Reasoning Over Virtual Knowledge Bases With Open Predicate Relations

Haitian Sun * [1] **Pat Verga** [2] **Bhuwan Dhingra** [2] **Ruslan Salakhutdinov** [1] **William W. Cohen** [2]

Abstract

We present the Open Predicate Query Language (OPQL); a method for constructing a virtual KB (VKB) trained entirely from text. Large Knowledge Bases (KBs) are indispensable for a wide-range of industry applications such as question answering and recommendation. Typically, KBs encode world knowledge in a structured, readily accessible form derived from laborious human annotation efforts. Unfortunately, while they are extremely high precision, KBs are inevitably highly incomplete and automated methods for enriching them are far too inaccurate. Instead, OPQL constructs a VKB by encoding and indexing a set of relation mentions in a way that naturally enables reasoning and can be trained without any structured supervision. We demonstrate that OPQL outperforms prior VKB methods on two different KB reasoning tasks and, additionally, can be used as an external memory integrated into a language model (OPQL-LM) leading to improvements on two open-domain question answering tasks.

1. Introduction

Large knowledge bases (KBs) structure information around triples of entities and relation types that describe the relationships between the subject and object entities, for example, [*Charles Darwin*, author of, *On the Origin of Species*]. While KBs have been a key component of artificial intelligence since the field's inception (Newell & Simon, 1956; Newell et al., 1959) broad-coverage KBs are inevitably incomplete (Min et al., 2013), despite efforts to automate their creation through text extraction (Angeli et al., 2015; Mitchell et al., 2015).

[1]Carnegie Mellon University [2]Google Research. Correspondence to: Haitian Sun (* Most work done at Google Research) <haitians@cs.cmu.edu>, Pat Verga <patverga@google.com>, Bhuwan Dhingra <bdhingra@google.com>, Ruslan Salakhutdinov <rsalakhu@cs.cmu.edu>, William W. Cohen <wcohen@google.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

An alternative to extracting information to augment existing KBs relies on directly answering queries using text corpora (Chen et al., 2017). Recently, Dhingra et al. (2020) proposed DrKIT which answers questions based on a *virtual KB* (VKB) constructed automatically from a text corpus. The key idea behind DrKIT is to greatly simplify the construction of a KB by building a "soft KB", closely related to the original text, and compensate for the lack of structure in the VKB by employing more sophisticated neural methods to answer questions. In DrKIT, each element of the VKB is composed of an entity mention and its surrounding context, encoded into a dense embedding representation. Given a query, a learned encoder projects that query into the same embedding space of the VKB mentions. The query vector is scored amongst all embedded mentions resulting in the retrieval of relevant mentions, analogous to the retrieval of a triple from a standard structured KB. Other neural operations can be used to differentiably reason over the VKB, for instance by answering "multihop" questions that combine information from multiple embedded mentions.

A weakness of DrKIT is that constructing the VKB relied on distant supervision using existing KB triples. This leads to two limitations: (1) it is unclear if the VKB will correctly encode relations not present in the original structured KB used for distant supervision, and (2) the approach is inapplicable in domains without existing structured KBs (such as many technical areas.) To address these problems, we introduce (1) a novel VKB construction method which can be trained without any distant KB supervision and (2) a set of differentiable reasoning operations on the VKB, which we call the Open Predicate Query Language (OPQL). Hence, unlike DrKIT's procedure, an OPQL virtual KB can be created from any entity-linked corpus.

The key idea in constructing the OPQL VKB is to use a dual-encoder pre-training process. Our process is similar to that used in (Baldini Soares et al., 2019), which was shown to be useful for tasks such as relation classification. Here we show that this pre-training process can also construct a VKB that effectively supports more complex reasoning operations, and also that OPQL operations can be tightly integrated with a neural language model (LM).

To summarize this paper's contributions: (1) we describe an effective VKB pre-training method that, unlike previous

methods, does not require an initial structured KB for distant supervision, and (2) demonstrate that this VKB can be used to answer multi-hop semi-structured queries effectively—in fact (3) OPQL outperforms previous state-of-the-art VKB methods significantly, despite needing less supervision. We also (4) demonstrate that OPQL can be injected as an external memory into a neural LM to obtain a new state-of-the-art on a widely-studied QA task (Yih et al., 2015). Finally we (5) extend our VKB-injected LM with the ability to combine multiple OPQL operations, and demonstrate this leads to a new state-of-the-art on a QA task requiring compositional and conjunctive reasoning (Talmor & Berant, 2018), even outperforming much larger text-to-text models.

2. Model

In this work, we propose OPQL, a method for building a VKB from text without any structured supervision. OPQL naturally supports compositional reasoning and can serve as the knowledge source for a memory augmented LM. Entries in the OPQL memory encode the relationship between pairs of entities, described in natural language. For example, a sentence "*Charles Darwin* published his book *On the Origin of Species* in 1859" describes the authorship of entity *On the Origin of Species*.

Importantly, these unstructured relationships expressed in text can be extremely fine-grained, covering semantics that would never be included in a pre-defined KB schema. This is made possible because OPQL has the ability of learning without any structured supervision. These relationships are organized into a key-value memory index (§2.4). Each key is the composition of a topic entity (e.g. *On the Origin of Species*) and an associated latent relationship expressed in text, constructed using pretrained entity embeddings (§2.3 and a pretrained text encoder (§2.2). Its value is the corresponding target entity, in this case, *Charles Darwin*. When the memory is queried, the returned value can be used directly to answer the input query (§3) or it can be integrated into an LM for further reasoning before producing a final prediction (§4).

2.1. Background

Input The input to the model used for pretraining OPQL is a sequence of n tokens $C = [c_0, c_1 ... c_n]$ for an arbitrary text span, e.g. "*Charles Darwin* published his book *On the Origin of Species* in 1859", that contains a set of demarcated entity mentions M. A mention "*Charles Darwin*" $\in M$ is denoted as $m = (c_i, c_j, e_m) \in M$, where i and j denote the first and last token of the mention span and the mention is linked to entity e_m in a predefined entity vocabulary $\mathcal{E}$, e.g. Charles_Darwin (Q1035) in Wikidata.

Enity Pairs OPQL learns to encode the relationship between a pair of entities (e_1, e_2), where e_1 is referred as topic entity and e_2 as target entity. In the example above, *On the Origin of Species* is the topic entity e_1 and *Charles Darwin* is the target entity $e2$. Potentially[1] each distinct entity pair in a sentence can be encoded. The relationship between a pair of entities is directional, so the pairs (e_1, e_2) and (e_2, e_1) are encoded differently.

Entity Embeddings A global entity embedding table $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times d_e}$ is pretrained using the RELIC strategy (Ling et al., 2020); here $\mathcal{E}$ is the entity vocabulary and d_e is the embedding dimensionality. Similar to pretraining a masked language model (MLM), an entity mention is randomly masked from the context and the goal is to retrieve the masked entity from the entity vocabulary $\mathcal{E}$.

Key-Value Memory We structure OPQL as a key-value memory. A key-value memory ($\mathbf{K}$, $\mathbf{V}$) is a general way of storing and retrieving knowledge (Miller et al., 2016). When queried, key embeddings $\mathbf{k}_i \in \mathbf{K}$ are matched with the query vector, and the corresponded value embedding $\mathbf{v}_i$ is returned. Each entry in OPQL encodes a pair of entities (e_1, e_2), along with a piece of text C that describes the relationship between e_1 and e_2. The key-value memory is constructed from the pretrained entity embedding table $\mathbf{E}$ and relation embeddings precomputed from a pretrained relation encoder. A key memory holds information from the topic entity and the relationship, and a value memory holds the target entity. We will discuss the pretrained entity and relation embeddings first and then discuss how to construct the key-value memory using the pretrained embeddings.

2.2. Relation Encoder

Preprocessing Text for the Relation Encoder OPQL encodes the relationship between a pair of entities (e_1, e_2) in a natural language sentence C where both e_1 and e_2 are mentioned. Intuitively C describes the relationship between e_1 and e_2 in the context, and we train a relation encoder to represent the relationship as a vector.

To indicate the location of the topic and target entities e_1 and e_2 we introduce two special tokens [R1] and [R2] that are inserted directly after the mentions of the topic and target entities in the sentence (e.g., C_r becomes "*Charles Darwin* [R2] published his book *On the Origin of Species* [R1] in 1859"). The contextual encodings of [R1] and [R2] will be used to compute relation embeddings. We also introduce another special token [ENT] to mask the topic and target entity mentions; thus sentence C_r finally becomes "[ENT] [R2] published his book [ENT] [R1] in 1859". Masking the mentions of entities prevents the relation encoder from memorizing the surface forms of the entities, and helps it generalize to similar relations involving other entities. The contextual embeddings at [ENT] tokens are not used by the

[1] Heuristics for limiting the number of entity pairs derived from text are discussed in Section 2.5.

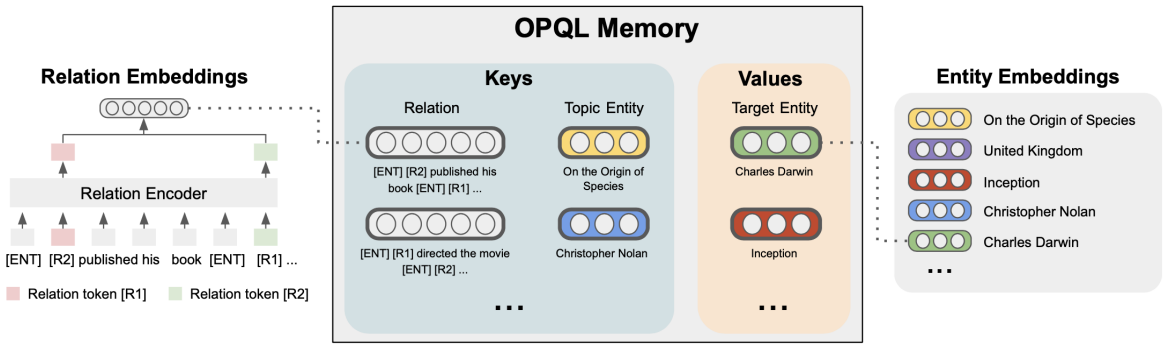

Figure 1. **OPQL memory structure.** The OPQL memory is a key-value memory. Keys are computed from embeddings of the topic entity, e.g. *On the Origin of Species*, looked up from the entity embedding table, and relation embeddings from the pretrained relation encoder. Values are embeddings of target entities. The memory is constructed from any entity-linked text corpus, e.g. Wikipedia.

relation encoder, but will be used for (masked) entity linking in §4.

Computing Relation Embedding Given this preprocessing of an entity-mention pair (e_1, e_2), the relation embedding $\mathbf{r}_{e_1,e_2}$ is defined as follows. Let s and t be the location of tokens [R1] and [R2] in the masked sentence C_r. We construct a projection of the concatenation of the contextual embeddings $\mathbf{h}_s$ and $\mathbf{h}_t$ at the locations s and t, as follows:

$$\mathbf{r}_{e_1,e_2} = \mathbf{W}_r^T \, [\mathbf{h}_s; \mathbf{h}_t] \tag{1}$$

Training the Relation Encoder We train our relation encoder following Baldini Soares et al. (2019), who train embeddings such that relation mentions containing the same entity pairs are more similar to each other than relation mentions that contain different entity pairs.

Specifically, mini-batches are constructed which contain at least two documents that contain entity pair (e_1, e_2), as well as negative documents containing different relation pairs (see Section 2.5 for details.) Use $\mathbf{r}_{e_1,e_2}$ to denote the embedding from input C that of (e_1, e_2), use $\mathbf{r}_{e_i,e_j}$ for embeddings from documents $C'_0, \ldots, C'_n$ of other pairs (e_i, e_j)'s, and let $\mathbb{I}_{e_i=e_1,e_j=e_2}$ be an equality indicator for entity pairs in the minibatch. We maximize the inner product between relation embeddings iff the same pair (e_1, e_2) is mentioned in the candidates:

$$L_{\text{rel}} = \text{cross\_ent}(\text{softmax}(\mathbf{r}_{e_1,e_2}{}^T \mathbf{r}_{e_i,e_j}), \mathbb{I}_{e_i=e_1,e_j=e_2}) \tag{2}$$

2.3. Entity Linking

Additionally, we use a multi-task training objective to learn representations of individual entities by learning to link other mentions in the context—i.e., mentions other than the topic and target entities—to the correct entity. For example, in the masked sentence C_r, "[ENT] [R2], an *English* naturalist, published his book [ENT] [R1] in 1859", the mention *English* is not part of the pair (*On the Origin of Species*, *Charles Darwin*) and thus not masked with [ENT], but should be linked to an entity for "England". These *context entity* mentions are represented as $m_{e_a} = (c_i, c_j, e_a)$ where c_i and c_j are the start and end positions, and e_a is the entity to which this mention should be linked. We construct an embedding of mentions m_{e_a} from the contextual embedding $\mathbf{h}_i$ at the start position c_i for the mention, i.e. $\mathbf{m}_{e_a} = \mathbf{W}_e^T \mathbf{h}_i$. This is used to retrieve the most similar entity from the embedding table $\mathbf{E}$, scored with inner product distance, using this loss:

$$L_{\text{el}} = \text{cross\_ent}(\text{softmax}(\mathbf{m}_{e_a}{}^T \mathbf{e}_i), \mathbb{I}_{e_i=e_a}) \tag{3}$$

2.4. Storing OPQL's VKB as a key-value memory

OPQL stores relationships between pairs of entities in a key-value memory ($\mathbf{K}$, $\mathbf{V}$). Given an input C that mentions a pair of entities e_1 and e_2, the key embedding $\mathbf{k}_{e_1,e_2}$ for the pair (e_1, e_2) is constructed compositionally using the embeddings of topic entities $\mathbf{e}_1$ from the entity embedding table $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times d_e}$ and the pretrained relation embedding $\mathbf{r}_{e_1,e_2}$. $\mathbf{W}_k$ is a linear projection matrix that will be learned in the finetuning tasks. The value $\mathbf{v}_{e_1,e_2}$ is the embedding of the target entity $\mathbf{e}_2$.

$$\mathbf{k}_{e_1,e_2} = \mathbf{W}_k^T \, [\mathbf{e}_1; \mathbf{r}_{e_1,e_2}] \in \mathbf{K}, \quad \mathbf{v}_{e_1,e_2} = \mathbf{e}_2 \in \mathbf{V}$$

We iterate through all sentences in the Wikipedia corpus that contain two or more entities to construct the OPQL memory. Note that each entity pair can be mentioned one or multiple times in the corpus. The memory could keep all mentions of an entity pair (e_1, e_2), or instead reduce multiple mentions of the same entity pair to a single entry, for example averaging the key embedding of the individual mentions. This choice can be made based on application, or computational constraints. In the relational following task (§3), we keep all mentions of entity pairs in the OPQL memory. For the open-domain QA task (§4), the text corpus is larger,[2] and keeping all pairs of entities in the memory

[2]We use the entire Wikipedia as our text corpus for the open-

is not feasible. We randomly select up to 5 mentions for each entity pair and average their key embedding as the final entry in the OPQL memory. (Note the the value embeddings of an entity pair are the same across multiple mentions).

2.5. Pretraining Data

Entity Linking Data We use Wikipedia passages with hyperlinks as our pretraining. The hyperlinks link a span of tokens to a Wikipedia page for an entity e. The spans of tokens that are linked are considered as mentions m, and entity e is the corresponding entity. We take the top 1M entities that are most frequently mentioned in Wikipedia as our entity vocabulary $\mathcal{E}$. Passages are split into text pieces of 128 tokens. Entity linking is trained on the mentions of entities in the context, excluding the ones that are treated as topic and target entities. The pretraining corpus contains 93.5M mentions of the top 1M entities in the vocabulary.

Relation Encoder Data We first construct the vocabulary of entity pairs (e_1, e_2) from the Wikipedia corpus. We count the pairs of entities that co-occur in the same text piece, and discard the that appears less than 5 times. The pairs are sorted by their point-wise mutual information (PMI). The top 800k pairs of entities are selected as our candidates for training. During training, an input with pair (e_1, e_2) will be paired with 2 positive examples that mentions the same pair of entities, and 8 hard negative examples (e_i, e_j) that mention either the same topic or target entity, i.e. $e_i = e_1$ or $e_j = e_2$, but not both. Batch negatives are also included in pretraining. 30.6M training data on the top 800k entity pairs are constructed from the Wikipedia corpus.

Pretraining Loss The relation encoder is pretrained with entity linking loss L_{el} in Eq. 3 and relation encoder loss L_{rel} in Eq. 2, i.e. $L = L_{\text{el}} + L_{\text{rel}}$.

3. Relation Following with OPQL

In the previous section we introduced the method for pretraining the representations in the VKB. Here we describe an application of the VKB to answering multi-hop questions.

Background A relation following operation $Y = X.\text{follow}(R)$ maps a set of entities X and a set of relations R to a set of entities Y such that $Y = \{y \mid \exists x \in X, r \in R : r(x, y)\}$. One common application of the relation following operation is to solve QA tasks (Cohen et al., 2020; Sun et al., 2020). For example, a question "Who is the author of *On the Origin of Species*?" can be answered by computing $X.\text{follow}(R)$ with $X = \{\textit{On the Origin of Species}\}$ and $R = \{\textit{author\_of}\}$. In a learning task, R is a weighted set of relations, and the model learns to predict the relation weights from the question.[3] The resulting relation following query is then executed with a key-value lookup into the OPQL memory.

Preprocessing Data for Fine-Tuning We finetune the pretrained relation encoder to compute the queries for the relation following task as follows. Let Q be a question, which we will assume contains one known topic entity e_1. We let $X = \{e_1\}$ contain that entity. The model must predict a weighted set of target entities Y that answer the question. Answers are always appended to the question, e.g. "Who is the author of *On the Origin of Species*? *Charles Darwin*". To encode the relation embeddings $\mathbf{r}_{X,Y}$ for the preprocessed question/answer pair, [R1] and [R2] are inserted after the mentions of the topic entity X and target (answer) entity Y, and both mentions are masked with [ENT]. In the example above, the masked question thus becomes "Who is the author of [ENT] [R1]? [ENT] [R2]". This transformation ensures that the masked question has similar form to the input of the relation encoder, so the pretrained relation encoder can be easily finetuned for relation following tasks.

Relation Following for OPQL Recall that a follow query is executed with a key-value lookup from the OPQL memory. Specifically, a query vector $\mathbf{q}_{X,Y}$ is composed using the embedding of the topic entity X and the relation embedding $\mathbf{r}_{X,Y}$ of the question:

$$\mathbf{e}_X = \sum_i \alpha_i \cdot \mathbf{e}_{x_i}, \quad x_i \in X \tag{4}$$

$$\mathbf{q}_{X,Y} = \mathbf{W}_q^T [\mathbf{e}_X; \mathbf{W}_t^T \mathbf{r}_{X,Y}] \tag{5}$$

where in general $\mathbf{e}_X$ is the weighted average of embeddings of entities $e_{x_i} \in X$ (this general case applies for multi-hop questions), α_i is the weight of x_i in X, and $\mathbf{W}_t$ is a learned projection matrix.

The query $\mathbf{q}_{X,Y}$ is then used to retrieve against the key memory $\mathbf{K}$, returning the top k entries with the largest inner product scores (k_{e_i,e_j}, e_j).[4] The T_k set of top-k retrieved values $\{e_j\}$ are the answers Y. The weights β_{e_i,e_j} of entity e_j is the softmax of scores of the top k retrieval result, denoted as $\text{top}_k(\mathbf{q}_{X,Y}, \mathbf{K})$.

$$\beta_{e_i,e_j} = \begin{cases} \text{softmax}(\mathbf{q}_{X,Y}^T \mathbf{k}_{e_i,e_j} \text{ for } (e_i, e_j) \in T_k) \\ 0, \quad \text{else} \end{cases}$$

To improve retrieval accuracy, we also apply a sparse filter on the retrieval result to ensure the topic entity of the retrieved pair (e_j, e_k) is in X (Seo et al., 2019; Dhingra et al., 2020). To train the parameters of the relation following operation we optimize the retrieved set of answers against the ground truth using the loss

$$L_{\text{follow}} = \text{cross\_ent}(\beta_{e_i,e_j}, \mathbb{I}_{e_j \in \text{Ans}})$$

domain QA tasks.

[3] The set of entities X is the topic entities in the question, which in the experiments below is either provided, or easily obtained.

[4] The value v_{e_i,e_j} of the entry is always e_j, so we write it as (k_{e_i,e_j}, e_j) for short.

Multi-hop Relational Following Relation following operations can be chained to answer multi-hop questions: e.g., two-hop question can be decomposed into $Y = X.\text{follow}(R_1).\text{follow}(R_2)$. In this case, the predicted set of intermediate entities from the previous hop $X^{(t)} = X^{(t-1)}.\text{follow}(R_{t-1})$ becomes the input to the next hop, i.e. $X^{(t+1)} = X^{(t)}.\text{follow}(R_t)$. In our model we use a single relation embedding $\mathbf{r}_{X,Y}$ for the question but learn a different projection matrix $\mathbf{W}_t^{(t+1)}$ for each hop: e.g. we use

$$\mathbf{q}_{X,Y}^{(2)} = \mathbf{W}_q^T[\mathbf{e}_{X^{(1)}}; {\mathbf{W}_t^{(2)}}^T \mathbf{r}_{X,Y}]$$

to form the query $\mathbf{q}_{X,Y}^{(2)}$ for the second hop. In the multi-hop setting, the loss is only computed at the last step. The number of hops is a hyper-parameter and in our experiments it is given for each dataset.

Finetuning Details For the relational following task, the relation embeddings of entity pairs in the memory is precomputed and fixed at finetuning time. The pretrained relation encoder is finetuned to compute $\mathbf{r}_{X,Y}$ in Eq.5 for the query vector $\mathbf{q}_{X,Y}$. We also trained the projection matrices $\mathbf{W}_q$ and $\mathbf{W}_t^{(\cdot)}$, but fixed the entity embedding table $\mathbf{E}$. The finetuning job is only trained with the loss L_{follow}, since entity and relation embeddings are fixed in this task.

3.1. Experiments: Reasoning Over VKB

3.1.1. Datasets

MetaQA (Zhang et al., 2018) is a multi-hop QA dataset that extends the WikiMovies dataset to 2-hop and 3-hop questions in the movie domain. The questions are generated from templates, e.g. "Who starred in the movies directed by Christopher Nolan". Questions in MetaQA are answerable by the corpus in the original WikiMovies dataset. The corpus contains 18k passages that describes 7 different relations between 43k entities. We extracted 106k entity pairs from the corpus whose topic entity is the Wikipedia page title and target entity is one of the mentions in the passage. Reverse pairs are included, for a total of 213k entity pairs.

Multi-hop Slot Filling (MSF) (Dhingra et al., 2020) presents a large scale multi-hop reasoning dataset constructed from WikiData that contains 120k passages, 888 relations, and more than 200k entities. Queries are constructed from multi-hop paths in WikiData and turned into natural language questions by concatenating the head entity with a series of relations, e.g. ("Steve Jobs, founder, headquarter in, ?"). Similar to MetaQA, we constructed entity pairs by taking the Wikipedia page title as the topic entity and the entities mentioned in the passages as target entities. We extracted 1.3m and 781k entity pairs for 2-hop and 3-hop questions respectively.

Model	MetaQA		MSF	
	2Hop	3Hop	2Hop	3Hop
KVMem	7.0	19.5	3.4	2.6
DrQA	32.5	19.7	14.1	7.0
GRAFT-Net	36.2	40.2	-	-
PullNet	81.0	78.2	-	-
PIQA	-	-	36.9	18.2
DrKIT	86.0	**87.6**	46.9	24.4
OPQL-pretrained	84.7	84.3	48.5	28.1
OPQL	**88.5**	87.1	**49.2**	**29.7**

Table 1. Hits@1 results on multi-hop relational following task.

3.1.2. Baselines

GRAFT-Net (Sun et al., 2018) is a GCN based model that can perform reasoning jointly over text and knowledge bases. PullNet (Sun et al., 2019) extends GRAFT-NET by introducing an iterative retrieve-and-classify mechanism to solve multi-hop questions. PIQA (Seo et al., 2018) and DrKIT (Dhingra et al., 2020) build mention-level indexes using a pretrained encoder. Training the DrKIT index requires distant supervision using artificial queries constructed from KB. KV-Mem (Miller et al., 2016) and DrQA (Chen et al., 2017) are another two widely-used open-domain QA baselines.

3.1.3. Results

We experiment on the relational following task on both a pretrained memory (OPQL-pretrained) and a finetuned memory (OPQL). The Hits@1 results of the model are listed in Table 1. The OPQL-pretrained memory directly applies the pretrained relation encoder on entity pairs extracted from the dataset corpus, without any finetuning of the relation encoder. OPQL-pretrained achieves the state-of-the-art performance on both MSF 2-hop and 3-hop datasets, though OPQL-pretrained is slightly lower than DrKIT on the MetaQA. This is because DrKIT finetuned its mention encoder on MetaQA corpus. The MetaQA corpus only contains 14 relations (including their inverse) in the movie domain, so it's easy for the model to learn only capturing these relationship between entities. To mitigate this bias, we finetune the OPQL relation encoder on MetaQA corpus. The finetuning data is distantly constructed from 1-hop questions in the MetaQA dataset, by masking the topic entity from the question and inserting a placeholder for the target entity at the end of the question. We end up with 10K finetuning data for MetaQA and 19K for MSF. OPQL with finetuned memory outperforms DrKIT on 2-hop questions by 2.5 points and is very comparable on 3-hop questions.

3.1.4. Generalization to Novel Relations

The previous state-of-the-art model, DrKIT, uses training data in its pretraining procedure that is distantly constructed from KB. This restricts the capacity of DrKIT to only en-

code KB relations observed at pretraining time. However, pretraining OPQL does not require any signal from knowledge bases, so it can also encode relations that are out of the relation vocabulary in KB. To demonstrate this difference, we hold out some portions of relations in the pretraining phase of DrKIT but evaluate on queries where at least one of the held-out relations is required to answer the queries. This simulates a scenario where KBs are incomplete and limited in their relation vocabularies. We compare DrKIT to the pretrained OPQL memory. The result is shown in Figure 2. The performance of DrKIT drops significantly when evaluated on queries with novel relations not seen at pretraining time, while the performance of OPQL is consistent.

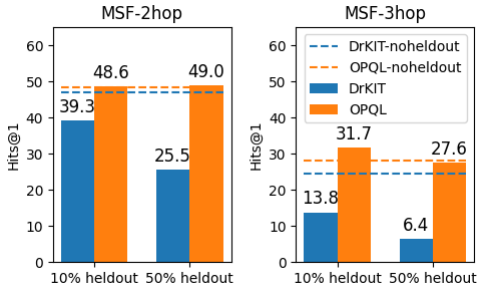

Figure 2. Hits@1 on multi-hop queries containing at least one novel relation. The dashed lines represent the accuracy on the full dataset from Table 1.

4. Augmenting OPQL with a LM

Recent advances in LM architectures have incorporated large external memories which can lead to better performance given fewer activated parameters (retrieved memories are only sparsely activated, rather than dense models which utilize nearly every parameter on every input). Verga et al. (2020) proposed Facts-as-Experts (FaE) which injected an external fact memory into an LM, increasing its performance on open-domain QA. However, this approach relied on a structured KB, suffering from many of the shortcomings discussed in previous sections such as limited coverage and applicability. In this section, we address these issues by replacing the KB-based fact memory of FaE with our OPQL memory.

Background A memory injected language model, e.g. FaE (Verga et al., 2020), learns to retrieve relevant entries from an external memory and mix the retrieved information into the language model to make its final predictions (see Figure 3). OPQL follows the same implementation as FaE but utilizes the pre-computed OPQL memory as the external memory. The relation embedding in the OPQL memory is both more diverse and fixed during training, leading to retrieval from the OPQL memory being harder than the fact-based memory from FaE. To address this, we make several key changes to the original FaE architecture. First, the query vector is constructed as a function of the topic mention embedding and relation embedding. Second, we modify the learned a mixing weight between the memory retrieval results and language model predictions. Third, we extend OPQL-LM to answer multi-hop questions. We will elaborate these changes in the rest of this section.[5]

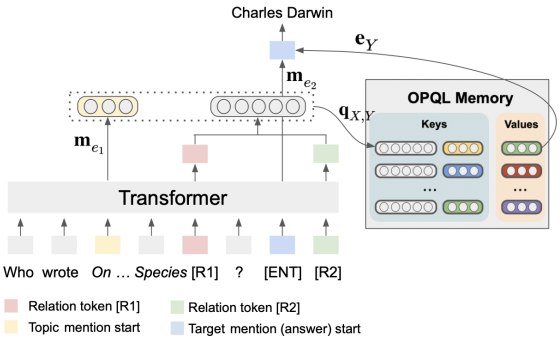

Figure 3. **OPQL-LM model architecture**. A query vector $\mathbf{q}_{X,Y}$ is constructed from the contextual embedding of the topic mention $\mathbf{m}_{e_1}$ and the relation embedding $\mathbf{r}_{X,Y}$, and retrieves the top few entries from the OPQL memory. The retrieval results are aggregated into a single vector $\mathbf{e}_Y$, and mixed with the contextual embedding of the masked mention (answer) $\mathbf{m}_{e_2}$ to make the final prediction.

Input Similar to §3, the special tokens [R1] and [R2] are inserted directly after the mentions of topic entity e_1 and target entity (answer) e_2. But the mention of topic entity e_1 will not be masked with [ENT], e.g. "Who published the book *On the Origin of Species* [R1] in 1859? [ENT] [R2]". We denote the mention of topic entity as m_{e_1}. The mention m_{e_1} is used to extract contextual mention embedding $\mathbf{m}_{e_1}$ for the topic entity e_1, which will be used for entity linking (§2.3), as well as constructing the query to retrieve from the OPQL memory. The training objective of OPQL-LM is to predict the masked target entity (answer) e_2.

Query Embedding The query $\mathbf{q}_{X,Y}$ is computed compositionally by concatenating the contextual mention embedding $\mathbf{m}_{e_1}$ of the topic mention m_{e_1} (as described by entity linking §2.3) and the relation embedding $\mathbf{r}_{X,Y}$. Ideally, $\mathbf{m}_{e_1}$ should be close to the embedding of the oracle topic entity $\mathbf{e}_1$, which is supervised with entity linking loss (Eq. 3). The relation embedding $\mathbf{r}_{X,Y}$ comes from the relation encoder (Eq. 1) that operates on special tokens [R1] and [R2].[6]

$$\mathbf{q}_{X,Y} = \mathbf{W}_q^T[\mathbf{m}_{e_1}; \mathbf{W}_t^T \mathbf{r}_{X,Y}]$$

Mixing with LM Let $\mathbf{e}_Y$ be the retrieved embedding re-

[5] Please refer to the appendix and the Verga et al. (2020) for more details.

[6] Since the mention of entity e_1 is not masked from the input, the relation embedding $\mathbf{r}_{X,Y}$ may contain some leaked information from the topic entity e_1. One could encode the relation embedding $\mathbf{r}_{X,Y}$ separately from an input where both e_1 and e_2 are masked. We do not take this solution as it doubles the computation cost of the expensive Transformer layers.

turned from the OPQL memory and $\mathbf{m}_{e_2}$ be contextual embedding of the masked mention [ENT] predicted from the language model. $\mathbf{e}_Y$ and $\mathbf{m}_{e_2}$ are mixed with a mixing factor λ. λ decides whether retrieved embedding should be included to make final predictions. λ should be large if there is a relevant entry with pair (e_1, e_2) in the memory, so retrieving this pair can help predict the masked entity e_2. Different from Verga et al. (2020)[7], OPQL-LM introduced a `null` relation to the OPQL memory, whose embedding $\mathbf{r}_{\texttt{null}}$ is a learned variable, and constructed a `null` entry with key embedding $\mathbf{k}_{\texttt{null}} = \mathbf{W}_k^T[\mathbf{m}_{e_1}; \mathbf{r}_{\texttt{null}}]$ and value embedding $\mathbf{v}_{\texttt{null}} = \overrightarrow{\mathbf{0}}$. The query is encouraged to retrieve the `null` entry if no relevant pair of (e_1, e_2) exists in the OPQL memory. We re-use the memory key projection matrix $\mathbf{W}_k$ to compute the key embedding $\mathbf{k}_{\texttt{null}}$.

$$\mathbf{m}'_{e_2} = \mathbf{m}_{e_2} + (1 - \lambda) \cdot \mathbf{e}_Y, \;\; \lambda = \text{softmax}(\mathbf{q}_{X,Y}^T \, \mathbf{k}_{\texttt{null}}) \tag{6}$$

Multi-hop OPQL-LM Some open-domain questions require multi-hop reasoning to find the answers, e.g. "Where is the author of *On the Origin of Species* educated?". To answer such questions, the model should first find the answer of the first-hop of the question, and use the it as topic entity to answer the second-hop of the questions. OPQL-LM can naturally solve this problem by repeating the retrieval and mixing steps.

Let $\mathbf{m}'^{\,(t)}_{e_2}$ from Eq. 6 be the memory-injected contextual embedding that was used to predict the answer of the t'th hop. In the second hop, $\mathbf{m}'^{\,(t)}_{e_2}$ becomes the embedding of the topic entities and used to compute query $\mathbf{q}_{X,Y}^{(t+1)}$ for the $(t+1)$'th hop. Again, we keep the relation embedding $\mathbf{r}_{X,Y}$ unchanged, but learn another projection matrix $\mathbf{W}_t^{(t+1)}$. $\mathbf{m}'^{\,(t+1)}_{e_2}$ will be computed accordingly with updated $\mathbf{m}_{e_2}^{(t+1)}$, $\lambda^{(t+1)}$ and $\mathbf{e}_Y^{(t+1)}$ at the $(t+1)$'th hop.

$$\mathbf{q}_{X,Y}^{(t+1)} = \mathbf{W}_q^T[\mathbf{m}'^{\,(t)}_{e_2}; \mathbf{W}_t^{(t+1)T} \mathbf{r}_{X,Y}]$$

4.1. Pretraining OPQL-LM

We propose a second stage pretraining for OPQL-LM that learns to retrieve from the OPQL memory and compute the mixing weight λ. Relation embeddings $\mathbf{r}_{e_1,e_2}$ are precomputed and fixed at the second stage pretraining. We continue training Transformer layers and the entity embedding table $\mathbf{E}$ for entity linking. The second stage pretraining is not required for the relational following task (§3).

Data We use Wikipedia passages as our pretraining data. Given an input C that contains M mentions, we randomly select one mention as our target entity e_2 and mask it with [ENT]. All other mentions are considered topic entities $\{e_1\}$. Special tokens [R1] and [R2] are inserted after mentions of topic and target entities. Mentions of the topic entities will not be masked. Each entity pair (e_1, e_2) will be treated independently. In an example with three mentions, "[ENT] [R2], an *English* [R1] naturalist, published his book *On the Origin of Species* [R1] in 1859.", the mention *Charles Darwin* is selected as the target entity e_2. Retrieval and mixing steps are performed on both topic entities *English* and *On the Origin of Species* independently to predict the masked entity *Charles Darwin*. OPQL-LM is trained on 85.6M text pieces with a length of 128 tokens.

Please refer to the paper by Verga et al. (2020) for more discussion on the loss terms and finetuning details. Besides the parameters in Verga et al. (2020), we additionally train the null relation embedding $\mathbf{r}_{\text{null}}$ and finetune the relation projection matrix $\mathbf{W}_t^{(\cdot)}$.

4.2. Experiments: Integrating VKB with LMs

Next, we experiment with OPQL in another realistic scenario – open-domain QA. In the relational following task (§3) where questions are often semi-structured and oracle topic entities are provided. In open-domain QA, questions are more diverse natural language and do not contain oracle-linked entities. In our experiments, we make a weaker assumption that mention detection has been run on the input providing boundaries of entity mentions to the model. OPQL can effectively solve open-domain QA by retrieving relevant entries from the memory and return the corresponding values as answers and the retrieval results from the memory can be mixed with language model predictions to further improve the performance. We call this OPQL-LM.

4.2.1. Dataset

WebQuestionSP (WebQSP) (Yih et al., 2015) is an open-domain Question Answering dataset that contains 4737 factual questions posed in natural languages. Answers to the questions are labeled with entities in Freebase. Since the pretraining was performed on Wikidata, we convert the Freebase entity ids (MIDs) to Wikidata ids (QIDs). After the conversion, 88.2% questions are answerable by QIDs.

ComplexWebQuestions (ComplexWebQ) (Talmor & Berant, 2018) extends WebQuestionsSP to multi-hop questions. The ComplexWebQuestions dataset contains 34,689 complex questions, including 45% composition questions, 45% conjunction questions[8], and 10% others. Similar to the WebQuestionsSP dataset, we convert Freebase MIDs to Wikidata QIDs. 94.2% of the questions are answerable.

4.2.2. Baselines

We compare OPQL with several strong open-domain QA baselines. GRAFT-Net and PullNet (Sun et al., 2018; 2019)

[7] FaE (Verga et al., 2020) introduced a `null` fact whose embedding $\mathbf{k}_{\texttt{null}}$ is a learned variable.

[8] See Appendix for details of solving conjunction questions.

are Graph-CNN based models that are introduced in the previous experiments. BART (Lewis et al., 2019) and T5 (Raffel et al., 2019) are large pretrained text-to-text Transformer models.[9] EaE (Févry et al., 2020) is trained to predict masked entities, but without an external (virtual) KB memory. EaE does not have an external reasoning module, so it requires the reasoning process performed all from the Language Model. DPR (Karpukhin et al., 2020) is a retrieve-and-read approach that pairs a dense retriever in the embedding space with a BERT based reader. To extend DPR to handle multi-hop questions, we concatenate answers from the previous hops to the question as the query for the next hop. This is referred as DPR-cascade.[10] We also include results of FaE though its external memory is built from KB.

4.2.3. Results

OPQL-LM outperforms the baseline models on both datasets (Table 2). We also experimented with an ablated model OPQL-follow that only performs multi-hop relational following on questions,[11] i.e. retrieved embeddings $\mathbf{e}_Y$ is directly used to make predictions. In WebQuestionsSP, the accuracy of retrieving the relevant pair is 85.4% and that leads to 46.6% accuracy of the WebQuestionsSP dataset. However, in ComplexWebQuestions, the coverage of OPQL memory to answer both hops of the questions is only 22.1% for compositional questions.[12] So the accuracy of OPQL-follow is bounded. Mixing OPQL with LM can further improve the performance of OPQL-follow by 5.3% on WebQuestionsSP and 22.4% on ComplexWebQuestions.

The entity-dependent null embedding $\mathbf{k}_{\text{null}}$ is crucial for OPQL-LM. In an ablation study that the null embedding does not depend on the topic entity e_1, i.e. $\mathbf{k}_{\text{null}}$ is a learned variable shared by all entities, the performance on WebQuestions drops from 51.9 to 46.3 with the retrieval accuracy dropping from 85.4 to 69.5. The accuracy on Complex WebQuestions drops from 40.7 to 19.3.

4.2.4. Injecting Entries into the OPQL Memory

We show that OPQL can efficiently injecting new pairs to the memory to improve the coverage of knowledge at finetuning time, without having to retrain the relation encoder or LM. In the WebQuestionsSP experiment above, the pretrained OPQL memory that contains 1.6M popular entity pairs (800k pairs plus their inverse) only covers 54.6% of the questions, and each question entity was only included in 8.4 pairs in the memory of pre-selected entity pairs. To improve the coverage, we added 100 pairs per question entity with the highest PMI to the memory. The updated memory contains 1.8M pairs and the coverage increases to 82.9%.

The result with the updated memory is presented in Table 2. The retrieval accuracy on questions that has relevant pairs in the memory drops from 85.4% to 62.2% as a result of adding 10 times more pairs for each question entities, but the overall Hits@1 accuracy of the model improves from 51.9% to 53.7%. We also finetune OPQL-LM on ComplexWebQuestions dataset. The improvement is less significant since retrieval is harder on complex questions.

Model	WebQSP	ComplexWebQ (dev)
GRAFT-NET	25.3	10.6
PullNet	24.8	13.1
BART-Large	30.4	-
EaE	47.4	31.3
DPR	48.6	24.6
DPR-cascade	-	25.1
T5	49.7	38.7
OPQL-follow	46.6	18.5
OPQL-LM	**51.9**	**40.7**
+ *webqsp pairs*	*53.7*	*41.1*
FaE	54.7	-
SoA (KB)	69.0 (F1)	47.2

Table 2. Hits@1 performance on open-domain QA datasets. OPQL+webqsp pairs injects additional data specific memories. The state-of-the-art model on WebQSP (NSM (Liang et al., 2017)) and ComplexWebQ (PullNet-KB (Sun et al., 2019)) both use Freebase to answer the questions. FaE (Verga et al., 2020) has an external memory with KB triples.

5. Related Work

Automatically extracting triples from a text corpus has been pursued for many years as a means of improving the coverage of KBs (Mitchell et al., 2015). Rather than extracting triples into a predefined vocabulary, OpenIE (typically) uses linguistic patterns to extract open vocabulary relations from text (Etzioni et al., 2008; Fader et al., 2011). Gupta et al. (2019)) build an Open Knowledge Graph over Open IE extractions similar to a VKB.

A few methods (Seo et al., 2018; Dhingra et al., 2020) have built pre-computed memories of mention embeddings which are used directly to answer questions. These methods are most similar to our work though OPQL differs by embedding mention pairs, rather than single mentions, and additionally, our method requires no structured supervision. Another related of work proposed to build a passage level index (Karpukhin et al., 2020) to improve retrieval accu-

[9] WebQuestionsSP is finetuned on T5-11B (Verga et al., 2020). Due to hardware constraint, we finetune T5-3B for ComplexWebQuestions.

[10] We run DPR and DPR-cascade with the pretrained checkpoint on Natural Questions (Kwiatkowski et al., 2019). Finetuned DPR reader on ComplexWebQuestions only gets 18.2% Hits@1.

[11] We do not assume oracle entity linking here. The model use the contextual embedding of the topic entity to construct the query.

[12] This number is computed from the intermediate answers provided in the dataset. We do not use the intermediate answers in training.

racy in the "retrieve and read" pipeline. Multi-hop retrieval models (Qi et al., 2020) are proposed for complex questions. Other approaches propose to retrieve embedded passages which are then passed to an LM to reason over (Karpukhin et al., 2020; Lewis et al., 2020; Guu et al., 2020; Lee et al., 2019). In contrast OPQL does not include a separate model for reading retrieved documents.

Other work has shown that injecting an external memory constructed from KB into a LM can help training LMs (Peters et al., 2019), improve generation tasks, (Logan et al., 2019b), and enable reasoning over an updated memory (Verga et al., 2020). Additionally, our model's memory scales to millions of entries, whereas most prior systems that use KB triples have been with only a few hundreds of triples in the model at any point, necessitating a separate heuristic process to retrieve candidate KB triples (Ahn et al., 2016; Henaff et al., 2016; Weissenborn et al., 2017; Chen et al., 2018; Mihaylov & Frank, 2018; Logan et al., 2019a).

6. Conclusion

We proposed OPQL that can construct a virtual knowledge base from a text corpus without any supervision from existing KB. The pretrained OPQL can effectively solve relational following task, achieving the state-of-the-art performance on two multi-hop relational following datasets. The improvement is more significant if evaluated on queries with relations not seen at training time. OPQL can be injected into a language model to answer open-domain questions. It outperforms several large pretrained language models on two benchmark open-domain QA datasets.

7. Acknowledgement

This work was supported in part by NSF IIS1763562.

References

Ahn, S., Choi, H., Pärnamaa, T., and Bengio, Y. A neural knowledge language model. *arXiv preprint arXiv:1608.00318*, 2016.

Angeli, G., Premkumar, M. J. J., and Manning, C. D. Leveraging linguistic structure for open domain information extraction. 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)*, pp. 344–354, 2015.

Baldini Soares, L., FitzGerald, N., Ling, J., and Kwiatkowski, T. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2895–2905, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1279. URL https://www.aclweb.org/anthology/P19-1279.

Chen, D., Fisch, A., Weston, J., and Bordes, A. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1870–1879, 2017.

Chen, Q., Zhu, X., Ling, Z.-H., Inkpen, D., and Wei, S. Neural natural language inference models enhanced with external knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2406–2417, 2018.

Cohen, W. W., Sun, H., Hofer, R. A., and Siegler, M. Scalable neural methods for reasoning with a symbolic knowledge base. *International Conference on Learning Representations*, 2020.

Dhingra, B., Zaheer, M., Balachandran, V., Neubig, G., Salakhutdinov, R., and Cohen, W. W. Differentiable reasoning over a virtual knowledge base. In *International Conference on Learning Representations*, 2020.

Etzioni, O., Banko, M., Soderland, S., and Weld, D. S. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74, 2008.

Fader, A., Soderland, S., and Etzioni, O. Identifying relations for open information extraction. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pp. 1535–1545, 2011.

Févry, T., Soares, L. B., FitzGerald, N., Choi, E., and Kwiatkowski, T. Entities as experts: Sparse memory access with entity supervision. *Conference on Empirical Methods in Natural Language Processing*, 2020.

Gupta, S., Kenkre, S., and Talukdar, P. CaRe: Open knowledge graph embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 378–388, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1036. URL https://www.aclweb.org/anthology/D19-1036.

Guu, K., Lee, K., Tung, Z., Pasupat, P., and Chang, M.-W. Realm: Retrieval-augmented language model pretraining. *arXiv preprint arXiv:2002.08909*, 2020.

Henaff, M., Weston, J., Szlam, A., Bordes, A., and LeCun, Y. Tracking the world state with recurrent entity networks. *arXiv preprint arXiv:1612.03969*, 2016.

Karpukhin, V., Oğuz, B., Min, S., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*, 2020.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Kelcey, M., Devlin, J., Lee, K., Toutanova, K. N., Jones, L., Chang, M.-W., Dai, A., Uszkoreit, J., Le, Q., and Petrov, S. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 2019.

Lee, K., Chang, M.-W., and Toutanova, K. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6086–6096, 2019.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*, 2020.

Liang, C., Berant, J., Le, Q., Forbus, K. D., and Lao, N. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision, 2017.

Ling, J., FitzGerald, N., Shan, Z., Soares, L. B., Févry, T., Weiss, D., and Kwiatkowski, T. Learning cross-context entity representations from text. *arXiv preprint arXiv:2001.03765*, 2020.

Logan, R., Liu, N. F., Peters, M. E., Gardner, M., and Singh, S. Barack's wife hillary: Using knowledge graphs for fact-aware language modeling. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019a. doi: 10.18653/v1/p19-1598. URL http://dx.doi.org/10.18653/v1/P19-1598.

Logan, R., Liu, N. F., Peters, M. E., Gardner, M., and Singh, S. Barack's wife hillary: Using knowledge graphs for fact-aware language modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5962–5971, 2019b.

Mihaylov, T. and Frank, A. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 821–832, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1076. URL https://www.aclweb.org/anthology/P18-1076.

Miller, A., Fisch, A., Dodge, J., Karimi, A.-H., Bordes, A., and Weston, J. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1400–1409, 2016.

Min, B., Grishman, R., Wan, L., Wang, C., and Gondek, D. Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 777–782, 2013.

Mitchell, T., Cohen, W., Hruschka, E., Talukdar, P., Betteridge, J., Carlson, A., Dalvi, B., Gardner, M., Kisiel, B., Krishnamurthy, J., Lao, N., Mazaitis, K., Mohamed, T., Nakashole, N., Platanios, E., Ritter, A., Samadi, M., Settles, B., Wang, R., Wijaya, D., Gupta, A., Chen, X., Saparov, A., Greaves, M., and Welling, J. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, 2015.

Newell, A. and Simon, H. The logic theory machine–a complex information processing system. *IRE Transactions on information theory*, 2(3):61–79, 1956.

Newell, A., Shaw, J. C., and Simon, H. A. Report on a general problem-solving program. In *Proceedings of the International Conference on Information Processing*, 1959.

Peters, M. E., Neumann, M., Logan, R., Schwartz, R., Joshi, V., Singh, S., and Smith, N. A. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 43–54, 2019.

Qi, P., Lee, H., Sido, O., Manning, C. D., et al. Retrieve, rerank, read, then iterate: Answering open-domain questions of arbitrary complexity from text. *arXiv preprint arXiv:2010.12527*, 2020.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

Seo, M., Kwiatkowski, T., Parikh, A., Farhadi, A., and Hajishirzi, H. Phrase-indexed question answering: A new challenge for scalable document comprehension. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 559–564,

Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1052. URL https://www.aclweb.org/anthology/D18-1052.

Seo, M., Lee, J., Kwiatkowski, T., Parikh, A., Farhadi, A., and Hajishirzi, H. Real-time open-domain question answering with dense-sparse phrase index. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4430–4441, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1436. URL https://www.aclweb.org/anthology/P19-1436.

Sun, H., Dhingra, B., Zaheer, M., Mazaitis, K., Salakhutdinov, R., and Cohen, W. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4231–4242, 2018.

Sun, H., Bedrax-Weiss, T., and Cohen, W. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2380–2390, 2019.

Sun, H., Arnold, A., Bedrax Weiss, T., Pereira, F., and Cohen, W. W. Faithful embeddings for knowledge base queries. *Advances in Neural Information Processing Systems*, 33, 2020.

Talmor, A. and Berant, J. The web as a knowledge-base for answering complex questions. In *North American Association for Computational Linguistics (NAACL)*, 2018.

Verga, P., Sun, H., Soares, L. B., and Cohen, W. W. Facts as experts: Adaptable and interpretable neural memory over symbolic knowledge. *arXiv preprint arXiv:2007.00849*, 2020.

Weissenborn, D., Kočiskỳ, T., and Dyer, C. Dynamic integration of background knowledge in neural nlu systems. 2017.

Yih, W.-t., Chang, M.-W., He, X., and Gao, J. 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)*, pp. 1321–1331, Beijing, China, July 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-1128.

Zhang, Y., Dai, H., Kozareva, Z., Smola, A. J., and Song, L. Variational reasoning for question answering with knowledge graph. In *AAAI*, 2018.

8. Appendix

8.1. OPQL-ML Retrieval and Mixing Details

Retrieval from OPQL Memory Retrieving from the OPQL memory is analogous to running a relational following operation $Y = X.\text{follow}(R)$ with a learned set of relations R, but the set of topic entities X is also unknown. Recall that the relational following task uses the weighted X to compute its query embedding $\mathbf{q}_{X,Y} = \mathbf{W}_q^T[\mathbf{e}_X; \mathbf{W}_t^T \mathbf{r}_{X,Y}]$. In the open-domain QA task, we do not assume oracle entity linking is provided, and thus not able to compute $\mathbf{e}_X$ explicitly from set X. Instead, we consider the contextual mention embedding $\mathbf{m}_{e_1}$ as an approximation of the centroid $\mathbf{e}_X$. Ideally, $\mathbf{m}_{e_1}$ should be close to the embedding $\mathbf{e}_1$ of the oracle topic entity. The query $\mathbf{q}_{X,Y}$ is computed compositinoally by concatenating the contextual mention embedding $\mathbf{m}_{e_1}$ and the relation embedding $\mathbf{r}_{X,Y}$. The relation embedding $\mathbf{r}_{X,Y}$ comes from the relation encoder (Eq. 1) that operates on special tokens [R1] and [R2].

$$\mathbf{q}_{X,Y} = \mathbf{W}_q^T[\mathbf{m}_{e_1}; \mathbf{W}_t^T \mathbf{r}_{X,Y}]$$

Since the mention of entity e_1 is not masked from the input, the relation embedding $\mathbf{r}_{X,Y}$ may contains some leaked information from the topic entity e_1. One potential fix is to encode the relation embedding $\mathbf{r}_{X,Y}$ separately from an input where both e_1 and e_2 are masked. We do not take this solution as it doubles the computation cost of the expensive Transformer layers.

As discussed in §3, the result of the relational operation $Y = X.\text{follow}(R)$ contains values $\{e_j\}$ of the top k retrieved pairs $\{(e_i, e_j)\}$, s.t. $(e_i, e_j) \in \text{top}_k(\mathbf{q}_{X,Y}, \mathbf{K})$. The weight β_{e_i,e_j} is the softmax of the retrieval score. We aggregate the embeddings of the retrieved entities $e_j \in Y$ into a single vector $\mathbf{e}_Y$. $\mathbf{e}_Y$ will be mixed with the contextual embedding of the masked mention [ENT] to predict the masked target entity e_2.

$$\mathbf{e}_Y = \sum_{(e_i,e_j)} \beta_{e_i,e_j} \mathbf{e}_j, \;\; (e_i, e_j) \in \text{top}_k(\mathbf{q}_{X,Y}, \mathbf{K}) \quad (7)$$

$$L_{\text{re}} = \text{cross\_entropy}(\beta_{e_i,e_j}, \mathbb{I}_{e_j \in \text{Ans}}) \quad (8)$$

Mixing with LM The language model computes the contextual embedding of the masked entity e_2. Let $m_{e_2} = (c_i, c_i, e_2)$ be the masked mention [ENT] of the target entity (answer) e_2 that locates at the i'th token of the input sequence. The contextual embedding $\mathbf{m}_{e_2}$ is a projection of Transformer output $\mathbf{h}_i$ at the token c_i, that shares the same projection matrix $\mathbf{W}_e$ with the entity linking task in §2.3.

$$\mathbf{m}_{e_2} = \mathbf{W}_e^T \mathbf{h}_i \quad (9)$$

The retrieved embeddings $\mathbf{e}_Y$ (Eq. 7) and contextual embeddings $\mathbf{m}_{e_2}$ (Eq.9) of the masked mention are mixed with a mixing factor λ. λ decides whether retrieved memory should be added to the contextual embedding. λ should be large if there is a relevant entry with pair (e_1, e_2) in the memory, so retrieving this pair can help predict the masked entity e_2. As suggested by Verga et al. (2020), we introduced a `null` relation to the OPQL memory, whose embedding $\mathbf{r}_{\texttt{null}}$ is a learned variable, and constructed a `null` entry with key embedding $\mathbf{k}_{\texttt{null}} = \mathbf{W}_k^T[\mathbf{m}_{e_1}; \mathbf{r}_{\texttt{null}}]$ and value embedding $\mathbf{v}_{\texttt{null}} = \overrightarrow{0}$. The query is encouraged to retrieve the `null` pair if no relevant pair of (e_1, e_2) exists in the OPQL memory. We re-use the projection matrix $\mathbf{W}_k$ to compute the key embedding $\mathbf{k}_{\texttt{null}}$.

$$\lambda = \text{softmax}(\mathbf{q}_{X,Y}^T \, \mathbf{k}_{\texttt{null}}) \quad (10)$$

$$\mathbf{m}'_{e_2} = \mathbf{m}_{e_2} + \lambda \cdot \mathbf{e}_Y \quad (11)$$

The memory-injected contextual embedding $\mathbf{m}'_{e_2}$ is used to predict the masked entities e_2.

$$L_{\text{mel}} = \text{cross\_entropy}(\text{softmax}({\mathbf{m}'_{e_2}}^T \mathbf{e}_i), \mathbb{I}_{e_i \in \text{Ans}}) \quad (12)$$

Loss Besides the entity linking loss on masked mention $\mathbf{m}'_{e_2}$ (Eq. 12), we jointly train entity linking on topic entity e_1 (Eq. 3), and also provide supervision on OPQL memory retrieval (Eq. 8).

$$L_{\text{OPQL-LM}} = L_{\text{el}} + L_{\text{re}} + L_{\text{mel}}$$

8.2. OPQL-LM for Conjunction Questions

A conjunction questions, e.g. "Which *English* author publish his book *On the Origin of Species*?", contains more than one topic entity *English* and *On the Origin of Species*. Both topic entities can potentially help to predict the answer, e.g. with pairs (*English*, *Charles Darwin*) that describes his nationality and (*On the Origin of Species*, *Charles Darwin*) that describes his publications. We convert the input by appending the masked answer to the end of the question and inserting the special tokens [R1] and [R2] accordingly, e.g. "Which *English* [R1] author publish his book *On the Origin of Species* [R1]? [ENT] [R2]".

A query vector $\mathbf{q}_{X,Y,e_i}$ is constructed for each topic entity e_i, with the retrieval results $\mathbf{e}_{Y,e_i}$ returned from the memory. All retrieved embeddings are mixed with the contextual embedding $\mathbf{m}_{e_2}$ to predict the final answer

$$\mathbf{m}'_{e_2} = \mathbf{m}_{e_2} + \sum_{e_i} \lambda_i \cdot \mathbf{e}_{Y,e_i}$$

where λ_i is the mixing weight that is determined by each query independently.