Memory Graph Networks for Explainable Memory-grounded Question Answering

Seungwhan Moon, Pararth Shah, Anuj Kumar, Rajen Subba
Facebook Assistant
{shanemoon, pararths, anujk, rasubba@}fb.com

Abstract

We introduce Episodic Memory QA, the task of answering personal user questions grounded on memory graph (MG), where episodic memories and related entity nodes are connected via relational edges. We create a new benchmark dataset first by generating synthetic memory graphs with simulated attributes, and by composing 100K QA pairs for the generated MG with bootstrapped scripts. To address the unique challenges for the proposed task, we propose Memory Graph Networks (MGN), a novel extension of memory networks to enable dynamic expansion of memory slots through graph traversals, thus able to answer queries in which contexts from multiple linked episodes and external knowledge are required. We then propose the Episodic Memory QA Net with multiple module networks to effectively handle various question types. Empirical results show improvement over the QA baselines in top-k answer prediction accuracy in the proposed task. The proposed model also generates a graph walk path and attention vectors for each predicted answer, providing a natural way to explain its QA reasoning.

1 Introduction

The task of question and answering (QA) has been extensively studied, where many of the existing applications and datasets have been focused on the fact retrieval task from a large-scale knowledge graph (KG) (Bordes et al., 2015), or machine reading comprehension (MRC) approaches given unstructured text (Rajpurkar et al., 2018). In this work, we introduce the new task and dataset for **Episodic Memory QA**, in which the model answers personal and retrospective questions based on **memory graphs** (MG), where each episodic memory and its related entities (*e.g.* knowledge graph (KG) entities, participants, ...) are represented as the nodes connected via corresponding edges (Figure 1). Examples of such queries include "*Where did we go **after** we had brunch with Jon?*", "*How many times did I go to **jazz** concerts last year?*", etc. For Episodic Memory QA, a machine has to understand the contexts of a question and navigate multiple MG episode nodes as well as KG nodes to gather comprehensive information to match the query requirement.

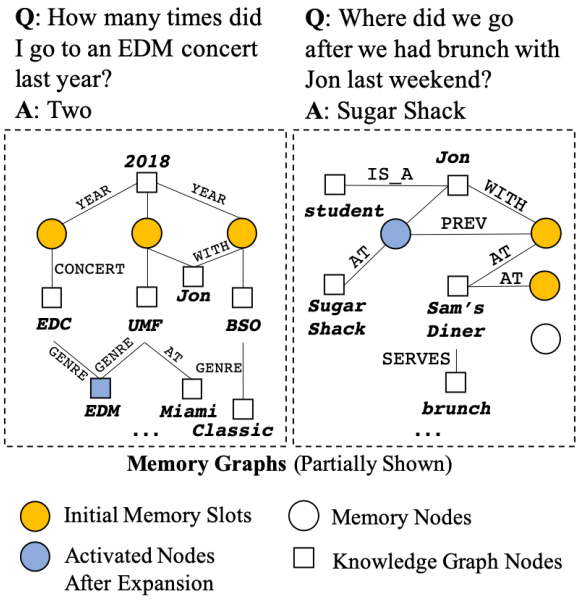

Figure 1: Illustration of **Episodic Memory QA** with user queries and memory graphs (MG) with knowledge graph (KG) entities. Relevant memory nodes are provided as initial memory slots via graph search lookup. The Memory Graph Network walks from the initial nodes to attend to relevant contexts and expands the memory slots when necessary. The main QA model takes these graph traversal paths and expanded memory slots as input, and predicts correct answers via multiple module networks (*e.g.* `COUNT`, `CHOOSE`, etc.)

While the ability of querying a personal database could lead to many potential applications, previous work in this domain (Jiang et al., 2018) is limited due to the lack of a large-scale

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 728–736
Hong Kong, China, November 3-4, 2019.

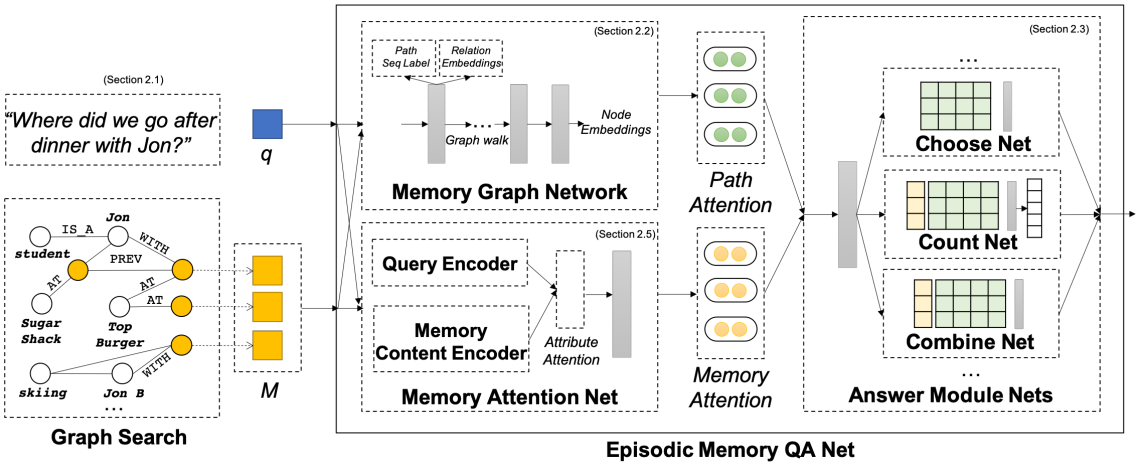

Figure 2: **Overall architecture** of the Episodic Memory QA Network. For an input query $\mathbf{q}$, candidate memory nodes $\mathbf{m} = \{\mathbf{m}^{(k)}\}$ are provided as input memory slots for the Memory QA Network. The Memory Graph Network then traverses the memory graph to expand the initial memory slots and activate other relevant entity and memory nodes based on the input queries. The Answer Module Networks execute the predicted neural programs to decode answers given the memory graph network outputs.

dataset and a unique set of challenges unseen in other tasks: For example, we observe that 1) Memory QA queries often include ambiguous and incomplete descriptions of reference memory (as opposed to many conventional Fact QAs with unambiguous mentions, *e.g.* "*Who painted the Mona Lisa?*"), hence requiring extensive candidate memory generation. Another challenge we observe is the case where 2) target memory is only indirectly linked to reference memory or entities (*e.g.* "*Where did we go after brunch?*"), which makes the conventional information retrieval (IR) approaches for generating answer candidates ineffective. In addition, 3) queries are not confined to retrieval tasks, but include various types of questions such as counting, set comparing, etc., many of which remain unsolved or not considered in many QA tasks.

To this end, we propose a new model called the Memory Graph Network (MGN) to address the specific challenges stated above that come with Memory QA. While Memory Networks have successfully been used in QA applications, typical limitations are that memory slots are limited to the fixed number of slots, often in sentence or bag-of-symbols forms. MGN extends the popular memory networks by storing graph nodes as memory slots and by allowing the network to dynamically expand memory slots through graph traversals. We then implement the main Episodic Memory QA Network with multiple module networks such as `CHOOSE`, `COUNT`, etc., to effectively handle various question types not easily handled via graph networks.

To bootstrap a large-scale dataset collection for Episodic Memory QA, we first build a synthetic memory graph generator, which creates multiple episodic memory graph nodes connected with real entities (*e.g.* locations, events, public entities) appearing on common-fact KGs. By creating a realistic memory graph that is synthetically generated, we avoid the need for inferring memory graphs from other structured data (*e.g.* photo albums) which are often limited in size. We then generate 100K QA pairs for each memory node with templates composed by human annotators, combined with 1K manual paraphrasing steps. More details for dataset collection are provided in Section 3.

2 Method

Figure 2 illustrates the overall architecture and the model components that make up the Episodic Memory QA Net. We examine each module in detail and provide our rationale about its formulation in the following sections.

Input Module (Section 2.1): For a given query $\mathbf{q}$, its relevant memory nodes $\mathbf{m} = \{\mathbf{m}^{(k)}\}_{k=1}^{K}$ for slot size K are given as initial memory slots. At test time, relevant memory nodes can be retrieved from a graph search engine that measures textual similarity (*e.g.* n-gram TF-IDF) between its connected node contexts and query. The Query Encoder then encodes the input query with a language model, and the Memory Encoder encodes each memory slot for both structural and semantic properties of each memory.

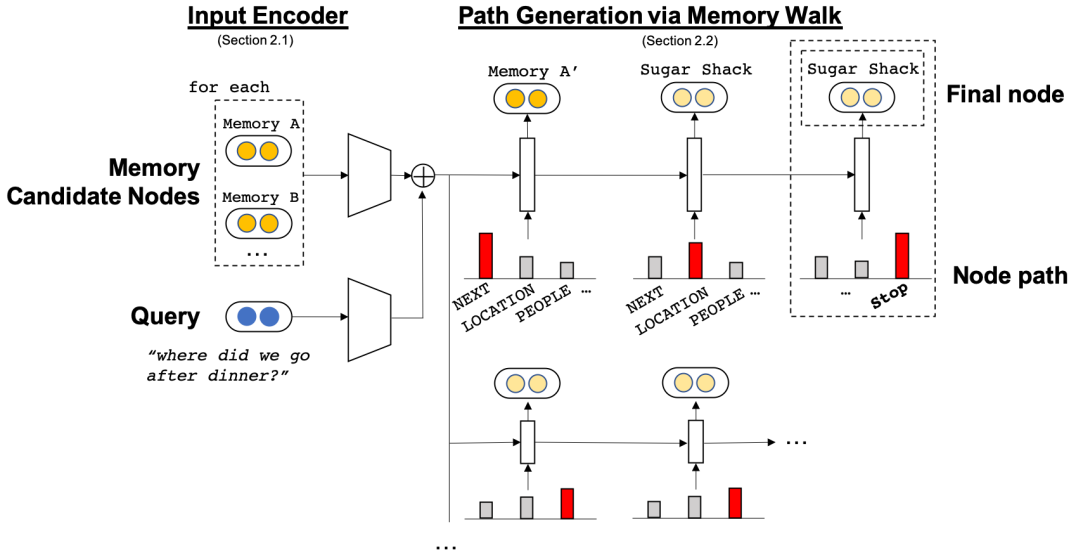

Figure 3: **Memory Graph Network (MGN) Walker**. Input query $\mathbf{q}$ and candidate memory nodes $\mathbf{m} = \{\mathbf{m}^{(k)}\}$ are encoded with the query encoder and the memory encoder, respectively (left). The decoder (right) predicts both the optimal paths and the final nodes $\mathbf{p} = \{\mathbf{p}_t\}_{t=1}^{T}$ based on their relevance scores as well as soft-attention based walk paths, which expands the initial memory slots.

Memory Graph Networks (MGN) (Section 2.2): Many previous work in QA or MRC systems use memory networks to evaluate multiple answer candidates with transitive reasoning, and typically store all potentially relevant raw sentences or bag-of-symbols as memory slots. However, naive increase of memory slot size or retention-based sequential update of memory slots often increase search space for answer candidates, leading to poor precision especially for the Episodic Memory QA task. To overcome this issue, with MGN we store memory graph nodes as initial memory slots, where additional contexts and answer candidates can be succinctly expanded and reached via graph traversals. For each $(\mathbf{q}, \mathbf{m}^{(k)})$ pair, MGN predicts optimal memory slot expansion steps: $\mathbf{p}^{(\mathbf{k})} = \{[\mathbf{p}_{e,t}^{(k)}; \mathbf{p}_{n,t}^{(k)}]\}_{t=1}^{T}$ for edge paths $\mathbf{p}_e$ and corresponding node paths $\mathbf{p}_n$ (Figure 3).

QA Modules (Section 2.3, 2.4): An estimated answer $\hat{\mathbf{a}} = \text{QA}(\mathbf{m}, \mathbf{q})$ is predicted given a query and MGN graph path output from initial memory slots. Specifically, the model outputs a module program $\{\mathbf{u}^{(k)}\}$ for several module networks (*e.g.* `CHOOSE`, `COUNT`, ...) via module selector, each of which produces an answer vector . The aggregated result of module network outputs determines the top-k answers.

2.1 Input Encoding

Query encoder: We represent each textual query with an attention-based Bi-LSTM language model (Conneau et al., 2017) with GloVe (Pennington et al., 2014) distributed word embeddings trained on the Wikipedia and the Gigaword corpus with a total of 6B tokens.

Memory encoder: We represent each memory node based on both its structural features (graph embeddings) and contextual multi-modal features from its neighboring nodes (*e.g.* attribute values).

Structural contexts of each memory node ($\mathbf{m}_s$) are encoded via graph embeddings projection approaches (Bordes et al., 2013), in which nodes with similar relation connectivity are mapped closer in the embeddings space. The model for obtaining embeddings from a MG (composed of subject-relation-object (s, r, o) triples) can be formulated as follows:

$$P(\mathbb{I}_r(s,o)=1|\theta) = \text{score}\big(\mathbf{e}(s), \mathbf{e}_r(r), \mathbf{e}(o)\big) \quad (1)$$

where $\mathbb{I}_r$ is an indicator function of a known relation r for two entities (s,o) (1: valid relation, 0: unknown relation), $\mathbf{e}$ is a function that extracts embeddings for entities, $\mathbf{e}_r$ extracts embeddings for relations, and score$(\cdot)$ is a function (*e.g.* multi-layer perceptrons) that produces a likelihood of a valid triple.

For contextual representation of memories ($\mathbf{m}_c$), we compute attention-weighted sum of textual representation of neighboring nodes and attributes (connected via $\mathbf{r}_j \in \mathbf{R}$), using the same

language model as the query encoder:

$$\mathbf{m}_c = \sum \gamma_j \mathbf{m}_{c,j}$$
$$\gamma = \sigma(\mathbf{W}_{q\gamma}\mathbf{q})$$

Note that the query attention vector γ attenuates or amplifies each attribute of memory based on a query vector to better account for query-memory compatibility accordingly. We then concatenate the structural features with semantic contextual features to obtain the final memory representation ($\mathbf{m} = [\mathbf{m}_s; \mathbf{m}_c]$).

2.2 Memory Graph Network

Inspired by the recently introduced graph traversal networks (Moon et al., 2019) which output discrete graph operations given input contexts, we formulate our MGN as follows. Given a set of initial memory slots ($\mathbf{m}$) and a query ($\mathbf{q}$), the MGN model outputs a sequence path of walk steps ($\mathbf{p}$) within MG to attend to relevant nodes or expand initial memory slots (Figure 3):

$$\{\mathbf{p}^{(k)}\} = \text{MGN}(\mathbf{q}, \{\mathbf{m}^{(k)}\}) \tag{2}$$

Specifically, we define the attention-based graph decoder model which prunes unattended paths, which effectively reduce the search space for memory expansion. We formulate the decoding steps for MGN as follows (bias terms for gates are omitted for simplicity of notation):

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{ci}\mathbf{c}_{t-1}) \\
\mathbf{c}_t &= (1 - \mathbf{i}_t) \odot \mathbf{c}_{t-1} \\
&\quad + \mathbf{i}_t \odot \tanh(\mathbf{W}_{zc}\mathbf{z}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1}) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{zo}\mathbf{z}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{co}\mathbf{c}_t) \\
\mathbf{h}_t &= \text{WALK}(\overline{\mathbf{x}}, \mathbf{z}_t) = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned} \tag{3}$$

where $\mathbf{z}_t$ is a context vector at decoding step t, produced from the attention over graph relations which is defined as follows:

$$\begin{aligned}
\overline{\mathbf{x}} &= \mathbf{W}_{qmx}[\mathbf{q}; \mathbf{m}^{(k)}] \\
\alpha_t &= \sigma(\mathbf{W}_{h\alpha}\mathbf{h}_{t-1} + \mathbf{W}_{x\alpha}\overline{\mathbf{x}}) \\
\mathbf{z}_t &= \mathbf{h}_{t-1} + \sum_{\mathbf{r}_j \in \mathbf{R}} \alpha_{t,j}\mathbf{r}_j
\end{aligned} \tag{4}$$

where $\alpha_t \in \mathbb{R}^{|\mathbf{R}|}$ is an attention vector over the relations space, $\mathbf{r}_k$ is relation embeddings, and $\mathbf{z}_t$ is a resulting node context vector after walking from its previous node on an attended path.

The graph decoder is trained with the ground-truth walk paths by computing the combined loss of $\mathcal{L}_{\text{walk}}(\mathbf{m}, \mathbf{q}, \mathbf{p}) = \sum_{i,t} \mathcal{L}_e + \mathcal{L}_n$ between predicted paths and each of $\{\mathbf{p}_e, \mathbf{p}_n\}$, respectively ($\mathcal{L}_e$: loss for edge paths, and $\mathcal{L}_n$ for node paths):

$$\begin{aligned}
&\sum_{\tilde{\mathbf{p}_e} \neq \mathbf{p}_{e,t}^{(i)}} \max[0, \tilde{\mathbf{p}_e} \cdot \mathbf{p}_{e,t}{}^{(i)} - \alpha_t \mathbf{r} \cdot (\mathbf{p}_{e,t}^{(i)} - \tilde{\mathbf{y}_e})^\top] \\
+ &\sum_{\tilde{\mathbf{p}_n} \neq \mathbf{p}_{n,t}^{(i)}} \max[0, \tilde{\mathbf{p}_n} \cdot \mathbf{p}_{n,t}{}^{(i)} - \mathbf{h}_t{}^{(i)} \cdot (\mathbf{p}_{n,t}^{(i)} - \tilde{\mathbf{p}_n})^\top]
\end{aligned}$$

At test time, we expand the memory slots by activating the nodes along the optimal paths based on the sum of their relevance scores (left) and soft-attention-based output path scores (right) at each decoding step:

$$\mathbf{p}_{n,t}^{(k)} = \underset{\mathbf{p}_n^{(k)} \in \mathbf{V}_{R,1}(\mathbf{p}_{n,t-1}^{(k)})}{\text{argmax}} \mathbf{h}_t \cdot \mathbf{p}_n^{(i)\top} + \sum \alpha_{t,j}\mathbf{r}_j \cdot \mathbf{p}_e^{(k)\top} \tag{5}$$

2.3 Module Networks

MGN outputs are then passed to module networks for the final stage of answer prediction. We extend the previous work in module networks (Kottur et al., 2018), often used in VQA tasks, to accommodate for graph nodes output via MGN. We first formulate the module selector which outputs the module label probability $\{\mathbf{u}^{(k)}\}$ given input contexts for each memory node, trained with cross-entropy loss $\mathcal{L}_{\text{module}}$:

$$\{\mathbf{u}^{(k)}\} = \text{Softmax}(\text{MLP}(\mathbf{q}, \{\mathbf{m}^{(k)}\})) \tag{6}$$

We then define the memory attention to attenuate or amplify all activated memory nodes based on their compatibility with query, formulated as follows:

$$\beta = \text{MLP}(\mathbf{q}, \{\mathbf{m}^{(k)}\}, \{\mathbf{p}^{(k)}\}) \tag{7}$$
$$\alpha = \text{Softmax}(\mathbf{W}_\beta^\top \beta) \in \mathbb{R}^K \tag{8}$$

For this work, we propose the following four modules: `CHOOSE`, `COUNT`, `CONFIRM`, `SET_OR`, and `SET_AND`, hence $\mathbf{u}^{(k)} \in \mathbb{R}^5$. Note that the formulation can be extended to the auto-regressive decoder in case sequential execution of modules is required.

`CHOOSE` module outputs answer space vector by assigning weighted sum scores to nodes along the MGN soft-attention walk paths. End nodes with the most probable walk paths thus get the

highest scores, and their node attribute values are considered as answer candidates. `COUNT` module counts the query-compatible among the activated nodes, $\mathbf{a} = \mathbf{W}_K^\top([\alpha; \max\{\alpha\}; \min\{\alpha\}])$. `CONFIRM` uses a similar approach to `COUNT`, except it outputs a binary label indicating whether the memory nodes match the query condition: $\mathbf{a} = \mathbf{W}_b^\top([\alpha; \max\{\alpha\}; \min\{\alpha\}])$. `SET` modules either combine or find intersection among answer candidates by updating the answer vectors with $\mathbf{a} = \max\{\mathbf{W}_s^\top\{\mathbf{a}^{(k)}\}\}$ or $\mathbf{a} = \min\{\mathbf{W}_s^\top\{\mathbf{a}^{(k)}\}\}$.

2.4 Answer Decoding

Answers from each module network (Section 2.3 are then aggregated as weighted sum of answer vectors with module probability (Eq.6), guided by memory attention (Eq.7). Predicted answers are evaluated with cross-entropy loss $\mathcal{L}_{\text{ans}}$

We observe that the model performs better when the MGN component of the model is pre-trained with ground-truth paths. We thus first train the MGN network with the same training split (without answer labels), and then train the entire model with module networks, fully end-to-end supervised with $\mathcal{L} = \mathcal{L}_{\text{walk}} + \mathcal{L}_{\text{module}} + \mathcal{L}_{\text{ans}}$.

3 Dataset

To empirically evaluate the proposed approach for the Episodic Memory QA task, we create a new dataset, *MemQA*, of 100K question and answer pairs composed based on synthetic memory graphs that are artificially generated. We specifically use the synthetically generated MG to avoid the need for inferring memory graphs from other structured data (such as publicly available photo albums such as Flicker, etc.) which are often limited in size and domains. We bootstrap our large-scale realistic memory graph dataset with the following procedures: first, we construct a synthetic social graph with a set number of artificial users, each with randomly generated interest embeddings. We then create a realistic memory graph by randomly choosing participants within the synthetic social graph as well as activities and associated entities from the curated list (of locations, events, public entities, etc.), which is a subset of common-fact Freebase knowledge base (Bast et al., 2014). Each generated memory node thus has connections to entities appearing in KGs, comprising the memory graph together. Finally, given a set of reference memory nodes and neighboring attributes, we pragmatically generate target ground-truth answer samples (*e.g.* single-hop / multi-hop node value, count, set comparison, yes/no, etc.) We then collect QA pairs for each sample with templates composed with human annotators, which are combined with 1K manual paraphrasing steps for added variety and confirmation. We randomly split the QA corpus into into train (70%), validation (15%), and test sets (15%).

Answer Type	%	**Examples**
Location	18	Mt. Rainier, AMC Theater
KG entities & events	17	Iron Man, Coachella
Common nouns, etc.	16	skiing, movie
Person / Group	16	Mark, Jon
Count	15	zero, three
Date / Time	10	01-02-2018, 7 PM
Yes/No	6	-
Miscellaneous	2	-

Table 1: Types of Answers in $MemQA$

4 Empirical Evaluation

Task: Given a query and a set of initial memory graph nodes via graph search, we evaluate the model on the open-ended question answering prediction task.

4.1 Baselines

We choose as baselines the following QA systems (see Section 5 for details), and modify them accordingly to make comparisons with our task:

- MemN2N (Sukhbaatar et al., 2016): uses the end-to-end memory networks with the static set of initial memory slots. Each memory slot is represented with a bag-of-symbols for surrounding attributes and nodes. We use a single Softmax layer for answer classification. Memory slot size is tuned as a hyper-parameter.

- MemexNet (Jiang et al., 2018): uses the textual representation for multi-modal attributes, and frames the problem into a classification problem via text kernel match approaches to predict approximate answers. Since the dataset does not contain images, we omit the CNN representations.

We also consider several configurations of our proposed approach to examine contributions of

Components	Model	Precision@k			
		k=1	3	5	10
A	`MemN2N` (Sukhbaatar et al., 2016)	29.7	43.8	51.0	50.5
A	`Memex` (Jiang et al., 2018)	32.6	50.3	58.2	59.1
G	`MemQANet` (ablation)	36.7	56.8	63.5	67.9
G + N	`MemQANet` (ablation)	41.1	65.8	75.5	80.0
G + N + E	`MemQANet` (**proposed**)	**45.8**	**68.1**	**75.7**	**80.9**

Table 2: QA performance on the *MemQA* dataset (metric: precision@k). Our proposed model is compared against the selected baseline models as well as several ablation variations of the proposed model (model components (G): Memory Graph Network, (N): Module Networks, (E): graph embeddings).

each component (model components (G): Memory Graph Network, (N): Module Networks, (E): structural graph embeddings).

- (**Proposed**; G+N+E): is the proposed approach as described in Figure 2.
- (G+N): does not structural graph embeddings for MGN, and relies on semantic representation of surrounding nodes.
- (G): does not use any of the module networks (*e.g.* `COUNT`, ...), and predicts MGN graph output as answers instead.

4.2 Results

Parameters: We tune the parameters of each model with the following search space (bold indicate the choice for our final model): graph embeddings size: {64, **128**, 256, 512}, Bi-LSTM hidden states for the language model: {64, **128**, 256, 512}, MGN hidden states: {64, **128**, 256, 512}, word embeddings size: {100, 200, **300**}, and max memory slots: {1, 5, **10**, 20, 40, 80}. We optimize the parameters with Adagrad (Duchi et al., 2011) with batch size 10, learning rate 0.01, epsilon 10^{-8}, and decay 0.1.

Main Results: Table 2 shows the results of the top-k predictions of the proposed model and the baselines. It can be seen that the proposed Memory QA model outperforms other QA baselines for precision at all ks.

Specifically, with the MGN walker model, the MemQA model learns to condition its walk path on query contexts and attend and expand memory nodes, thus outperforming the baseline models that simply rely on their initial memory slots, typically large in size to maintain reasonable recall. The node expansion via MGN allows the model to keep the initial memory slots small (10) and expand only when necessary (*e.g.* for queries that require references to related memories, *"... where did I go after ..."*), thus improving the precision performance. Note that memory slot sizes for baselines are tuned for their performance on the validation set.

In addition, it can be seen that the neural module components (G+N and G+N+E) greatly outperform the ablation model (G) and the baselines by aggregating answers with the modules specifically designed for various types of questions. These neural modules allow the model to answer questions that are typically hard to answer (*e.g.* count, set comparison, etc.) by explicitly reducing the answer space accordingly.

Note also that the graph embeddings (G+N+E) improve the performance over the ablation model that does not use structural contexts (G+N), indicating that the model learns to better leverage knowledge graph contexts to answer questions.

Error Analysis: Table 3 shows some of the example output from the proposed model, given the input question and memory graph nodes. It can be seen that the model is able to predict answers by combining answer contexts from multiple components (walk path, node attention, neural modules, etc.) In general, the MGN walker successfully explores the respective single-hop or multi-hop relations within the memory graph, while keeping the initial memory slots small enough. The activated nodes via graph traversals are then used as input for each neural module, the aggregated results of which are the final top-k answer predictions. There are some cases where the final answer prediction is incorrect, whereas its walk path is correctly predicted. This is due to inaccurate prediction of memory attention vector given a query and initial memory slots, which requires compre-

Question and Answer	Model Preidction		
	Walk Path	Neural Module	Top-k Answers
Q: *Where did Jon and I go after we watched* Avengers? // **A**: *Symphony Hall*	$m_1 \rightarrow$ (NEXT_ID) $\rightarrow m'_1$ $\rightarrow$ (LOCATION)	CHOOSE	Symphony Hall, AMC Theatre, ...
Q: *Who did I go skiing with last year?* **A**: {*Emma, Jacob*}	$m_1 \rightarrow$ (PARTICIPANT) $m_3 \rightarrow$ (PARTICIPANT)	SET_OR	{Emma, Jacob}, {Emma, Noah}, ...
Q: *How many sci-fi movies have I watched* last year? // **A**: *Two*	$m_1 \rightarrow$ (ENTITY) $\rightarrow e'_1$ $\rightarrow$ (GENRE) $\rightarrow a'_1$	COUNT	Three, Two, Four, ...

Table 3: **Error Analysis**: Model predictions of walk paths (with expanded memory nodes noted with ') are partially shown for each question and ground-truth answer pair. The full reference memory graphs are not shown here due to space constraints. Initial memory slot size = 10.

hensive understanding of surrounding knowledge nodes in the context of the query.

5 Related Work

Memory Networks: Weston et al. (2014); Sukhbaatar et al. (2016) propose Memory Networks with explicit memory slots to contain auxiliary multi-input, now widely used in many QA and MRC tasks for its transitive reasoning capability. Traditional limitations are that memory slots for storing answer candidates are fixed in size, and naively increasing the slot size typically decreases the precision. Several work extend this line of research, for example by allowing for dynamic update of memory slots given streams of input (Kumar et al., 2016; Tran et al., 2016; Xu et al., 2019), reinforcement learning based retention control (Jung et al., 2018), etc. By allowing for storing graph nodes as memory slots and for slot expansion via graph traversals, our proposed Memory Graph Networks (MGN) effectively bypass the issues.

Structured QA systems: often answer questions based on large-scale common fact knowledge graphs (Bordes et al., 2015; tau Yih et al., 2015; Xu et al., 2016; Jain, 2016; Yin et al., 2016; Dubey et al., 2018), typically via an entity linking system and a QA model for predicting graph operations through template matching approaches, etc. Our approach is inspired by this line of work, and we utilize the proposed module networks and the MGN walker model to address unique challenges to Episodic Memory QA.

Machine Reading Comprehension (MRC) systems: aim at predicting answers given evidence documents, typically in length of a few paragraphs (Seo et al., 2017; Rajpurkar et al., 2016, 2018; Cao et al., 2019; tau Yih et al., 2015). Several recent work address multi-hop reasoning within multiple documents (Yang et al., 2018; Welbl et al., 2018; Bauer et al., 2018; Clark et al., 2018) or conversational settings (Choi et al., 2018; Reddy et al., 2018), which require often complex reasoning tools. Unlike in MRC systems that typically rely on language understanding, we effectively utilize structural properties of memory graph to traverse and highlight specific attributes or nodes that are required to answer questions.

Visual QA systems: aim to answer questions based on contexts from images (Antol et al., 2015; Wang et al., 2018; Wu et al., 2018). Recently, neural modules (Kottur et al., 2018) are proposed to address specific challenges to VQA such as visual co-reference resolutions, etc. Our work extends the idea of neural modules for Episodic Memory QA by implementing modules that can take graph paths as input for answer decoding. Jiang et al. (2018) proposes visual memex QA which tackles similar problem domains given a dataset collected around photo albums. Instead of relying on meta information and multi-modal content of a photo album, our work explicitly utilizes semantic and structural contexts from memory and knowledge graphs. Another recent line of work for VQA includes graph based visual learning (Hudson and Manning, 2019), which aims to represent each image with a sub-graph of visual contexts. While graph-based VQA operates on a graph constructed from a single scene, Episodic Memory QA operates on a large-scale memory graph with knowledge nodes. We therefore propose memory graph networks to handle ambiguous candidate nodes, a main contribution of the proposed work.

6 Conclusions

We introduce Episodic Memory QA, the task of answering personal user questions grounded on memory graph (MG). The dataset is generated with synthetic memory graphs with simulated attributes, and accompanied with 100K QA pairs composed via bootstrapped scripts and manual annotations. Several novel model components are proposed for unique challenges for the Episodic Memory QA: 1) Memory Graph Networks (MGN) extends the conventional memory networks by enabling dynamic expansion of memory slots through graph traversals, which also naturally allows for explainable predictions. 2) Several neural module networks are proposed for the proposed task, each of which takes queries and memory graphs as input to infer answers. 3) The main Episodic Memory QA Net aggregates answer prediction from each neural module to generate final answer candidates. The empirical results demonstrate the efficacy of the proposed model in the Memory QA reasoning.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *ICCV*.

Hannah Bast, Florian Baurle, Bjorn Buchhold, and Elmar Haussmann. 2014. Easy access to the freebase dataset. In *WWW*.

Lisa Bauer, Yicheng Wang, and Mohit Bansal. 2018. Commonsense for generative multi-hop question answering tasks. *EMNLP*.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory network. *arxiv*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2019. Question answering by reasoning across documents with graph convolutional networks. *NAACL*.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. Quac: Question answering in context. *EMNLP*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

Mohnish Dubey, Debayan Banerjee, Debanjan Chaudhuri, and Jens Lehmann. 2018. Earl: Joint entity and relation linking for question answering over knowledge graphs. *ESWC*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*.

Drew A Hudson and Christopher D Manning. 2019. Gqa: A new dataset for real-world visual reasoning and compositional question answering. *CVPR*.

Sarthak Jain. 2016. Question answering over knowledge base using factual memory networks. *NAACL*.

Lu Jiang, Junwei Liang, Liangliang Cao, Yannis Kalantidis, Sachin Farfade, and Alexander Hauptmann. 2018. Memexqa: Visual memex question answering. *arXiv*.

Hyunwoo Jung, Moonsu Han, Minki Kang, and Sungju Hwang. 2018. Learning what to remember: Long-term episodic memory networks for learning from streaming data. *arXiv preprint arXiv:1812.04227*.

Satwik Kottur, José MF Moura, Devi Parikh, Dhruv Batra, and Marcus Rohrbach. 2018. Visual coreference resolution in visual dialog using neural module networks. In *ECCV*.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *ICML*.

Seungwhan Moon, Pararth Shah, Anuj Kumar, and Rajen Subba. 2019. Opendialkg: Explainable conversational reasoning with attention-based walks over knowledge graphs. *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *ACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv:1606.05250*.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2018. Coqa: A conversational question answering challenge. *arXiv preprint arXiv:1808.07042*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. *ICLR*.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2016. End-to-end memory networks. *NIPS*.

Ke Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory networks for language modeling. *NAACL*.

Peng Wang, Qi Wu, Chunhua Shen, Anthony Dick, and Anton van den Hengel. 2018. Fvqa: Fact-based visual question answering. *PAMI*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *TACL*.

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

Qi Wu, Chunhua Shen, Peng Wang, Anthony Dick, and Anton van den Hengel. 2018. Image captioning and visual question answering based on attributes and external knowledge. *PAMI*.

Kun Xu, Yuxuan Lai, Yansong Feng, and Zhiguo Wang. 2019. Enhancing key-value memory neural networks for knowledge based question answering. *NAACL*.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on freebase via relation extraction and textual evidence. *ACL*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *EMNLP*.

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

Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. 2016. Simple question answering by attentive convolutional neural network. *COLING*.