithful to the original semantics of the KB, expressive enough to model multi-hop inferences, and scalable enough to use with realistically large KBs. The sparse-matrix reified KB can be distributed across multiple GPUs, can scale to tens of millions of entities and facts, and is orders of magnitude faster than naive sparse-matrix implementations. The reified KB enables very simple end-to-end architectures to obtain competitive performance on several benchmarks representing two families of tasks: KB completion, and learning semantic parsers from denotations.

1 INTRODUCTION

There has been much prior work on using neural networks to *generalize* the contents of a KB (e.g., (Xiong et al., 2017; Bordes et al., 2013; Dettmers et al., 2018)), typically by constructing low-dimensional embeddings of the entities and relations in the KB, which are then used to score potential triples as plausible or implausible elements of the KB. We consider here the related but different problem of *incorporating a symbolic KB into a neural system*, so as to inject knowledge from an existing KB directly into a neural model. More precisely, we consider the problem of designing neural KB inference modules that are (1) fully differentiable, so that any loss based on their outputs can be backpropagated to their inputs (2) accurate, in that they are faithful to the original semantics of the KB; (3) expressive, so they can perform non-trivial inferences; and (4) scalable, so that realistically large KBs can be incorporated into a neural model.

To motivate the goal of incorporating a symbolic KB into a neural network, consider the task of learning neural semantic parsers from denotations. Many questions—e.g., *what's the most recent movie that Quentin Tarentino directed?* or *which nearby restaurants have vegetarian entrees and take reservations?*—are best answered by *knowledge-based question-answering* (KBQA) methods, where an answer is found by accessing a KB. Within KBQA, a common approach is *neural semantic parsing*—i.e., using neural methods to translate a natural-language question into a structured query against the KB (e.g., (Zhong et al., 2017; Finegan-Dollak et al., 2018; Shaw et al., 2019)), which is subsequently executed with a symbolic KB query engine. While this approach can be effective, it requires training data pairing natural-language questions with structured queries, which is difficult to obtain. Hence researchers have also considered *learning semantic parsers from denotations* (e.g., (Berant et al., 2013; Yih et al., 2015)), where training data consists of pairs (q, A), where q is a natural-language question and A is the desired answer, typically a set of KB entities—e.g., for the first sample question above, A would be[1] the singleton set containing *Once Upon a Time in Hollywood*.

Learning semantic parsers from denotations is difficult because the end-to-end process to be learned includes a non-differentiable operation—i.e., reasoning on the symbolic KB that contains the answers. To circumvent this difficulty, prior systems have used three different approaches. Some have used heuristic search to infer structured queries from denotations (e.g., (Pasupat & Liang, 2016; Dasigi et al., 2019)): this works in some cases but often an answer could be associated with many possible structured queries, introducing noise. Others have supplemented gradient approaches with reinforcement learning (e.g., (Misra et al., 2018)). Some systems have also "neuralized" KB reasoning, but to date only over *small* KBs: this approach is natural when answers are naturally constrained to

[1]At the time of this writing.

depend on a small set of facts (e.g., a single table (Zhong et al., 2017; Gupta & Lewis, 2018)), but more generally requires coupling a learner with some (non-differentiable) mechanism to retrieve an appropriate small question-dependent subset of the KB (e.g., (Sun et al., 2018; 2019)).

In this paper, we introduce a novel scheme for incorporating reasoning on a single, question-independent large KB into a neural network, by representing a symbolic KB with an encoding called a *sparse-matrix reified KB*. A sparse-matrix reified KB is very compact, can be distributed across multiple GPUs if necessary, and is well-suited to modern GPU architecture. For KBs with many relations, a reified KB can be up to four orders of magnitude faster than alternative implementations (even alternatives based on sparse-matrix representations), and in our experiments we demonstrate scalability to a KB with over 13 million entities and nearly 44 million facts. This new architectural component leads to radically simpler architectures for neural semantic parsing from denotations—architectures based on a single end-to-end differentiable process, rather than cascades of retrieval and neural processes.

We show that very simple instantiations of these architectures are still highly competitive with the state of the art for several benchmark tasks. *To our knowledge these models are the first fully end-to-end neural parsers from denotations that have been applied to these benchmark tasks.* We also demonstrate that these architectures scale to long chains of reasoning on synthetic tasks, and demonstrate similarly simple architectures for a second task, KB completion.

2 Neural reasoning with a symbolic KB

2.1 Background

KBs, entities, and relations, and types. A KB consists of *entities* and *relations*. We use x to denote an entity and r to denote a relation. A relation is a set of entity pairs, and represents a relationship between entities: for instance, if x represents "Quentin Tarentino" and x' represents "Pulp Fiction" then (x, x') would be an member of the relation *director_of*.

We assume each entity x has a *type*, written *type*(x), and let N_τ denote the number of entities of type τ. Each entity x in type τ also has a unique index $index_\tau(x)$, which is an integer between 1 and N_τ. We write $x_{\tau,i}$ for the entity that has index i in type τ, or x_i if the type is clear from context. Every relation r has a *subject type* τ_{subj} and an *object type* τ_{obj}, which constrain the types of x and x' for any pair $(x, x') \in r$. Hence r can be encoded as a subset of $\{1, \ldots, N_{\tau_{subj}}\} \times \{1, \ldots, N_{\tau_{obj}}\}$. Relations with the same subject and object types are called *type-compatible*. Finally a KB consists of a set of types, a set of typed relations, and a set of typed entities.

Representing weighted sets as vectors. Our differentiable operations are based on *weighted sets*, where each element x of weighted set X is associated with a non-negative real number, written $\omega[\![x \in X]\!]$. It is convenient to define $\omega[\![x \in X]\!] \equiv 0$ for all $x \notin X$. Conceptually, a weight less than 1 for element x is a confidence that the set contains x, and weights more than 1 make X a multiset. If all elements of X have weights 1, we say x is a *hard set*. Weighted sets X are also typed, and if *type*$(X) = \tau$ then X is constrained to contain only entities of type τ; hence X can be encoded as a subset of $\{1, \ldots, N_{\mathit{type}(X)}\}$.

Often we would like to reason about sets of relations[2], so we also assume every relation r in a KB is associated with an entity x_r, and hence, an index and a type. Sets of relations R are allowed if all members are type-compatible.[3] For example $R = \{\mathit{writer\_of}, \mathit{director\_of}\}$ might be a set of type-compatible relations.

A weighted set X of type τ can be encoded as an *entity-set vector* $\mathbf{v}_X \in \mathbb{R}^{N_\tau}$, where the i-th component of $\mathbf{v}_X$ is the weight of the i-th entity of that type in the set X: e.g., $\mathbf{v}_X[index_\tau(x)] = \omega[\![x \in X]\!]$. Notice that if X is a hard entity set, then this will be a k-hot vector for $k = |X|$. The set of indices of $\mathbf{v}$ with non-zero values is written *support*$(\mathbf{v})$, and we also use *type*$(\mathbf{v})$ to denote the type τ of the set encoded by $\mathbf{v}$.

Representing relations as sparse matrices. A relation r with subject type τ_1 and object type τ_2 can be encoded as a *relation matrix* $\mathbf{M}_r \in \mathbb{R}^{N_{\tau_1} \times N_{\tau_2}}$, where the value for $\mathbf{M}_r[i, j]$ is (in general)

[2] This is usually called *second-order* reasoning.

[3] To avoid complexity in notation we do not distinguish between sets of relations and sets of entities that are associated relations.

the weight of the assertion $r(x_i, x_j)$ in the KB, i.e., $\mathbf{M}_r[index_{\tau_1}(x), index_{\tau_2}(x')] = \omega[\![(x, x') \in r]\!]$. (However, in the experiments of this paper, all KB relations are hard sets, so $\mathbf{M}_r[i, j] \in \{0, 1\}$.)

For all but the smallest KBs, a relation matrix must be implemented using a *sparse matrix* data structure, as explicitly storing all $N_{\tau_1} \times N_{\tau_2}$ values is impractical.[4] A common sparse matrix data structure is a *sparse coordinate pair (COO)* structure, which consists of a $N_r \times 2$ matrix $\mathbf{Ind}_r$ containing pairs of entity indices, and a parallel vector $\mathbf{w}_r \in \mathbb{R}^{N_r}$ containing the weights of the corresponding entity pairs. In this encoding, if (i, j) is row k of **Ind**, then $\mathbf{M}_r[i, j] = \mathbf{w}_r[k]$, and if (i, j) does not appear in $\mathbf{Ind}_r$, then $\mathbf{M}[i, j]$ is zero. Hence with a COO encoding, each KB fact requires storing only two integers and one float.

Our implementations are based on Tensorflow (Abadi et al., 2016), which offers limited support for sparse matrices. In particular, driven by the limitations of GPU architecture, Tensorflow only supports matrix multiplication between a sparse matrix COO and a dense matrix, but not between two sparse matrices, or between sparse higher-rank tensors and dense tensors.

2.2 Reasoning in a KB

The relation-set following operation. Note that relations can also be viewed as labeled edges in a *knowledge graph*, the vertices of which are entities. Following this view, we define the *r-neighbors of an entity x* to be the set of entities x' that are connected to x by an edge labeled r, i.e., *r-neighbors(x)* $\equiv \{x' : (x, x') \in r\}$. Extending this to relation sets, we define

$$R\text{-}neighbors(X) \equiv \{x' : \exists r \in R, x \in X \text{ so that } (x, x') \in r\}$$

Computing the R-neighbors of an entity is a single-step reasoning operation: e.g., the answer to the question $q =$"*what movies were produced or directed by Quentin Tarentino*" is precisely the set R-neighbors(X) for $R = \{$*producer_of, writer_of*$\}$ and $X = \{$*Quentin_Tarentino*$\}$. "Multi-hop" reasoning operations require nested R-neighborhoods, e.g. if $R' = \{$*actor_of*$\}$ then R'-neighbors(R-neighbors(X)) is the set of actors in movies produced or directed by Quentin Tarentino.

We would like to approximate the R-neighbors computation with differentiable operations that can be performed on encodings of X and R. Let $\mathbf{v}_X$ encode a weighted set of entities X, and let $\mathbf{v}_R$ encode a weighted set of relations of type τ. We first define $\mathbf{M}_R$ to be a weighted mixture of the relation matrices for all relations of type τ, i.e., $\mathbf{M}_R \equiv (\sum_{k=1}^{N_{\tau_R}} \mathbf{v}_R[k] \cdot \mathbf{M}_{r_k})$. We then define the *relation-set following operation for $\mathbf{v}_X$ and $\mathbf{v}_R$* as:

$$follow(\mathbf{v}_X, \mathbf{v}_R) \equiv \mathbf{v}_X \mathbf{M}_R = \mathbf{v}_X (\sum_{k=1}^{N_{\tau_R}} \mathbf{v}_R[k] \cdot \mathbf{M}_k) \tag{1}$$

The numerical relation-set following operation of Eq 1 corresponds closely to the logical R-neighborhood operation, as shown by the claim below.

Claim 1 *The support of follow($\mathbf{v}_X, \mathbf{v}_R$) is exactly the set of R-neighbors(X).*

To better understand this claim, let $\mathbf{z} = follow(\mathbf{v}_X, \mathbf{v}_R)$. The claim states $\mathbf{z}$ can approximate the R neighborhood of any hard sets R, X by setting to zero the appropriate components of $\mathbf{v}_X$ and $\mathbf{v}_R$. It is also clear that $\mathbf{z}[j]$ decreases when one decreases the weights in $\mathbf{v}_r$ of the relations that link x_j to entities in X, and likewise, $\mathbf{z}[j]$ decreases if one decreases the weights of the entities in X that are linked to x_j via relations in R, so there is a smooth, differentiable path to reach this approximation.

2.3 Scalable relation-set following with a reified KB

Baseline implementations. Ignoring types for the moment, suppose the KB contains N_R relations, N_E entities, and N_T triples. Typically $N_R < N_E < N_T \ll N_E^2$. As noted above, we must implement each $\mathbf{M}_r$ as a sparse COO matrix, so collectively these matrices require space $O(N_T)$. Each triple appears in only one relation, so $\mathbf{M}_R$ in Eq 1 is also size $O(N_T)$. Since sparse-sparse matrix multiplication is not supported in Tensorflow we implement $\mathbf{x}\mathbf{M}_R$ using dense-sparse multiplication,

[4] For instance, if a KB contains 10,000 movie entities and 100,000 person entities, then a relationship like *writer_of* would require storing 1 billion values—far more than few tens of thousands of *writer_of* facts that would be in the KB (since most movies have only one or writers.)

Strategy	Batch?	Space complexity	# Operations		
			sp-dense matmul	dense + or $\odot$	sparse +
naive mixing	no	$O(N_T + N_E + N_R)$	1	0	N_R
late mixing	yes	$O(N_T + bN_E + bN_R)$	N_R	N_R	0
reified KB	yes	$O(bN_T + bN_E)$	3	1	0

Table 1: Summary of implementations of relation-set following, where N_T is the number of KB triples, N_E the number of entities, N_R the number of relations.

so $\mathbf{x}$ must be a dense vector of size $O(N_E)$, as is the output of relation-set following. So the space complexity of *follow*($\mathbf{x}, \mathbf{r}$) is $O(N_T + N_E + N_R)$, if implemented as suggested by Eq 1. We call this the *naive mixing* implementation (see Table 1).

The major problem with naive mxing is that, in the absence of general sparse tensor contractions, it is difficult to adapt to mini-batches—which generally make inference on GPUs much faster. We thus consider next a setting in which $\mathbf{x}$ and $\mathbf{r}$ are replaced with matrices $\mathbf{X}$ and $\mathbf{R}$ with minibatch size b. An alternative strategy is based on the observation that relation-set following for a *single* relation can be implemented as $\mathbf{x}\mathbf{M}_r$, which can be trivially extended to a minibatch as $\mathbf{X}\mathbf{M}_r$. The *late mixing* strategy mixes the *output* of many single-relation following steps, rather than mixing the KB itself:

$$\textit{follow}(\mathbf{X}, \mathbf{R}) = \sum_{k=1}^{N_{\tau_R}} (\mathbf{R}[:, k] \cdot \mathbf{X}\mathbf{M}_k) \quad (2)$$

Here $\mathbf{R}[:, k]$, the k-th column of $\mathbf{R}$, is "broadcast" to element of the matrix $\mathbf{X}\mathbf{M}_k$. While there are N_R matrices $\mathbf{X}\mathbf{M}_k$, each of size $O(bN_E)$, they need not all be stored at once, so the space complexity becomes $O(bN_E + bN_R + N_T)$; however we must now sum up N_R dense matrices.

A reified knowledge base. While semantic parses for natural questions often use small sets of relations (often singleton ones), there is substantial uncertainty about what these relations should in the learning process. Furthermore, realistic wide-coverage KBs have many relations—typically hundreds or thousands. When many relations are mixed, late mixing becomes quite expensive.

An alternative is to represent each KB assertion $r(e_1, e_2)$ as a tuple (i, j, k, w) where i, j, k are the indices of e_1, e_2, and r, and w is the confidence associated the triple. There are N_T such triples, so for $\ell = 1, \ldots, N_T$, let $(i_\ell, j_\ell, k_\ell, w_\ell)$ denote the ℓ-th triple. Let $\delta[\![a = b]\!]$ denote 1 if $a = b$ and 0 otherwise. We now define these sparse matrices, which collectively define the *sparse-matrix reified KB*:[5]

$$\mathbf{M}_{subj}[\ell, m] \equiv \delta[\![m = i_\ell]\!] \;;\; \mathbf{M}_{obj}[\ell, m] \equiv \delta[\![m = j_\ell]\!] \;;\; \mathbf{M}_{rel}[\ell, m] \equiv w_\ell \cdot \delta[\![m = k_\ell]\!]$$

Conceptually, $\mathbf{M}_{subj}$ maps the index ℓ of the ℓ-th triple to its subject entity; $\mathbf{M}_{obj}$ maps ℓ to the object entity; and $\mathbf{M}_{rel}$ maps ℓ to the relation of the ℓ-th triple, and incidentally encodes the triple confidence w_ℓ. We can now implement the relation-set following as below, where $\odot$ is Hadamard product:

$$\textit{follow}(\mathbf{X}, \mathbf{R}) = (\mathbf{X}\mathbf{M}_{subj}^T \odot \mathbf{R}\mathbf{M}_{rel}^T)\mathbf{M}_{obj} \quad (3)$$

For a single $\mathbf{x}, \mathbf{r}$ notice that $\mathbf{x}\mathbf{M}_{subj}^T$ are the triples with an entity in $\mathbf{x}$ as their subject, $\mathbf{r}\mathbf{M}_{rel}^T$ are the triples with a relation in $\mathbf{r}$, and the Hadamard product is the intersection of these. The final multiplication by $\mathbf{M}_{obj}$ finds the object entities of the triples in the intersection. These operations naturally extend to minibatches, as given in Eq 3. The reified KB has size $O(N_T)$, the sets of triples that are intersected have size $O(bN_T)$, and the final result is size $O(bN_E)$, giving a final size of $O(bN_T + bN_E)$, with no dependence on N_R.

Table 1 summarizes the complexity of the three mathematically equivalent but computationally different implementions given in Eqs 1, 2, and 3. The analysis suggests that the reified KB is preferable if there are many relations, which is the case for most realistic KBs[6].

Distributing a large reified KB. The reified KB representation is quite compact, using only six integers and three floats for each KB triple. However, since GPU memory is often limited, it is

[5] Reification in logic is related to reflection in programming languages.

[6] The larger benchmark datasets used in this paper have 200 and 616 relations respectively, and the widely used FreeBase-15k-237 has 1345 relations.

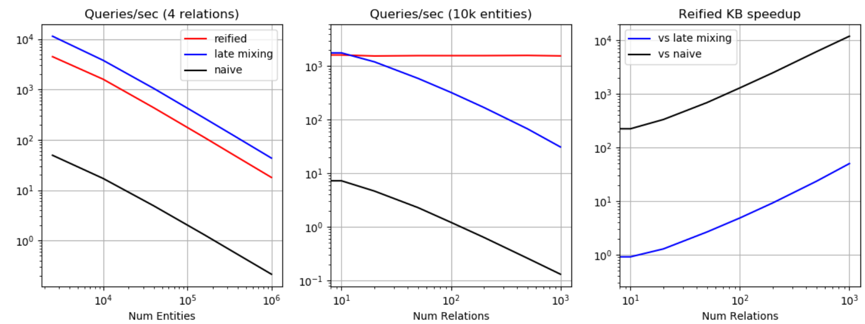

Figure 1: Left and middle: inference time in queries/sec on a synthetic KB as size and number of relations is varied. Right: speedups of reified KBs over the baseline implementations.

important to be able to *distribute* a KB across multiple GPUs. Fortunately matrix multiplication **XM** can be decomposed: **X** can be split into a "horizontal stacking" of m submatrices, which we write as $[\mathbf{X}_1; \ldots; \mathbf{X}_m]$, and **M** can be similarly partitioned into m^2 submatrices, and then

$$\mathbf{XM} = [\mathbf{X}_1; \mathbf{X}_2; \ldots; \mathbf{X}_m] \begin{bmatrix} \mathbf{M}_{1,1} & \mathbf{M}_{1,2} & \ldots & \mathbf{M}_{1,m} \\ \vdots & \vdots & & \vdots \\ \mathbf{M}_{m,1} & \mathbf{M}_{m,2} & \ldots & \mathbf{M}_{m,m} \end{bmatrix} = \left[(\sum_{i=1}^{m} \mathbf{X}_1 \mathbf{M}_{i,1}); \ldots; (\sum_{i=1}^{m} \mathbf{X}_m \mathbf{M}_{i,m}) \right]$$

This can be computed without storing either **X** or **M** on a single machine, and mathematically applies to both dense and sparse matrices. (Although to our knowledge prior implementations of distributed matrix operations (e.g., (Shazeer et al., 2018)) do not support sparse matrices.) *We thus implemented a distributed sparse-matrix implementation of reified KBs*. We distibuted the matrices that define a reified KB "horizontally", so that different triple ids ℓ are stored on different GPUs.

3 Experiments

3.1 Scalability

Like prior work (Cohen et al., 2017; De Raedt et al., 2007), we used a synthetic KB based on an n-by-n grid to study scalability of inference. Every grid cell is an entity, related to its immediate neighbors, via relations *north*, *south*, *east*, and *west*. The KB for an n-by-n grid thus has $O(n^2)$ entities and around $O(n)$ triples. We measured the time to compute the 2-hop inference *follow*(*follow*($\mathbf{X}, \mathbf{R}$), $\mathbf{R}$) for minibatches of $b = 128$ one-hot vectors, and report it as queries per second (qps) on a single GPU (e.g., qps=1280 would mean a single minibatch requires 100ms).[7]

The results are shown Figure 1 (left and middle), on a log-log scale because some differences are very large. With only four relations (the leftmost plot), late mixing is about 3x faster than the reified KB method, and about 250x faster than the naive approach. However, for more than around 20 relations, the reified KB is faster (middle plot). As shown in the rightmost plot, the reified KB is 50x faster than late mixing with 1000 relations, and nearly 12,000x faster than the naive approach.[8]

3.2 Models using reified KBs

As discussed below in Section 4, the reified KB is closely related to key-value memory networks, so it can be viewed as a more efficient implementation of existing neural modules, optimized for reasoning with symbolic KBs. However, being able to include an entire KB into a model can lead to a *qualitative* difference in model complexity, since it is not necessary to build machinery to retrieve from the KB. To illustrate this, below we present simple models for several tasks, each using the reified KB in different ways, as appropriate to the task. We consider two families of tasks: learning semantic parsers from denotations over a large KB, and learning to complete a KB.

[7] The matrix **R** weights all relations uniformly, and we vary the number of relations by inventing m new relation names and assigning existing grid edges to each new relation.

[8] Although we do not show the results, even the method we call "naive" here is *much* more memory-efficient than using dense relation-matrices. We do not show results for smaller minibatch sizes, but both reified and late mixing are about 40x slower with $b = 1$ than with $b = 128$.

KBQA for multi-hop questions. MetaQA (Zhang et al., 2018) consists of 1.2M questions, evenly distributed into one-hop, two-hop, and three-hop questions. (E.g, the question "*who acted in a movie directed by Quentin Tarentino?*" is a two-hop question.) The accompanying KB (Miller et al., 2016) contains 43k entities and 186k facts. Past work treated one-hop, two-hop and three-hop questions separately, and the questions are labeled with the entity ids for the "seed entities" that begin the reasoning chains (e.g., the question above would be tagged with the id of the entity for *Quentin Tarentino*).

Using a reified KB for reasoning means the neural model only needs to predict the relations used at each stage in the reasoning process. For each step of inference we thus compute relation sets $\mathbf{r}^t$ using a differentiable function of the question, and then chain them together with relation-set following steps. Letting $\mathbf{x}^0$ be the set of entities associated with q, the model we use is:

$$\text{for } t = 1, 2, 3: \quad \mathbf{r}^t = f^t(q); \quad \mathbf{x}^t = \textit{follow}(\mathbf{x}^{t-1}, \mathbf{r}^t)$$

To predict an answer on a T-hop subtask, we computed the softmax of the appropriate set $\mathbf{x}^T$. We used cross entropy loss of this set against the desired answer, represented as a uniform distribution over entities in the target set. Each $f^t(q)$'s is a different linear projection of a common encoding for q, specifically a mean-pooling of the tokens in q encoded with a pre-trained 128-dimensional word2vec model (Mikolov et al., 2013). The full KB was loaded into a single GPU in our experiments.

Note that in this model *the problem of learning the structured KB query has been reduced to training differentiable functions of the question*, and the loss on predicted denotations is back-propagated to these functions. Similar strategies are used in all the other models presented below.

KBQA on FreeBase. WebQuestionsSP (Yih et al., 2016) contains 4737 natural language questions, all of which are answerable using FreeBase (Bollacker et al., 2008), a large open-domain KB. Each question q is again labeled with the entities $\mathbf{x}$ that appear in it.

Above we ignored types in the KB, but here we consider them, as this leads to smaller vectors and faster inference. FreeBase contains two disjoint node types: real-world *entities*, and *compound value types* (CVTs), which represent non-binary relationships or events (e.g., a movie release event, which includes a movie id, a date, and a place.) Real-world entity nodes can be related to each other or to a CVT node, but CVT nodes are never directly related to each other, so there are there are three groupings of type-compatible relations: (1) relations mapping entities to entities, (2) relations mapping entities to CVT nodes; and (3) relations mapping CVT nodes to entity nodes.

In this dataset, all questions can be answered with 1- or 2-hop chains, and all 2-hop reasoning chains pass through a CVT entity; however, unlike MetaQA, the number of hops is not known. Our model thus derives from q three relation sets, one of for each grouping described above, then uniformly mixes both potential types of inferences:

$$\mathbf{r}_{\text{E}\to\text{E}} = f_{\text{E}\to\text{E}}(q); \quad \mathbf{r}_{\text{E}\to\text{CVT}} = f_{\text{E}\to\text{CVT}}(q); \quad \mathbf{r}_{\text{CVT}\to\text{E}} = f_{\text{CVT}\to\text{E}}(q)$$
$$\hat{\mathbf{a}} = \textit{follow}(\textit{follow}(\mathbf{x}, \mathbf{r}_{\text{E}\to\text{CVT}}), \mathbf{r}_{\text{CVT}\to\text{E}}) + \textit{follow}(\mathbf{x}, \mathbf{r}_{\text{E}\to\text{E}})$$

We again apply a softmax to $\hat{\mathbf{a}}$ and use cross entropy loss, and $f_{\text{E}\to\text{E}}$, $f_{\text{E}\to\text{CVT}}$, and $f_{\text{CVT}\to\text{E}}$ are again linear projections of a word2vec encoding of q.

In the experiments below we used a subset of Freebase with 43.7 million facts and 12.9 million entities, containing all facts in Freebase within 2-hops of entities mentioned in any question, excluding paths through some very common entities. We split the KB across three 12-Gb GPUs, and used a fourth GPU for the rest of the model.

Knowledge base completion. Following Yang et al. (2017) we treat KB completion as an inference task, analogous to KBQA: a query q is a relation name and a head entity $\mathbf{x}$, and from this we predict a set of tail entities. We assume the answers are computed with the disjunction of multiple inference chains of varying length. Each inference chain has a maximum length of T and we build N distinct inference chains in total, using this model:

$$\text{for } i = 1, \ldots, N \text{ and } t = 1, \ldots, T: \quad \mathbf{r}_i^t = f_i^t(q); \quad \mathbf{x}_i^{t+1} = \textit{follow}(\mathbf{x}_i^t, \mathbf{r}_i^t) + \mathbf{x}_i^t$$

The final output is a softmax of the mix of all the $\mathbf{x}_i^T$'s: i.e., we let $\hat{\mathbf{a}} = \textit{softmax}(\sum_{i\in\{1\ldots N\}} \mathbf{x}_i^T)$. The purpose of the update $\mathbf{x}_i^{t+1} = \textit{follow}(\mathbf{x}_i^t, \mathbf{r}_i^t) + \mathbf{x}_i^t$ is to give the model access to outputs of all chains

	ReifKB (ours)	ReifKB + mask	KV-Mem (baseline)	VRN	GRAFTNet	non-differentiable components of architectures	
WebQSP	52.7	—	46.7	—	**67.8**	KV-Mem	initial memory retrieval
MetaQA							
1-hop	96.2	—	95.8	**97.5**	97.0		
2-hop	81.1	**95.4**	25.1	89.9	94.8	VRN	question-specific
3-hop	72.3	**79.7**	10.1	62.5	77.2	GRAFTNet	subgraph retrieval
Grid							
5-hop	98.4	—	—	—	—		
10-hop	89.7	—	—	—	—	ReifKB(ours)	*none*

Table 2: Hits@1 on the KBQA datasets. Results for KV-Mem and VRN on MetaQA are from (Zhang et al., 2018); results for GRAFT-Net and KV-Mem on WebQSP are from (Sun et al., 2018).

of length less than t. The encoding of q is based on a lookup table, and each relation vector f_i^t is a learned linear transformation of q's embedding.[9]

An encoder-decoder architecture for varying inferential structures. To explore performance on more complex reasoning tasks, we generated simple artificial natural-language sentences describing longer chains of relationships on a 10-by-10 grid. For this task we used an encoder-decoder model which emits chains of relation-set following operations. The question is encoded with the final hidden state of an LSTM, written here $\mathbf{h}_0$. We then generate a reasoning chain of length up to T using a decoder LSTM. At iteration t, the decoder emits a scalar probability of "stopping", p^t, and a distribution over relations to follow $\mathbf{r}^t$, and then, as we did for the KBQA tasks, sets $\mathbf{x}^t = \textit{follow}(\mathbf{x}^{t-1}, \mathbf{r}^t)$. Finally the decoder updates its hidden state to $\mathbf{h}^t$ using an LSTM cell that "reads" the "input" $\mathbf{r}^{t-1}$. For each step t, the model thus contains the steps

$$p^t = f_p(\mathbf{h}^{t-1}); \quad \mathbf{r}^t = f_r(\mathbf{h}^{t-1}); \quad \mathbf{x}^t = \textit{follow}(\mathbf{x}^{t-1}, \mathbf{r}^t); \quad \mathbf{h}^t = \text{LSTM}(\mathbf{h}^{t-1}, \mathbf{r}^{t-1})$$

The final predicted location is a mixture of all the $\mathbf{x}_t$'s weighted by the probability of stopping p_t at iteration t, i.e., $\hat{\mathbf{a}} = \textit{softmax}(\sum_{t=1}^{T} \mathbf{x}^t \cdot p^t \prod_{t'<t}(1 - p^{t'}))$. The function f_r is a softmax over a linear projection, and f_p is a logistic function. In the experiments, we trained on 360,000 sentences requiring between 1 and T hops, for $T = 5$ and $T = 10$, and tested on an additional 12,000 sentences.

Experimental results. We next consider the performance of these models relative to strong baselines for each task. We emphasize our goal here is *not to challenge the current state of the art on any particular benchmark*, and clearly there are many ways the models of this paper could be improved. (For instance, our question encodings are based on word2vec, rather than contextual encodings (e.g., (Devlin et al., 2018)), and likewise relations are predicted with simple linear classifiers, rather than, say, attention queries over some semantically meaningful space, such as might be produced with language models or KB embedding approaches (Bordes et al., 2013)). Rather, our contribution is to present a generally useful scheme for including symbolic KB reasoning into a model, and we have thus focused on describing simple, easily understood models that do this for several tasks. However, it is important to confirm that the reified KB models "work"—e.g., that they are amenable to use of standard optimizers, etc.

Performance (using Hits@1) of our models on the KBQA tasks is shown in Table 2. For the non-synthetic tasks we also compare to a Key-Value Memory Network (KV-Mem) baseline (Miller et al., 2016). For the smaller MetaQA dataset, KV-Mem is initialized with all facts within 3 hops of the query entities, and for WebQuestionsSP it is initialized by a random-walk process seeded by the query entities (see (Sun et al., 2018; Zhang et al., 2018) for details). ReifKB consistently outperforms the baseline, dramatically so for longer reasoning chains. The synthetic grid task shows that there is very little degradation as chain length increases, with Hits@1 for 10 hops still 89.7%. It also illustrates the ability to predict entities in a KB, as well as relations.

We also compare these results to two much more complex architectures that perform end-to-end question answering in the same setting used here: VRN (Zhang et al., 2018) and GRAFT-Net (Sun et al., 2018). Both systems build question-dependent subgraphs of the KB, and then use graph CNN-like methods (Kipf & Welling, 2016) to "reason" with these graphs. Although not superior, ReifKB model is competitive with these approaches, especially on the most difficult 3-hop setting.

[9] In the experiments we tune the hyperparameters $T \in \{1, \ldots, 6\}$ and $N \in \{1, 2, 3\}$ on a dev set.

	NELL-995	
	H@1	H@10
ReifKB (Ours)	64.1	82.4
DistMult*	61.0	79.5
ComplEx*	61.2	82.7
ConvE*	**67.2**	**86.4**

	ReifKB (Ours)	MINERVA
NELL-995	64.1	**66.3**
Grid with seed entity		
10-hop NSEW	98.9	**99.3**
10-hop NSEW-VH	**73.6**	34.4
MetaQA 3-hop	**72.3**	41.7

Table 3: Left: Hits@1 and Hits@10 for KB completion on NELL 995. Starred KB completion methods are transductive, and do not generalize to entities not seen in training. Right: Comparison to MINERVA on several tasks for Hits@1.

	NELL-995	MetaQA	WebQuestionsSP
# Facts	154,213	196,453	43,724,175
# Entities	75,492	43,230	12,942,798
# Relations	200	9	616
Time (seconds)	44.3	72.6	1820

Table 4: Time to run 10K examples for KBs of different size.

We also observed that in the MetaQA 2-hop and 3-hop questions, the questions often exclude the seed entities (e.g., “other movies with the same director as Pulp Fiction”). This can be modeled by masking out seed entities from the predictions after the second hop (ReifKB + mask); this slightly extended model leads to better performance than GRAFT-Net on 2-hop and 3-hop questions.

For KB completion, we evaluated the model on the NELL-995 dataset (Xiong et al., 2017) which is paired with a KB with 154k facts, 75k entities, and 200 relations. On the left of Table 3 we compare our model with three popular embedding approaches (results are from Das et al. (2017)). The reified KB model outperforms DistMult (Yang et al., 2014), is slightly worse than ConvE (Dettmers et al., 2018), and is comparable to ComplEx (Trouillon et al., 2017). However, all these approaches are quite different from our model, because they are *transductive*—they only make predictions for entities seen in training. This is a substantial disadvantage in KBC, since it means that when new entities are added to a KB no predictions can be made for them.

As a non-transductive baseline, we also compared to the MINERVA model, which uses reinforcement learning (RL) methods to learn how to traverse a KB to find a desired answer. Although RL methods less suitable as “neural modules”, MINERVA is a arguably a plausible competitor to end-to-end learning with a reified KB. MINERVA slightly outperforms our simple KB completion model on the NELL-995 task. However, unlike our model, MINERVA is trained to find a *single answer*, rather than trained to infer a *set of answers*. To explore this difference, we compared to MINERVA on the grid task under two conditions: (1) the KB relations are the grid directions north, south, east and west, so only the result of the target chain is a *single* grid location, and (2) the KB relations also include a “vertical move” (north or south) and a “horizontal move”), so the result of the target chain can be a *set* of locations. As expected MINERVA’s performance drops dramatically in the second case, from 99.3% Hits@1 to 34.4 %, while our model’s performance is more robust. MetaQA answers can also be sets, so we also modified MetaQA so that MINERVA could be used[10] and noted a similarly poor performance. These results are shown on the right of Table 3.

Finally in Table 4 we compare the training time of our model with minibatch size of 10 on NELL-995, MetaQA, and WebQuestionsSP. With over 40 million facts and nearly 13 million entities from Freebase, it takes less than 10 minutes to run one epoch over WebQuestionsSP (with 3097 training examples) on four P100 GPUs.

4 Related Work

The relation-set following operation using reified KBs is implemented in an open-source package called NQL, for neural query language. NQL implements a broader range of operations for manipulating KBs, which are described in a short companion paper (Authors, 2019). This paper focuses on implementation and evaluation of the relation-set following operation with different KB representations, issues not covered in the companion paper.

NQL is semantically related to TensorLog (Cohen et al., 2017), a probabilistic logic which also can be compiled to Tensorflow, and hence is another differentiable approach to neuralizing a KB. However,

[10]By making the non-entity part of the sentence the “relation” input and the seed entity the “start node” input.

TensorLog does not support relation *sets*, making it unnatural to express the models shown in this paper, and does not use the more efficient reified KB representation. The differentiable theorem prover (DTP) is another differentiable logic (Rocktäschel & Riedel, 2017), but DPT appears to be much less scalable: it has not been applied to KBs larger than a few thousand triples. The Neural ILP system (Yang et al., 2017) uses approaches related to late mixing together with an LSTM controller to perform KB completion and some simple QA tasks, but it is a monolithic architecture focused on rule-learning, while in contrast we propose a re-usable neural component, which can be used in as a component in many different architectures, and a scalable implementation of this. It is also reported that neural ILP does not scale to the size of the NELL995 task (Das et al., 2017).

Relation-set following is much like the bilinear model for path following from Guu et al. (2015). However, we generalize this to path queries that include weighted sets of relations, allowing the relations in paths to be learned. Guu et al. (2015) also show KB embeddings produce cascaded errors in compositional reasoning, a problem only partially addressed by their methods, which involve low-rank dense approximations; in contrast we focus on accurate, models of reasoning an a symbolic KB, which requires consideration of novel scalability issues. Similar differences apply to the work of Hamilton et al. (2018).

Neural architectures like memory networks (Weston et al., 2014), or other architectures that use attention over some data structure approximating assertions (Andreas et al., 2016; Gupta & Lewis, 2018) can be used to build soft versions of relation-set following: however, they also do not scale well to large KBs, so they are typically used either with a non-differentiable *ad hoc* retrieval mechanism, or else in cases where a small amount of information is relevant to a question (e.g., (Weston et al., 2015; Zhong et al., 2017)). Similarly graph CNNs (Kipf & Welling, 2016) also can be used for reasoning, and often do use sparse matrix multiplication, but again existing implementations have not been scaled to tens of millions of triples/edges or millions of entities/graph nodes. Additionally while graph CNNs have been used for reasoning tasks, the formal connection between them and logical reasoning remains unclear, whereas there is a precise connection between relation-set following and inference.

Reinforcement learning (RL) methods been used to learn mappings from natural-language questions to non-differentiable logical representations (Liang et al., 2016; 2018) and have also been applied to KB completion tasks (Das et al., 2017; Xiong et al., 2017). (Above we compared experimentally to MINERVA, one such method.) However, the gradient-based approaches enabled by our methods are generally preferred as being easier to implement and tune on new problems, and easier to combine in a modular way with other architectural elements.

5 CONCLUSIONS

We introduced here a novel way of representing a symbolic knowledge base (KB) called a sparse-matrix reified KB. This representation enables neural modules that are fully differentiable, faithful to the original semantics of the KB, expressive enough to model multi-hop inferences, and scalable enough to use with realistically large KBs. In a reified KB, all KB relations are represented with three sparse matrices, which can be distributed across multiple GPUs, and symbolic reasoning on realistic KBs with many relations is much faster than with naive implementations—more than four orders of magnitude faster on synthetic-data experiments compared to naive sparse-matrix implementations.

The simple vector representation used for weighted sets in a reified KB makes several operations trivial to implement: set union corresponds to vector sum, intersection to Hadamard product, and the complement of X is $1 - \mathbf{v}_X$. Interestingly, only a single additional neural operation is needed to support the KB reasoning tasks needed for the five benchmark tasks considered here, namely the relation-set following operation, written $\textit{follow}(\mathbf{v}_X, \mathbf{v}_R)$. All the KB reasoning required here can be implemented by composing these operations.

This new architectural component leads to radically simpler architectures for neural semantic parsing from denotations and KB completion—in particular, they make it possible to learn neural KBQA models in a completely end-to-end way, mapping from text to KB entity sets, for KBs with tens of millions of triples and entities and hundreds of relations.

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283, 2016.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 39–48, 2016.

Anonymous Authors. Neural query language: A knowledge base query language for tensorflow. Details withheld to preserve anonymity of this submission, 2019.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1533–1544, 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 the 2008 ACM SIGMOD international conference on Management of data*, pp. 1247–1250. AcM, 2008.

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.

William W Cohen, Fan Yang, and Kathryn Rivard Mazaitis. Tensorlog: Deep learning meets probabilistic dbs. *arXiv preprint arXiv:1707.05390*, 2017.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851*, 2017.

Pradeep Dasigi, Matt Gardner, Shikhar Murty, Luke Zettlemoyer, and Eduard Hovy. Iterative search for weakly supervised semantic parsing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2669–2680, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1273. URL `https://www.aclweb.org/anthology/N19-1273`.

Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI*, volume 7, pp. 2462–2467. Hyderabad, 2007.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

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

Catherine Finegan-Dollak, Jonathan K Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. Improving text-to-sql evaluation methodology. *arXiv preprint arXiv:1806.09029*, 2018.

Nitish Gupta and Mike Lewis. Neural compositional denotational semantics for question answering. *CoRR*, abs/1808.09942, 2018. URL `http://arxiv.org/abs/1808.09942`.

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

Will Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. Embedding logical queries on knowledge graphs. In *Advances in Neural Information Processing Systems*, pp. 2026–2037, 2018.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *arXiv preprint arXiv:1611.00020*, 2016.

Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc V Le, and Ni Lao. Memory augmented policy optimization for program synthesis and semantic parsing. In *Advances in Neural Information Processing Systems*, pp. 9994–10006, 2018.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013.

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. *CoRR*, abs/1606.03126, 2016. URL `http://arxiv.org/abs/1606.03126`.

Dipendra Misra, Ming-Wei Chang, Xiaodong He, and Wen-tau Yih. Policy shaping and generalized update equations for semantic parsing from denotations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2442–2452, 2018.

Panupong Pasupat and Percy Liang. Inferring logical forms from denotations. *arXiv preprint arXiv:1606.06900*, 2016.

Tim Rocktäschel and Sebastian Riedel. End-to-end differentiable proving. In *Advances in Neural Information Processing Systems*, pp. 3788–3800, 2017.

Peter Shaw, Philip Massey, Angelica Chen, Francesco Piccinno, and Yasemin Altun. Generating logical forms from graph representations of text and entities. *arXiv preprint arXiv:1905.08407*, 2019.

Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyoukJoong Lee, Mingsheng Hong, Cliff Young, Ryan Sepassi, and Blake A. Hechtman. Mesh-tensorflow: Deep learning for supercomputers. *CoRR*, abs/1811.02084, 2018. URL `http://arxiv.org/abs/1811.02084`.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W Cohen. Open domain question answering using early fusion of knowledge bases and text. *EMNLP*, 2018.

Haitian Sun, Tania Bedrax-Weiss, and William W Cohen. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. *arXiv preprint arXiv:1904.09537*, 2019.

Théo Trouillon, Christopher R Dance, Éric Gaussier, Johannes Welbl, Sebastian Riedel, and Guillaume Bouchard. Knowledge graph completion via complex tensor factorization. *The Journal of Machine Learning Research*, 18(1):4735–4772, 2017.

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

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

Wenhan Xiong, Thien Hoang, and William Yang Wang. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690*, 2017.

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

Fan Yang, Zhilin Yang, and William W Cohen. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems*, pp. 2319–2328, 2017.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pp. 1321–1331, 2015.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pp. 201–206, 2016.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. Variational reasoning for question answering with knowledge graph. In *AAAI*, 2018.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 2017.