Adaptable and Interpretable Neural Memory Over Symbolic Knowledge

Pat Verga*, Haitian Sun*, Livio Baldini Soares, William W. Cohen
Google Research
{patverga, haitiansun, liviobs, wcohen}@google.com

Abstract

Past research has demonstrated that large neural language models (LMs) encode surprising amounts of factual information: however, augmenting or modifying this information requires modifying a corpus and retraining, which is computationally expensive. To address this problem, we develop a neural LM that includes an interpretable neuro-symbolic KB in the form of a "fact memory". Each element of the fact memory is formed from a triple of vectors, where each vector corresponds to a KB entity or relation. Our LM improves performance on knowledge-intensive question-answering tasks, sometimes dramatically, including a 27 point increase in one setting of WebQuestionsSP over a state-of-the-art open-book model, despite using 5% of the parameters. Most interestingly, we demonstrate that the model can be modified, without *any* re-training, by updating the fact memory.

1 Introduction

Neural language models (LMs) (Peters et al., 2018; Devlin et al., 2019; Raffel et al., 2019) that have been pre-trained by self-supervision on large corpora contain rich knowledge about the syntax and semantics of natural language (Tenney et al., 2019), and are the basis of much recent work in NLP. Pre-trained LMs also contain large amounts of *factual* knowledge about the world (Petroni et al., 2019; Roberts et al., 2020; Brown et al., 2020). However, while large LMs can be coerced to answer factual queries, they still lack many of the properties that knowledge bases (KBs) typically have. In particular, it is difficult to distinguish answers produced by memorizing factual statements in the pre-training corpus from lower-precision answers produced by linguistic generalization (Poerner et al., 2019). It is also difficult to add or remove factual information without retraining the LM, an expensive process[1]. The difficulty of updating knowledge in neural LMs contrasts with symbolic KBs, where it is very easy to add or modify triples, and is a major disadvantage of using a LM "as a KB"—as in many domains (news, product reviews, scientific publications, etc) the set of known facts changes frequently. Symbolic KBs thus remain practically important (Google, 2012; Dong, 2017), especially for NLP applications where text is hard to automatically process (e.g., scientific, technical, or legal) or tasks rich in information that exists only in structured form (e.g., technical specifications of a new product, where no product page or review text discussing it yet exists).

Motivated by this, past work has sought to combine the benefits of neural LMs with the large, broad-coverage KBs that now exist (Bollacker et al., 2008; Auer et al., 2007; Vrandečić and Krötzsch, 2014). This paper continues this research program with a new knowledge-augmented LM called *Fact Injected Language Model* (FILM). FILM is a masked LM, where masks can be filled either from the token vocabulary or an entity vocabulary. The vector representation of each entity in a KB is jointly learned alongside other parameters of a Transformer LM, and stored in a separate *entity memory*. FILM also includes a *fact memory* where each element is derived from a triple of vectors, representing a KB entity or relation. Since these triples are defined compositionally from (representations of) entities and relations, they have an interpretable symbolic meaning: e.g., if $\mathbf{e}_{\text{mtv}}$ is the vector representation of KB entity "Mountain View, CA" and $\mathbf{e}_{\text{google}}$ and $\mathbf{r}_{\text{hq}}$ similarly correspond to "Google Inc" and the relation "headquartered in", these vectors can be used to construct a memory element $f(\mathbf{e}_{\text{google}}, \mathbf{r}_{\text{hq}}, \mathbf{e}_{\text{mtv}})$ for the KB assertion "Google, Inc is headquartered in Mountain

[1]Models large enough to achieve good factual coverage require extreme amounts of compute, and the largest neural LMs now cost millions of dollars to train (Brown et al., 2020).

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 3678–3691
June 6–11, 2021.

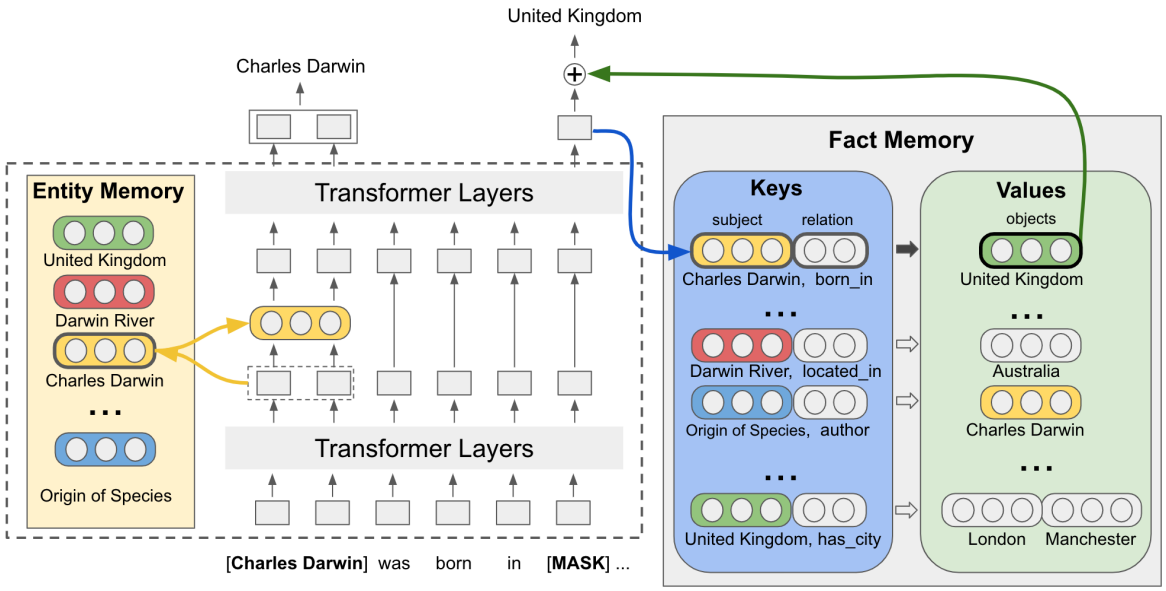

Figure 1: **Fact Injected Language Model architecture.** The model takes a piece of text (a question during fine-tuning or arbitrary text during pre-training) and first contextually encodes it with an entity enriched transformer. FILM uses the contextually encoded MASK token as a query to the fact memory. In this case, the contextual query chooses the fact key (*Charles Darwin, born_in*) which returns the set of values {*United Kingdom*} (The value set can be multiple entity objects such as the case from calling the key [*United Kingdom, has_city*]) . The returned object representation is incorporated back into the context in order to make the final prediction. Note that the entity representations in the facts (both in keys and values) are shared with the entity memory. The portion within the dashed line follows the procedure from Févry et al. (2020).

View, CA". This means that the fact memory can be easily extended with new facts.

In analysis on four benchmark question answering datasets we show that FILM improves significantly, and sometimes dramatically, over several strong baselines (e.g. BART (Lewis et al., 2019) and T5 (Raffel et al., 2019)) and this improvement is even larger when removing train-test overlap. In one setting of WebQuestionsSP, we outperform the next best performing model (RAG (Lewis et al., 2020a)) by 27 points despite using only 5% of the number of parameters.

Most interestingly, we demonstrate that FILM models can be updated without *any* re-training, by modifying the fact memory. Specifically, in §4.1, we show we can inject new fact memories at inference time, enabling FILM to correctly answer questions about pairs of entities that were *never* observed in the training (either during pre-training or fine-tuning). In §4.2 we also evaluate updating the model by inserting contra-positive facts that *contradict* facts mentioned in the pretraining data, and we show that FILM can correctly answer novel questions in this scenario as well. To summarize, this paper's contributions are:

1. We propose a neural LM for knowledge-intensive question-answering tasks that incorporates a symbolic fact memory.
2. We outperform most baselines on several benchmark open-domain QA datasets, and dramatically if test-train overlap in the datasets are removed.
3. We show FILM can easily adapt to newly injected and modified facts without retraining.

2 Fact Injected Language Model Model

The Fact Injected Language Model (FILM) model (see Figure 1) extends the Transformer (Vaswani et al., 2017) architecture of BERT (Devlin et al., 2019) with additional entity and facts memories. These memories store semantic information which can later be retrieved and incorporated into the representations of the transformer. Similar to the approach in Févry et al. (2020), entity embeddings will (ideally) store information about the textual contexts in which that entity appears, and by inference, the entity's semantic properties. The *fact memory* encodes triples from a symbolic KB, constructed compositionally from the learned embeddings of the entities that comprise it and implemented as a key-value memory which is used to retrieve entities given their KB properties. This combination results in a neural LM which learns to

access information from a symbolic KB.

2.1 Definitions

We represent a Knowledge Base $\mathcal{K}$ as a set of triples (s, r, o) where $s, o \in \mathcal{E}$ are the subject and object entities and $r \in \mathcal{R}$ is the relation, where $\mathcal{E}$ and $\mathcal{R}$ are pre-defined vocabularies of entities and relations. A text corpus $\mathcal{C}$ is a collection of paragraphs[2] $\{p_1, \ldots, p_{|C|}\}$. Let $\mathcal{M}$ be the set of entity mentions in the corpus $\mathcal{C}$. A mention m_i is encoded as (e_m, s_m^p, t_m^p), indicating entity e_m is mentioned in paragraph p starting at token position s_m^p and ending at t_m^p. We will usually drop the superscript p and use s_m and t_m for brevity.

2.2 Input

The input to our model is a piece of text; either a question during fine tuning (see §A.2.2) or a paragraph in pre-training (see §A.2.1). Pretraining is formulated as a cloze-type Question Answering (QA) task: given a paragraph $p = \{w_1, \ldots, w_{|p|}\}$ with mentions $\{m_1, \ldots, m_n\}$, we sample a single mention m_i to act as the cloze answer and replace all tokens of m_i with [MASK] tokens. The entity in $\mathcal{E}$ named by the masked entity is the answer to the cloze question q ('United Kingdon' in the example input of Figure 1). Mentions in the paragraph other than m are referred to below as *context mentions*. In the following sections we describe how our model learns to jointly link context entities (§2.3) and predict answer entities (§2.5).

2.3 Entity Memory

Our entity memory $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times d_e}$ is a matrix containing a vector for each entity in $\mathcal{E}$ and trained as an entity-masked LM. The model input is a text span containing unlinked entity mentions with known boundaries[3]. Mentions are masked with some probability. Our entity memory follows Entity as Experts (EaE) (Févry et al., 2020) which interleaves standard Transformer (Vaswani et al., 2017) layers with layers that accesss the entity memory[4].

Given a piece of text $q = \{w_1, \ldots, w_{|q|}\}$ the contextual embedding $\mathbf{h}_i^{(l)}$ is the output at the i'th token of the l'th intermediate transformer layer. These contextual embeddings are used to compute query vectors that interface with the entity memory.

For each context mention $m_i = (e_{m_i}, s_{m_i}, t_{m_i})$ in q, we form a query vector to access the Entity memory by concatenating the context embeddings for the mention m_i's start and end tokens, $\mathbf{h}_{s_{m_i}}^{(l)}$ and $\mathbf{h}_{t_{m_i}}^{(l)}$ and projecting them into the entity embedding space. We use this query to compute attention weights over the full entity vocabulary and produce an attention-weighted sum of entity embeddings $\mathbf{u}_{m_i}^l$. The result is then projected back to the dimension of the j-indexed contextual token embeddings, and added to what would have been the input to the next layer of the Transformer:

$$\mathbf{h}_{m_i}^{(l)} = \mathbf{W}_e^T [\mathbf{h}_{s_{m_i}}^{(l)}; \mathbf{h}_{t_{m_i}}^{(l)}] \quad (1)$$

$$\mathbf{u}_{m_i}^{(l)} = \text{softmax}(\mathbf{h}_{m_i}^{(l)}, \mathbf{E}) \times \mathbf{E} \quad (2)$$

$$\tilde{\mathbf{h}}_j^{(l+1)} = \mathbf{h}_j^{(l)} + \mathbf{W}_2^T \mathbf{u}_{m_i}^{(l)}, \;\; s_{m_i} < j < t_{m_i} \quad (3)$$

After the final transformer layer T, $\mathbf{h}_{m_i}^{(T)}$ is used to predict the context entities $\hat{e_{m_i}}$ and produce a loss with $\mathbb{I}_{e_{m_i}}$, the one-hot label of entity e_{m_i}. Following Févry et al. (2020), we supervise the entity access for the intermediate query vector in Eq. 1.

$$\hat{e}_{m_i} = \text{argmax}_{e_i \in \mathcal{E}} (\mathbf{c}_{m_i}^T \mathbf{e}_i)$$

$$\text{loss}_{\text{ctx}} = \text{cross\_entropy}(\text{softmax}(\mathbf{c}_{m_i}, \mathbf{E}), \mathbb{I}_{e_{m_i}})$$

$$\text{loss}_{\text{ent}} = \text{cross\_entropy}(\text{softmax}(\mathbf{h}_{m_i}^{(l)}, \mathbf{E}), \mathbb{I}_{e_{m_i}})$$

2.4 Fact Memory

FILM contains a second *fact memory*, populated by triples from the knowledge base $\mathcal{K}$, as shown on the right side of Figure 1[5]. The fact memory shares its on entity representations with the entity memory embeddings in $\mathbf{E}$, but each element of the fact memory corresponds to a symbolic substructure, namely a key-value pair $((s, r), \{o_1, \ldots, o_n\})$. The key (s, r) is a (subject entity, relation) pair, and the corresponding value $\{o_1, \ldots, o_n\}$ is the list of object entities associated with s and r, i.e. $(s, r, o_i) \in \mathcal{K}$ for $i = \{1, \ldots, n\}$. Conceptually, KB triples with the same subject entity and relation are grouped into a single element. We call the subject and relation pair $a_j = (s, r) \in A$ a *head pair* and the list of objects $b_j = \{o_1, \ldots, o_n\} \in B$ a *tail set*[6].

[2] Although we use the term paragraph here, in our experiments we use spans of 128 tokens, which need not follow paragraph boundaries.

[3] Févry et al. (2020) also showed the model is capable of learning to predict these boundaries. For simplicity, in this work we assume they are given.

[4] We follow the implementation of Févry et al. (2020) and have a single entity memory access between the fourth and fifth transformer layers.

[5] In our experiments we use a single fact memory access after the final (12th) transformer layer.

[6] The size of the tail set b_j can be large for a popular head pair (s, r). In such cases, we randomly select a few tails and drop the rest of them. The maximum size of the tail set is 32 in the experiments in this paper.

In more detail, we encode a head pair $a_j = (s, r) \in A$ by concatenating embeddings for the subject entity and relation, and then projecting them linearly to a new head-pair embedding space. More precisely, let $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times d_e}$ be the entity embeddings trained in §2.3, and $\mathbf{R} \in \mathbb{R}^{|\mathcal{R}| \times d_r}$ be embeddings of relations $\mathcal{R}$ in the knowledge base $\mathcal{K}$. We encode a head pair a as:

$$\mathbf{a}_j = \mathbf{W}_a^T \, [\mathbf{s}; \mathbf{r}] \; \in \mathbb{R}^{d_a}$$

where $\mathbf{s} \in \mathbf{E}$ and $\mathbf{r} \in \mathbf{R}$ are the embeddings of subject s and relation r, and $\mathbf{W}_a$ is a learned linear transformation matrix. We let $\mathbf{A} \in \mathbb{R}^{|A| \times d_a}$ denote the embedding matrix of all head pairs.

Let the answer for q be denoted e_{ans}, and its masked mention m_{ans} = (e_{ans}, s_{ans}, t_{ans}). For a masked mention m_{ans}, define a query vector to access the fact memory as:

$$\mathbf{v}_{m_{\text{ans}}} = \mathbf{W}_f^T \, [\mathbf{h}_{s_{\text{ans}}}^{(T)}; \mathbf{h}_{t_{\text{ans}}}^{(T)}] \tag{4}$$

where $\mathbf{h}_{s_{\text{ans}}}^{(T)}$ and $\mathbf{h}_{t_{\text{ans}}}^{(T)}$ are the contextual embeddings for the start and end tokens of the mention m_{ans}, and $\mathbf{W}_f$ is the linear transformation matrix into the embedding space of head pairs $\mathbf{A}$.

Head pairs in A are scored by the query vector $\mathbf{v}_{m_{\text{ans}}}$ and the top k head pairs with the largest inner product are retrieved. This retrieval process on the fact memory is distantly supervised. We define a head pair to be a *distantly supervised positive example* $a_{\text{ds}} = (s, r)$ for a passage if its subject entity s is named by a context mention m_i and the masked entity e_{ans} is an element of the corresponding tail set, i.e. $e_{\text{ans}} \in b_{\text{ds}}$. When no distantly supervised positive example exists for a passage, it is trained to retrieve a special "null" fact comprised of the s_{null} head entity and r_{null} relation: i.e. $a_{\text{ds}} = (s_{\text{null}}, r_{\text{null}})$ and its tail set is empty. This distant supervision is encoded by a loss function:

$$\text{TOP}_k(\mathbf{v}_{m_{\text{ans}}}, \mathbf{A}) = \text{argmax}_{k, j \in \{1, \ldots, |A|\}} \mathbf{a}_j^T \mathbf{v}_{m_{\text{ans}}}$$
$$\text{loss}_{\text{fact}} = \text{cross\_entropy}(\text{softmax}(\mathbf{v}_{m_{\text{ans}}}, \mathbf{A}), \mathbb{I}_{a_{\text{ds}}})$$

The result of this query is that the tail sets associated with the top k scored head pairs, i.e. $\{b_j | j \in \text{TOP}_k(\mathbf{v}, \mathbf{A})\}$, are retrieved from the fact memory.

2.5 Integrating Knowledge and Context

Next, tail sets retrieved from the fact memory are aggregated. Recall that a tail set b_j returned from the fact memory is the set of entities $\{o_1, \ldots, o_n\}$ s.t. $(s, r, o_i) \in \mathcal{K}$ for $i \in \{1, \ldots, n\}$ with the associated $a_j = (s, r)$. Let $\mathbf{o}_i \in \mathbf{E}$ be the embedding of entity o_i. We encode the returned tail set $\mathbf{b}_j$ as a weighted centroid of the embeddings of entities in the tail set b_j.

$$\mathbf{b}_j = \sum_{o_i \in b_j} \alpha_i \mathbf{o}_i \; \in \mathbb{R}^{d_e}$$

where α_i is a context-dependent weight of the object entity o_i. To compute the weights α_i, we use a process similar to Eq. 4: we compute a second query vector $\mathbf{z}_{m_{\text{ans}}}$ to score the entities inside the tail set b_j, and the weights α_i are the softmax of the inner products between the query vector $\mathbf{z}_{m_{\text{ans}}}$ and the embeddings of entities in the tail set b_j.

$$\mathbf{z}_{m_{\text{ans}}} = \mathbf{W}_b^T [\mathbf{h}_{s_{\text{ans}}}^{(T)}; \mathbf{h}_{t_{\text{ans}}}^{(T)}] \tag{5}$$

$$\alpha_i = \frac{\exp(\mathbf{o}_i^T \, \mathbf{z}_{m_{\text{ans}}})}{\sum_{o_l \in b_j} \exp(\mathbf{o}_l^T \, \mathbf{z}_{m_{\text{ans}}})} \tag{6}$$

where $\mathbf{W}_b$ is a transformation matrix distinct from $\mathbf{W}_e$ in Eq. 1 and $\mathbf{W}_f$ in Eq. 4. The top k tail sets $\mathbf{b}_j$ are further aggregated using weights β_j, which are the softmax of the retrieval (inner product) scores of the top k head pairs a_j. This leads to a single vector $\mathbf{f}_{m_{\text{ans}}}$ that we call the *knowledge embedding* for the masked mention m_{ans}.

$$\mathbf{f}_{m_{\text{ans}}} = \sum_{j \in \text{TOP}_k(\mathbf{v}_{m_{\text{ans}}}, \mathbf{A})} \beta_j \mathbf{b}_j \tag{7}$$

$$\beta_j = \frac{\exp(\mathbf{a}_j^T \, \mathbf{v}_{m_{\text{ans}}})}{\sum_{t \in \text{TOP}_k(\mathbf{v}_{m_{\text{ans}}}, \mathbf{A})} \exp(\mathbf{a}_t^T \, \mathbf{v}_{m_{\text{ans}}})} \tag{8}$$

Intuitively $\mathbf{f}_{m_{\text{ans}}}$ is the result of retrieving a set of entities from the fact memory. The last step is to integrate this retrieved set into the Transformer's contextual embeddings. Of course, KBs are often incomplete, and especially during pre-training, it might be necessary for the model to ignore the result of retrieval, if no suitable triple appears in the KB. To model this, the final step in the integration process is to construct an integrated query $\mathbf{q}_{m_{\text{ans}}}$ with a learnable mixing weight λ. Algorithmically, λ is computed as the probability of retrieving a special "null" head a_{null} from the fact memory, i.e. whether an oracle head pair exists in the knowledge base. $\mathbf{q}_{m_{\text{ans}}}$ is used to predict the masked entity.

$$\mathbf{q}_{m_{\text{ans}}} = \lambda \cdot \mathbf{c}_{m_{\text{ans}}} + (1 - \lambda) \cdot \mathbf{f}_{m_{\text{ans}}}, \quad \lambda = P(a_{\text{null}})$$
$$\hat{e}_{\text{ans}} = \text{argmax}_{e_i \in \mathcal{E}} (\mathbf{q}_{m_{\text{ans}}}^T \mathbf{e}_i)$$
$$\text{loss}_{\text{ans}} = \text{cross\_entropy}(\text{softmax}(\mathbf{q}_{m_{\text{ans}}}, \mathbf{E}), \mathbb{I}_{e_{\text{ans}}})$$

Model	P@1
K-Adapter †	29.1
BERT-Large †	33.9
BERT-KNN ‡	38.7
EaE	38.6
FILM	**44.2**

Table 1: **LAMA TREx** Precision@1. † copied from Wang et al. (2020a), ‡ copied from Kassner and Schütze (2020)

The final loss is the sum of the individual losses (See §A.2.1 and §A.2.2 for additional details.)

$$\text{loss}_{\text{final}} = \text{loss}_{\text{ent}} + \text{loss}_{\text{ctx}} + \text{loss}_{\text{fact}} + \text{loss}_{\text{ans}}$$

3 Experiments

The primary focus of this work is investigating the incorporating of new symbolic knowledge by injecting new facts without retraining (§4.1) and updating stale facts (§4.2). However, we first validate the efficacy of our model on standard splits of widely used knowledge-intensive benchmarks against many state-of-the-art systems (§3.3), as well as two subsets of these benchmarks restricted to examples answerable with wikidata (§3.4) and examples filtered for train/test overlap (§3.5).

3.1 Data

We evaluate on four knowledge intensive tasks[7].
WebQuestionsSP is an Open-domain Question Answering dataset containing 4737 natural language questions linked to corresponding Freebase entities and relations (Yih et al., 2015) derived from WebQuestions(Berant et al., 2013).
LAMA TREx is a set of fact-related cloze questions. Since we are interested in entity prediction models, we restrict our LAMA investigations to TREx, which has answers linked to Wikidata.
TriviaQA (open) contains questions scraped from quiz-league websites (Joshi et al., 2017). We use the open splits following Lee et al. (2019).
FreebaseQA is an Open-domain QA dataset derived from TriviaQA and other trivia resources (See Jiang et al. (2019) for full details). Every answer can be resolved to at least one Freebase entity and each question contains at least one entity.

3.2 Baselines

T5 (Raffel et al., 2019) and **BART** (Lewis et al., 2019) are large text-to-text transformers.
Dense Passage Retrieval (**DPR**) (Karpukhin et al., 2020) is a two stage retrieve and read model.
Retrieval Augmented Generation (**RAG**) (Lewis et al., 2020a) and Fusion in Decoder (**FID**) (Izacard and Grave, 2020) use DPR retrieval, followed by generative decoders based on BART and T5 respectively. FID is the current state-of-the-art on the open domain setting of TriviaQA.
K-Adapter (Wang et al., 2020a) and **Bert-KNN** (Kassner and Schütze, 2020) are recent BERT extensions that perform at or near state-of-the-art on the LAMA benchmark.
Entities-as-Experts (**EaE**) (Févry et al., 2020) is discussed in §2.3. Our EaE models are trained using the same hyperparameters and optimization settings as FILM.

3.2.1 Open vs Closed Book models

Generally, open book models refer to 'retrieve and read' pipelines (Chen et al., 2017) which, given a query, 1) retrieve relevant passages from a corpus, 2) separately re-encode the passages conditioned on the question and then 3) produce an answer. Conversely, closed book models answer questions directly from their parameters without additional processing of source materials. We consider FILM and EaE closed-book models as they do not retrieve and re-encode any source text, and instead attend to parameterized query-independent memories.

3.3 Results in Convention Settings

LAMA TREx. In Table 1, we can see that FILM outperforms several recently proposed models on the LAMA TREx task. FILM outperforms the next best performing model, BERT-KNN by 5.5 points.
Question-Answering. In Table 2, we compare FILM to five close-book and three open-book QA models on WebQuestionsSP and TriviaQA. The columns denoted Full Dataset-Total show results for the standard evaluation. For WebQuestionsSP, despite using far fewer parameters (see Table 3 and A.3 for details), FILM outperforms all other models - including the top open-book model RAG. On TriviaQA, FILM outperforms all other closed-book models—though the open-book models are substantially more accurate on this task, likely because of the enormous size of the models and their access to all of Wikipedia, which contains all (or nearly all) of the answers in TriviaQA.

3.4 Results on KB-Answerable Questions

WebQuestionsSP (and similarly FreebaseQA discussed in §4) was constructed such that all questions are answerable using the FreeBase KB, which was last updated in 2016. Because our pretraining corpus is derived from larger and more recent versions of Wikipedia, we elected to use a KB con-

[7] All data is English. See A.1 for additional details.

	Model	WebQuestionsSP				TriviaQA			
		Full Dataset		Wikidata Answer		Full Dataset		Wikidata Answer	
		Total	No Overlap	Total	No Overlap	Total	No Overlap	Total	No Overlap
Closed-book	**FILM**	**54.7**	**36.4**	**78.1**	**72.2**	**29.1**	**15.6**	**37.3**	**28.4**
	EaE	47.4	25.1	62.4	42.9	19.0	9.1	24.4	17.1
	T5-11B	49.7	31.8	61.0	48.5	–	–	–	–
	BART-Large	30.4	5.6	36.7	8.3	26.7	0.8	30.6	1.0
Open-Book	RAG	**50.1**	**30.7**	**62.5**	**45.1**	56.8	29.2	64.9	45.2
	DPR	48.6	34.1	56.9	45.1	57.9	31.6	66.3	48.8
	FID	–	–	–	–	**67.6**	**42.8**	**76.5**	**64.5**
	EmQL†	75.5	-	74.6	-	-	-	-	-

Table 2: **Open Domain QA Results**. Columns denoted *Full Dataset-Total* are conventional splits discussed in §3.3, *Wikidata Answer* are answerable using Wikidata (§3.4), and *No Overlap* removes train-test overlap (§3.5). Highest closed-book and open-book numbers are bolded. Other than FILM and EaE, all results are derived from the prediction files used in Lewis et al. (2020b) including the nearest neighbor (NN) baselines. †is a dataset specific graph reasoning model and the state-of-the-art WebQuestionSP.

Model	B	M	T
FILM	0.11	0.72	0.83
EaE	0.11	0.26	0.37
BERT-L	0.35	0	0.35
BART-L	0.39	0	0.39
T5-11B	11	0	11
DPR	0.11	16	16.11
RAG	0.39	16	16.39
FID	0.77	16	16.77

Table 3: **Model Parameters** Approximate billions of parameters for each model's (B)ase, (M)emories and (T)otal. Excludes token embeddings.

structed from Wikidata. Many entities in Freebase are unmappable to the more recent Wikidata KB which means that some questions are no longer answerable using the KB. Because of this, we created reduced versions of these datasets which are *Wikidata answerable*—i.e., containing only questions answerable by triples from our Wikidata-based KB. The model should learn to rely on the KB to answer the questions. We do the same for TriviaQA.[8]

As seen in Table 2 in the column Wikidata answer-Total, FILM does much better on Wikidata answerable questions on WebQuestionsSP. EmQL (Sun et al., 2020), the state-of-the-art dataset specific model, gets 75.5% accuracy on the full dataset. Not surprisingly, this is because EmQL operates over the Freebase knowledge base, giving it full upperbound recall. However, when we restrict to Wikidata answerable questions, thus giving both EmQL and FILM potential for full recall, FILM outperforms EmQL by 3.5 points and the next best model (RAG) by over 15 points.

[8]TriviaQA does not have linked entities in its questions so for those results we relax this restriction to include all examples where the answer resolves to a Wikidata entity.

3.5 Train-Test Overlap

We are interested in the ability of models to use external knowledge to answer questions, rather than learning to recognize paraphrases of semantically identical questions. Unfortunately, analysis showed that many of the test answers also appear as answers to some training-set question: this is the case for 57.5% of the answers in WebQuestionsSP and 75.0% for FreebaseQA. This raises the possibility that some of the performance can be attributed to simply memorizing specific question/answer pairs, perhaps in addition to recognizing paraphrases of the question from its pretraining data.

Overlap in fine-tuning train/test splits was concurrently observed by Lewis et al. (2020b), who created human verified filtered splits for TriviaQA and WebQuestions. We evaluate our models on those splits and report results in Table 2 in the "No Overlap" columns. We see that the gap between FILMand the next best performing model RAG increasses from 4.6 to 5.7 points on WebQuestionSP. On TriviaQA, FILMis still able to answer many questions correctly after overlap is removed. In contrast, the majority of closed book models such as BART get less than 1% of answers correct.

3.6 Filtering to Avoid Pretrain, Finetune, and Test Overlap

The filtering procedure from Lewis et al. (2020b) addresses finetuning train/test overlap but does not account for overlap with the pretraining data. To investigate this further, we looked at FreebaseQA and WebQuestionsSP which both contain entity linked questions and answers. We first perform a similar procedure to Lewis et al. (2020b) and

discard questions in the fine-tuning training data that contain answers which overlap with answers to questions in the dev and test data. We end up with 9144/2308/3996 data (train/dev/test) in FreebaseQA and 1348/151/1639 data in WebQuestionsSP. This setting is referred to as *Fine-tune* column in Table 4 which shows the effects of different filterings of the data.

Next we want to ensure that the model will be unable to simply memorize paraphrases of question answer pairs that it observed in the text by removing all overlap between the pretraining data and finetuning test data. For every question answer entity pair in our finetuning dataset (coming from any split), we filter every example from our Wikipedia pretraining corpus where those pair of entities co-occur. Additionally, we filter every fact from our fact memory containing any of these entity pairs. Results for this setting are in the column labeled *Pretrain*. The *All* column combines both pretrain and fine tune filtering. We see that the models perform substantially worse when these filterings are applied and they are forced to reason across multiple examples, and in the case of FILM, the fact memory. Finally, the column denoted None has no filtering and is the same as the Full Dataset.

4 Modifying the Knowledge Base

Because our model defines facts symbolically, it can in principle reason over new facts injected into its memory, *without retraining any parameters of the model*. Since existing datasets do not directly test this capability, we elected to construct variants of FreebaseQA and WebQuestionsSP where we could simulate asking questions that are answerable only from newly injected KB facts.

The approach we used was to (1) identify pairs of entities that occur in both a question and answer of some test example; (2) filter out such pairs from the KB as well as all pre-training and fine-tuning data; and (3) test the system trained on this filtered data, and then manually updated by injecting facts about those entity pairs. This filtering procedure is reminiscent of that used by Lewis et al. (2020b), but also addresses pretraining / test-set overlap.

4.1 Injecting New Facts to Update Memory

We evaluate EaE and FILM given full knowledge (the original setting); given filtered knowledge; and given filtered knowledge followed by injecting test-question-related facts into the KB. The gap between the filtered knowledge setting and injected knowledge setting will indicate how well the model incorporates newly introduced facts.

In more detail, we first perform a similar procedure to Lewis et al. (2020b) and discard questions in the fine-tuning training data that contain answers which overlap with answers to questions in the dev and test data. We end up with 9144/2308/3996 data (train/dev/test) in FreebaseQA and 1348/151/1639 data in WebQuestionsSP. Next, to ensure that the model will be unable to memorize paraphrases of question-answer pairs that it observed in the pretraining text, we remove all overlap between the pretraining data and fine-tuning test data: specifically, for every question-answer entity pair in our fine-tuning dataset (from any split), we filter every example from our Wikipedia pretraining corpus in which that pair of entities co-occur. Additionally, we filter every fact from our fact memory containing any of these entity pairs.

In these sections we compare against EaE for two reasons: 1) we are specifically looking at closed-book open domain entity based QA and EaE is shown to be at or near state-of-the-art for that task (Févry et al., 2020), 2) most importantly, we want to be able to precisely control for memorization in the training corpus and therefore did not consider existing unconstrained pre-trained models like T5 (Raffel et al., 2019). For reference, the previous state-of-the-art FOFE (Jiang et al., 2019) on FreebaseQA had a score of 37.0% using the original train-test split, while FILM is at 63.3%.

The results are shown in Table 5. In the "Full" column, we pretrain and finetune the FILM model with the full knowledge base and corpus. In the "Filter" setting, facts about the finetuning data are hidden from the model at both pretraining and fine-tuning time. In this case, the model must fall back to the language model to predict the answer, and as shown in Table 5, the accuracies of FILM and EaE are similar. In the "Inject Facts" setting, Facts are hidden at pretraining time, but are injected at test time. The results show that FILM can effectively use the newly injected facts to make prediction, obtaining an absolute improvement of 9.3% compared to the "Filter" setting. EaE does not have a natural mechanism for integrating this new information[9].

[9]There are various heuristics one could apply for finetuning a standard language model on this type of data by applying one or a small number of gradient steps on textualized facts. We leave this exploration for future research.

	FreebaseQA				WebQuestionsSP			
Filter Type	None	Pretrain	Fine-tune	All	None	Pretrain	Fine-tune	All
EaE	53.4	45.2	45.8	28.6	48.1	45.4	30.9	29.4
FILM	63.3	57.5	56.5	48.0	56.1	55.4	40.7	39.2

Table 4: **Effects of Different Data Filtering**. The column denoted *None* has no filtering. *Pretrain* removes all entity pair overlap between the eval datasets (all splits) and the pretraining text and kb. The *Fine-tune* column removes all entity pair overlap between the eval train and test splits. The *All* column combines both pretrain and fine tune filtering.

	FreebaseQA			WebQuestionsSP		
	Full	Filter	Inject	Full	Filter	Inject
EaE	45.8	28.6	-	30.9	29.4	-
FILM	56.5	38.7	48.0	40.7	32.3	39.2

Table 5: **Injecting New Facts**. In the Filter setting, the models have access to no direct knowledge about question answer entity pairs from either the pretraining corpus or KB. In the Inject setting, the pretraining corpus and training KB are still Filtered, but at inference time, new facts are injected into the models memory allowing it to recover most of the drop from the Full setting. In the Full setting the model is exposed to full knowledge. In all cases, we remove the overlap between the finetune train and eval sets.

4.2 Updating Stale Memories

One of the main motivations for our model is to provide knowledge representations that can be incrementally updated as the world changes, avoiding stale data. In order to accomplish this, the model must learn to utilize the fact memory even in the case where those facts have changed such that they may no longer be consistent with the data the model was initially trained on. Further, it needs to accomplish that without any additional training.

To probe this ability, we simulate an extreme version of stale facts where all answers to QA pairs in the FreebaseQA test set are 'updated' with plausible alternatives. For each QA pair, we replace the original answer entity e_{original} with another entity, e_{new}, from our vocabulary that has: 1) been used as an object in at least one of the same relation types in which e_{original} was used as an object, and 2) shares at least three Wikipedia categories with e_{original}.

We use the same pretrained models from our earlier experiments and fine-tune on the filtered FreebaseQA train set for 10,000 steps. We then modify the memory of this model without applying any additional training on the new memory. In addition to adding new memories which correspond to our newly created facts, we also must remove the original stale facts that we are updating. We look at two methods for filtering those 'stale facts' from the fact memory.

Basic Filter deletes every modified fact e_{question}, r, e_{original} and replaces it with a new fact e_{question}, r, e_{new}. This would be a low recall filter as it does not account for all possible related facts. The *Strict Filter* is a high recall filter that more aggressively removes information that may conflict with the newly added fact, additionally removing all facts that contain e_{question} or e_{original}. This is important for cases such as when a question contains multiple entities, or the linking relation is one-to-many, leading to multiple plausible answers. Together these two settings define rough bounds on the model's ability to perform this task. In Table 6, we see that FILM is able to utilize the modified KB to make the correct prediction for **54.5%** of questions in the *Basic Filter* setting and **70.3%** in the *Strict Filter* setting.

Model	Basic Filter	Strict Filter
FILM	0.0	0.0
+Update Memory	54.5	70.3

Table 6: **Updating Stale Memories**. Basic filter removes only facts connecting the original question entity to the answer entity. Strict filter removes all facts containing the original question or answer (not just facts connecting them).

5 Related Work

Symbolic KBs have been a core component of AI since the beginning of the field (Newell and Simon, 1956; Newell et al., 1959), and widely available public KBs have been invaluable in research and industry (Bollacker et al., 2008; Auer et al., 2007; Google, 2012; Dong, 2017; Vrandečić and Krötzsch, 2014). In machine learning, a well studied problem is learning KB embeddings (Bordes et al., 2013; Lin et al., 2015; Trouillon et al., 2017;

Dettmers et al., 2018) which enable generalization from known KB triples to novel triples that are plausibly true. KB embeddings can often be improved by incorporating raw text and symbolic KGs into a shared embedding space (Riedel et al., 2013; Verga et al., 2016, 2017), to be jointly reasoned over (Sun et al., 2018, 2019). Many prior neural-symbolic methods have attempted to unify symbolic KBs and neural methods (Pinkas, 1991; de Penning et al., 2011; Laird et al., 2017; Besold et al., 2017). Recently, researchers have explored query languages for embedded KBs that are similar to symbolic KB query languages (Cohen et al., 2017; Hamilton et al., 2018; Ren et al., 2020; Cohen et al., 2020).

Our fact memory builds on this prior work, and is most closely related to the memory used in EmQL (Sun et al., 2020), one KB embedding model that supports compositional query language. EmQL implements "projection" using neural retrieval over vectorized KB triples. Unlike this work, however, EmQL did not embed its fact memory into a LM, which could be finetuned for many NLP tasks: instead requiring the implementation of a "neural module" into some task-specific architecture. At a more abstract level, the fact memory is a key-value memory (Weston et al., 2014; Miller et al., 2016), a construct used in many neural models in the past.

It has been shown that sufficiently large LMs trained through self supervision (Peters et al., 2018; Devlin et al., 2019; Raffel et al., 2019; Brown et al., 2020) also encode factual information, motivating work on the extent to which a LM can serve as a KB (Roberts et al., 2020; Petroni et al., 2019; Poerner et al., 2019). Other work has explored techniques to improve the performance of large LMs in answering factual probes, by adding additional supervision in pre-training (Xiong et al., 2019; Wang et al., 2020b) or by adding entity embeddings into an extended LM (Peters et al., 2019; Zhang et al., 2019; Févry et al., 2020).

Our entity memory extends the Entities-as-Experts (EaE) model (Févry et al., 2020). It is both the current state-of-the-art for a number of tasks and simpler to use than most prior models because it does not require external components for entity linking or entity encoding (like (Peters et al., 2019; Zhang et al., 2019; Logan et al., 2019)) and is not restricted to lexical KBs like WordNet and ConceptNet (like (Weissenborn et al., 2017; Chen et al., 2018; Mihaylov and Frank, 2018)).

Our model's use of memory also scales to KBs with millions of entities, whereas prior systems that make use of 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 and Frank, 2018; Logan et al., 2019).

There have been a few exploratory experiments on modifying the predictions of retrieval augmented language models by changing the underlying text corpus (Guu et al., 2020; Lewis et al., 2020a). However, text passages are not easily interpretable resulting in them being less inspectible and modifiable than a symbolic fact based memory.

6 Conclusion

We presented FILM, a neural LM with an interpretable symbolically bound fact memory. We demonstrated the effectiveness of this method by outperforming many state-of-the-art methods on four benchmark knowledge intensive datasets. We used the model's symbolic interface to change the output of the LM by modifying only the non-parametric memories, without any additional training. We showed FILM could incorporate newly injected facts unseen during training. Additionally, we can modify facts, such that they contradict the initial pre training text, and our model is still largely able to answer these questions correctly.

7 Ethics and Broader Impacts

All language models learn to exploit correlations in the data they were trained on. As such, they inherit all of the underlying biases within that data (Zhao et al., 2019; Bender et al., 2021). These models require vast amounts of data to train on and therefore tend to rely on internet corpora which have skewed representations of particular groups, cultures, and languages, as well as variable levels of factuality. Our hope is that research into endowing these models with interpretable and modifiable memories will allow us to more readily identify and remedy some of these failures.

Acknowledgments

We thank the anonymous reviewers for their helpful feedback and Patrick Lewis for providing prediction files. We also thank Thibault Févry, Nicholas FitzGerald, Eunsol Choi, Tom Kwiatkowski and other members of Google Research for discussions and feedback.

References

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *arXiv preprint arXiv:1608.00318*.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big?. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Tarek R Besold, Artur d'Avila Garcez, Sebastian Bader, Howard Bowman, Pedro Domingos, Pascal Hitzler, Kai-Uwe Kühnberger, Luis C Lamb, Daniel Lowd, Priscila Machado Vieira Lima, et al. 2017. Neural-symbolic learning and reasoning: A survey and interpretation. *arXiv preprint arXiv:1711.03902*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

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

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei. 2018. 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)*, pages 2406–2417.

William W Cohen, Haitian Sun, R Alex Hofer, and Matthew Siegler. 2020. Scalable neural methods for reasoning with a symbolic knowledge base. *International Conference on Learning Representations*.

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

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Luna Dong. 2017. Amazon product graph.

Thibault Févry, Livio Baldini Soares, Nicholas FitzGerald, Eunsol Choi, and Tom Kwiatkowski. 2020. Entities as experts: Sparse memory access with entity supervision. *Conference on Empirical Methods in Natural Language Processing*.

Google. 2012. Introducing the knowledge graph: things, not strings.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*.

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

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2016. Tracking the world state with recurrent entity networks. *arXiv preprint arXiv:1612.03969*.

Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*.

Kelvin Jiang, Dekun Wu, and Hui Jiang. 2019. Freebaseqa: A new factoid qa data set matching trivia-style question-answer pairs with freebase. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 318–323.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly

supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wentau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Nora Kassner and Hinrich Schütze. 2020. Bert-knn: Adding a knn search component to pretrained language models for better qa. *arXiv preprint arXiv:2005.00766*.

John E Laird, Christian Lebiere, and Paul S Rosenbloom. 2017. A standard model of the mind: Toward a common computational framework across artificial intelligence, cognitive science, neuroscience, and robotics. *Ai Magazine*, 38(4):13–26.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020a. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*.

Patrick Lewis, Pontus Stenetorp, and Sebastian Riedel. 2020b. Question and answer test-train overlap in open-domain question answering datasets. *arXiv preprint arXiv:2008.02637*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*.

Robert Logan, Nelson F. Liu, Matthew E. Peters, Matt Gardner, and Sameer Singh. 2019. Barack's wife hillary: Using knowledge graphs for fact-aware language modeling. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Todor Mihaylov and Anette Frank. 2018. 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)*, pages 821–832, Melbourne, Australia. Association for Computational Linguistics.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409.

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

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

H Leo H de Penning, Artur S d'Avila Garcez, Luís C Lamb, and John-Jules C Meyer. 2011. A neural-symbolic cognitive agent for online learning and reasoning. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.

Gadi Pinkas. 1991. Symmetric neural networks and propositional logic satisfiability. *Neural Computation*, 3(2):282–291.

Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2019. Bert is not a knowledge base (yet): Factual knowledge vs. name-based reasoning in unsupervised qa. *arXiv preprint arXiv:1911.03681*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Hongyu Ren, Weihua Hu, and Jure Leskovec. 2020. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. *International Conference on Learning Representations*.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*.

Haitian Sun, Andrew O Arnold, Tania Bedrax-Weiss, Fernando Pereira, and William W Cohen. 2020. Guessing what's plausible but remembering what's true: Accurate neural reasoning for question-answering. *Advances in neural information processing systems*.

Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 2019. 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)*, pages 2380–2390.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. 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*, pages 4231–4242.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2016. Multilingual relation extraction using compositional universal schema. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 886–896.

Patrick Verga, Arvind Neelakantan, and Andrew McCallum. 2017. Generalizing to unseen entities and entity pairs with row-less universal schema. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 613–622.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Cuihong Cao, Daxin Jiang, Ming Zhou, et al. 2020a. K-adapter: Infusing knowledge into pre-trained models with adapters. *arXiv preprint arXiv:2002.01808*.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2020b. K-adapter: Infusing knowledge into pre-trained models with adapters.

Dirk Weissenborn, Tomáš Kočiskỳ, and Chris Dyer. 2017. Dynamic integration of background knowledge in neural nlu systems.

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

Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. 2019. Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model. *arXiv preprint arXiv:1912.09637*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. 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)*, pages 1321–1331, Beijing, China. Association for Computational Linguistics.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. Ernie: Enhanced language representation with informative entities. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Ryan Cotterell, Vicente Ordonez, and Kai-Wei Chang. 2019. Gender bias in contextualized word embeddings. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 629–634, Minneapolis, Minnesota. Association for Computational Linguistics.

A Appendix

A.1 Data

A.1.1 Evaluation Data Statistics

For WebQuestionsSP, we mapped question entities and answer entities to their Wikidata ids. 87.9% of the questions are answerable by at least one answer entity that is mappable to Wikidata. For all questions in FreebaseQA there exists at least one relational path in Freebase between the question entity e_i and the answer e_{ans}. The path must be either a one-hop path, or a two-hop path passing through a mediator (CVT) node, and is verified by human raters. 72% of the question entities and 80% of the answer entities are mappable to Wikidata, and 91.7% of the questions are answerable by at least one answer entity that is mappable to Wikidata.

		Full Dataset	Wikidata Answerable
FreebaseQA	Train	20358	12535
	Dev	3994	2464
	Test	3996	2440
WebQuestionsSP	Train	2798	1388
	Dev	300	153
	Test	1639	841

Table 7: **Dataset stats.** Number of examples in train, dev, and test splits for our three different experimental setups. Full are the original unaltered datasets. Wikidata Answerable keeps only examples where at least one question entity and answer entity are mappable to Wikidata and there is at least one fact between them in our set of facts.

A.1.2 Pretraining Data Details

FILM is pretrained on Wikipedia and Wikidata using the same data from Févry et al. (2020). Text in Wikipedia is chunked into 128 token pieces. To compute the entity-linking loss loss_{ent}, we use as training data entities linked to the 1 million most frequently linked-to Wikidata entities. Text pieces without such entities are dropped. This results in 30.58 million text pieces from Wikipedia. As described in §2.1, we generate n training examples from a piece of text containing n entity mentions, where each mention serves as the masked target for its corresponding example, and other entity mentions in the example are treated as context entities[10]. This conversion results in 85.58 million pre-training examples. The knowledge base $\mathcal{K}$ is a subset of Wikidata that contains all facts with subject and object entity pairs that co-occur at least 10 times on Wikipedia pages.[11] This results in a KB containing 1.54 million KB triples from Wikidata (or 3.08 million if reverse triples are included). Below, this is called the *full setting* of pretraining—we will also train on subsets of this example set, as described below. We pretrain the model for 500,000 steps with the batch size 2048, and we set $k = 1$[12] in the TOP_k operation for fact memory access.

A.2 Training Details

A.2.1 Pretraining

FILM is jointly trained to predict context entities and the masked entity. Context entities are predicted using the contextual embeddings described in §2.3; intermediate supervision with oracle entity linking labels is provided in the entity memory access step for context entities; the masked entity is predicted using the knowledge-enhanced contextual embeddings (§2.5); and distant supervised fact labels are also provided at training time. The final training loss is the unweighted sum of the four losses:

$$\text{loss}_{\text{pretrain}} = \text{loss}_{\text{ent}} + \text{loss}_{\text{ctx}} + \text{loss}_{\text{fact}} + \text{loss}_{\text{ans}}$$

A.2.2 Finetuning on Question Answering

In the Open-domain Question Answering task, questions are posed in natural language, e.g. "Where was Charles Darwin born?", and answered by a sequence of tokens, e.g. "United Kingdom". In this paper, we focus on a subset of open-domain questions that are answerable using entities from a knowledge base. In the example above, the answer "United Kingdom" is an entity in Wikidata whose identity is Q145.

We convert an open-domain question to an input of FILM by appending the special [MASK] token to the end of the question, e.g. {'Where', 'was', 'Charles', 'Darwin', 'born', '?', [MASK]}. The task is to predict the entity named by mask. Here, "Charles Darwin" is a context entity, which is also referred to as *question entity* in the finetuning QA task.

At finetuning time, entity embeddings $\mathbf{E}$ and relation embeddings $\mathbf{R}$ are fixed, and we finetune

[10] We mask context entities randomly with probability .15

[11] This leads to more KB triples than entity pairs, since a pair of entities can be connected by more than one relation.

[12] We experimented with other values of k during fine tuning and evaluation but did not observe significant differences.

all transformer layers and the four transformation matrices: $\mathbf{W}_a$, $\mathbf{W}_b$, $\mathbf{W}_e$, $\mathbf{W}_f$. Parameters are tuned to optimize unweighted sum of the the fact memory retrieval loss $\text{loss}_{\text{fact}}$ and the final answer prediction loss loss_{ans}. If multiple answers are available, the training label $\mathbb{I}_{e_{\text{ans}}}$ becomes a k-hot vector uniformly normalized across the answers.

$$\text{loss}_{\text{finetune}} = \text{loss}_{\text{fact}} + \text{loss}_{\text{ans}}$$

A.3 Model Parameters

The number of Base parameters includes the encoder and (where applicable) decoder transformer parameters derived from the original papers. We exclude token embeddings in this count following prior work. The Memory parameter count for DPR, RAG, and FID includes the number of parameters required to cache and index the full 26 million passage wikipedia corpus with dimension 768 used by those models. For EaE, the Memory is for the entity embedding matrix. For FILM it is both the entity embedding matrix and the cached fact embedding matrix comprised of the 1.7 million precomputed triple embeddings.