Entities as Experts: Sparse Memory Access with Entity Supervision

Thibault Févry [*] tfevry@google.com
Livio Baldini Soares liviobs@google.com
Nicholas Fitzgerald nfitz@google.com
Eunsol Choi eunsolc@google.com
Tom Kwiatkowski tomkwiat@google.com

Google Research

Abstract

We focus on the problem of capturing declarative knowledge in the learned parameters of a language model. We introduce a new model—Entities as Experts (EAE)—that can access distinct memories of the entities mentioned in a piece of text. Unlike previous efforts to integrate entity knowledge into sequence models, EAE's entity representations are learned directly from text. These representations capture sufficient knowledge to answer TriviaQA questions such as *"Which Dr. Who villain has been played by Roger Delgado, Anthony Ainley, Eric Roberts?"*. EAE outperforms a Transformer model with 30× the parameters on this task. According to the LAMA knowledge probes, EAE also contains more factual knowledge than a similar sized BERT. We show that associating parameters with specific entities means that EAE only needs to access a fraction of its parameters at inference time, and we show that the correct identification, and representation, of entities is essential to EAE's performance. We also argue that the discrete and independent entity representations in EAE make it more modular and interpretable than the Transformer architecture on which it is based.

1 Introduction

Neural network sequence models, pre-trained as language models, are currently a standard component in state-of-the-art text understanding (Dai and Le, 2015; Peters et al., 2018; Howard and Ruder, 2018; Devlin et al., 2018). Recent work has shown that these models capture some declarative knowledge, and has suggested that neural language models could take the place of curated knowledge bases or textual corpora for tasks such as question answering (Petroni et al., 2019; Roberts et al., 2020).

[*]Work done during Google AI residency.

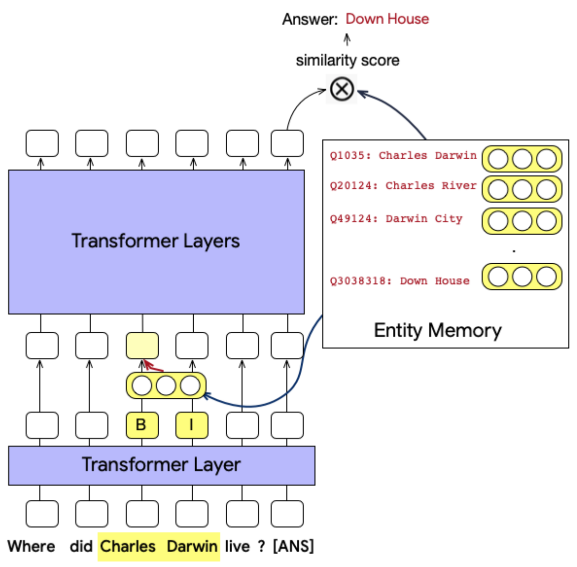


Figure 1: The Entities as Experts model with an entity memory, applied to the open domain question answering task. The red arrows shows the integration of the entity embedding representations in the token representations.

In this paper, we focus on developing neural sequence models that capture the knowledge required to answer questions about the world. To this end, we introduce a new model architecture that can access distinct and independent representations of the entities mentioned in text. Unlike other efforts to inject entity specific knowledge into sequence models (Peters et al., 2019; Zhang et al., 2019; Poerner et al., 2019) our model does not rely on an external knowledge base for its entity representations, and instead learns them directly from text along with all the other model parameters. We call our model Entities as Experts (EAE). This name reflects EAE's similarities with the Massive Mixture of Experts (Shazeer et al., 2017) and other work that integrates learned memory stores into sequence models (Weston et al., 2014; Lample et al., 2019).

It also highlights the fact that EAE's memory store represents each entity distinctly and independently.

To understand the motivation for distinct and independent entity representations, consider Figure 1. A traditional Transformer (Vaswani et al., 2017) would need to build an internal representation of Charles Darwin from the words "Charles" and "Darwin", both of which can also be used in reference to very different entities such as the Charles River, or Darwin City. Conversely, EAE can access a dedicated representation of "Charles Darwin", which is a memory of all of the contexts in which this entity has previously been mentioned. This representation can also be accessed for other mentions of Darwin, such as "Charles Robert Darwin" or "the father of natural selection". Having retrieved and re-integrated this memory it is much easier for EAE to relate the question to the answer.

We train EAE to predict masked out spans in English Wikipedia text (Devlin et al., 2018). We also train EAE to only access memories for entity mentions, and to access the correct memories for each entity mention. Supervision for the memory access comes from an existing mention detector, and Wikipedia hyperlinks. By associating memories with specific entities, EAE can learn to access them sparsely. The memory is only accessed for spans that mention entities. Furthermore, only the memories associated with the entities mentioned need to be retrieved.

We evaluate EAE's ability to capture declarative knowledge using the cloze knowledge probes introduced by Petroni et al. (2019), as well as the TriviaQA (Joshi et al., 2017) question answering task used by Roberts et al. (2020). In contrast to tasks that focus on text representations (Rajpurkar et al., 2016; Wang et al.), knowledge probing tasks test a model's ability to predict the text that completes a fact. On these tasks, EAE outperforms related approaches. Training EAE to focus on entities is better than a similar-sized network with an unconstrained memory store. EAE also outperforms the much larger model from Roberts et al. (2020) on TriviaQA, and it does so by only accessing a small proportion of its parameters at inference time.

Finally, we present an in-depth analysis of EAE's predictions on TriviaQA. This analysis shows that the correct identification and reintegration of entity representations is essential for EAE's performance. With this analysis, we also argue that our entity memory layers have led to a model that is more modular and interpretable than previous sequence models that have no discrete representations beyond word-piece tokens.

2 Approach

2.1 Definition

Let $\mathcal{E} = \{e_1 \dots e_N\}$ be a predefined set of entities, and let $\mathcal{V} = \{\texttt{[MASK]}, w_1 \dots w_M\}$ be a vocabulary of tokens. A *context* $\mathbf{x} = [x_0 \dots x_L]$ is a sequence of tokens $x_i \in \mathcal{V}$. Each context comes with the list of the *mentions* it contains, $\mathbf{m} = [m_0 \dots m_M]$, where each mention $m_i = (e_{m_i}, s_{m_i}, t_{m_i})$ is defined by its linked entity e_{m_i}, start token index s_{m_i} and end token index t_{m_i}. Entity mention might not be linked to the entity vocabulary $\mathcal{E}$, thus $e_{m_i} \in \mathcal{E} \cup e_\varnothing$, where $e_\varnothing$ refers to the null entity.

2.2 Model Architecture

The basic model architecture follows the Transformer (Vaswani et al., 2017), interleaved with our entity memory layer. Our model has two embedding matrices – token and entity embeddings. Our model is:

$$\begin{aligned}
\mathbf{X}^0 &= \texttt{TokenEmbed}(\mathbf{x}) \\
\mathbf{X}^1 &= \texttt{Transformer}(\mathbf{X}^0, \texttt{num\_layers} = l_0) \\
\mathbf{X}^2 &= \texttt{EntityMemory}(\mathbf{X}^1) \\
\mathbf{X}^3 &= \texttt{LayerNorm}(\mathbf{X}^2 + \mathbf{X}^1) \\
\mathbf{X}^4 &= \texttt{Transformer}(\mathbf{X}^3, \texttt{num\_layers} = l_1) \\
\mathbf{X}^5 &= \texttt{TaskSpecificHeads}(\mathbf{X}^4)
\end{aligned}$$

The entity memory layer constructs an entity embedding E_{m_i} for each mention m_i. The output of the entity memory layer and preceding transformer layer are summed and normalized, then processed by additional transformer layers. Figure 2 illustrates our model.

Entity Memory Layer Let $\mathbf{E}$ be a matrix of learned entity embeddings of shape (N, d_{ent}). $\texttt{EntEmbed}(e_i)$ maps an entity e_i to its row in $\mathbf{E}$. The entity memory layer takes the output sequence from the preceding transformer layer ($\mathbf{X}^l$) and outputs a new sequence ($\mathbf{X}^{l+1}$), sparsely populated with entity representations. For each entity mention m_i, the output sequence has an entity representation, a projection of the weighted sum of entity embeddings in $\mathbf{E}$, at position s_{m_i}.

$$x_i^{l+1} = \mathbf{W}_b E_{m_k} \quad \texttt{if } i = s_{m_k} \tag{1}$$

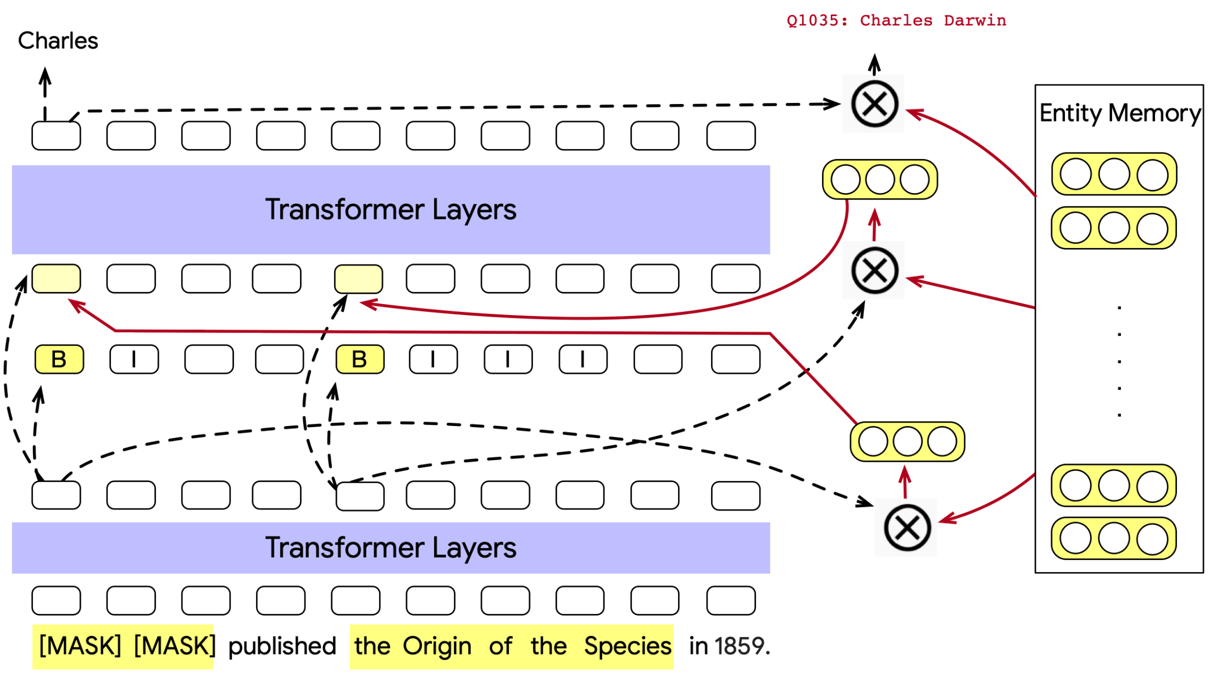


Figure 2: The **E**ntities **as E**xperts model: the initial transformer layer output is used (i) to predict mention boundaries, (ii) to retrieve entity embeddings from entity memory, and (iii) to construct input to the next transformer layer, augmented with the retrieved entity embeddings of (ii). The final transformer block output is connected to task specific heads: token prediction and entity prediction. The entity retrieval after the first transformer layer (ii) is also supervised with an entity linking objective during pre-training.

where $\mathbf{W}_b$ maps the entity representation E_{m_k} to the dimension of x_i^l.

We now describe how to generate E_{m_i} for each mention m_i. First, we generate a pseudo entity embedding h_{m_i} based on the mention's span representation $[x_{s_{m_i}}^l || x_{t_{m_i}}^l]$, a concatenation of its start and tail representations.

$$h_{m_i} = \mathbf{W_f}[x_{s_{m_i}}^l || x_{t_{m_i}}^l] \qquad (2)$$

where $\mathbf{W_f}$ is of shape $(d_{ent}, 2 \cdot d_{emb})$, where d_{emb} is the dimension of $\mathbf{X^1}$.

KNN

We find the k nearest entity embeddings of h_{m_i} from $\mathbf{E}$ by computing the dot product, and E_{m_i} is a weighted sum of them. More formally:

$$E_{m_i} = \sum_{e_j \in \texttt{topK}(\mathcal{E}, h_{m_i}, k)} \alpha_j \cdot (\texttt{EntEmbed}(e_j))$$

$$\alpha_j = \frac{\exp(\texttt{EntEmbed}(e_j) \cdot h_{m_i})}{\sum_{e \in \texttt{topK}(\mathcal{E}, h_{m_i}, k)} \exp(\texttt{EntEmbed}(e) \cdot h_{m_i})}$$

Where $\texttt{topK}(\mathcal{E}, h_{m_i}, k)$ returns the k entities that yield the highest score $\texttt{EntEmbed}(e_j) \cdot h_{m_i}$.

When $k = 1$, E_{m_i} is the nearest entity embedding to h_{m_i} (argmax), when $k = N$, it is a weighted sum of all entity embeddings. The entity memory layer can be applied to any sequence output without loss of generality. We describe it applied to the output of the first Transformer.

Task-Specific Heads The final transformer layer can be connected to multiple task specific heads. In our experiments, we introduce two heads: `TokenPrediction` and `EntityPrediction`.

The `TokenPrediction` head predicts masked tokens for a cloze task. Each masked token's final representation x_i^4 is fed to an output softmax over the token vocabulary, as in BERT.

The `EntityPrediction` head predicts entity id for each entity mention span (i.e., entity linking). We build the pseudo entity embedding (h_{m_i}) from the last sequence output (X^4), as described in Entity Memory Layer. Then, the model predicts the entity whose embedding in $\mathbf{E}$ is the closest to the pseudo entity embedding.

Inference-time Mention Detection In Section 4 we introduce a number of tasks in which the input is raw text, without mention boundaries. We introduce a mention detection layer to avoid dependence at inference on an external mention detector for these tasks.

The mention detection layer applies a three-way

BIO[1] classifier to the first transformer block's output. We decode the entire BIO sequence directly, ensuring that inconsistent sequences are disallowed.

2.3 Training

2.3.1 Data and Preprocessing

We assume access to a corpus $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{m_i})\}$, where all entity mentions are detected but not necessarily all linked to an entity. We require our mentions to be non-overlapping so the BIO-tagged sequence of tokens is well-defined. In practice, we use Wikipedia. Entity mentions come from hyperlinks, which provide sparse mention boundaries and entity labels, and the Google Cloud Natural Language API[2] , which provides all non-hyperlink mention boundaries.

We pre-process this data by removing 20% of randomly chosen entity mentions (all tokens in the mention boundaries are replaced with [MASK]) to support a masked language modeling objective. Contiguous masking was explored in Joshi et al. 2020 and masking mention spans specifically is similar to the salient span masking used in Guu et al. 2020.

2.3.2 Learning Objective

The pre-training objective is the sum of (1) an entity linking loss, (2) a mention boundary detection loss, and (3) a masked language modeling loss.

Entity Linking We use the hyperlinked entities e_{m_i} to supervise entity memory assess. For each hyperlinked mention $m_i = (e_{m_i}, s_{m_i}, t_{m_i})$, where $e_i \neq e_\varnothing$, the pseudo embedding h_{m_i} (Equation. 2) should be close to the entity embedding of the annotated entity, $\texttt{EntEmbed}(e_{m_i})$.

$$\texttt{ELLoss} = \sum_{m_i} \alpha_i \cdot \mathbb{1}_{e_{m_i} \neq e_\varnothing}$$

$$\alpha_i = \frac{\exp(\texttt{EntEmbed}(e_{m_i}) \cdot h_{m_i})}{\sum_{e \in \mathcal{E}} \exp(\texttt{EntEmbed}(e) \cdot h_{m_i})}$$

Note that this loss is not applied to the vast majority of mentions, which do not have hyperlinks. Memory access for those mentions is unsupervised. For tasks that have an `EntityPrediction` head, the same loss is used to train that head.

[1]In a BIO encoding, each token is classified as being the Beginning, Inside, or Outside of a mention.

[2]https://cloud.google.com/natural-language/docs/basics#entity_analysis

Mention Detection The three-way BIO classification of tokens is supervised with a cross-entropy loss over the labels. We assume that the mention boundaries in $\mathcal{D}$ are complete, and apply this supervision to all tokens.

Masked Language Modelling We follow existing work in masked language modeling (Devlin et al., 2018), and train the `TokenPrediction` head to independently predict each of the masked out tokens in an input context. We train this prediction by applying a cross entropy loss to the distribution over $\mathcal{V}$ that is returned by the `TokenPrediction` head.

3 Related Work

Knowledge-augmented language models Our work shares inspiration with other approaches seeking to inject knowledge into language models (Ahn et al., 2016; Yang et al., 2016; Logan et al., 2019; Zhang et al., 2019; Levine et al., 2019; Xiong et al., 2019a; Peters et al., 2019; Poerner et al., 2019; Wang et al., 2020). These have used a variety of knowledge sources (WikiData, WordNet relations, outputs from dependency parsers) and additional training objectives (synonym and hyponym-hypernym prediction, word-supersense prediction, replaced entity detection, predication prediction, dependency relation prediction, entity linking).[3] Our focus is mostly on adding knowledge about entities, so our work is closer to Zhang et al. (2019); Peters et al. (2019); Xiong et al. (2019b); Wang et al. (2020); Poerner et al. (2019) than to the linguistically-augmented approaches of Levine et al. (2019); Lauscher et al. (2019). In particular, Peters et al. (2019) introduce an entity memory layer that is similar to the one in EAE. However, this prior work focuses on integrating entity-specific knowledge from external sources (Peters et al. (2019) represent entities using graph embeddings of WikiData and WordNet), while we focus on learning entity representations from text.

Memory Augmented Neural Networks Our entity memory layer is closely tied to memory-based neural layers (Weston et al., 2014; Sukhbaatar et al., 2015). In particular, it can be seen as a memory network where memory access is supervised through entity linking, and memory slots each correspond to a learned entity representation. When uncon-

[3]See also Table 1 of Wang et al. (2020) for a useful review of such approaches.

strained, these memory networks can be computationally expensive and supervising access through entity linking limits this issue. Another approach to scale memory networks is given by Lample et al. (2019) who introduce product-key memories to efficiently index a large store of values.

Conditional Computation Conditional computation models seek to increase model capacity without a proportional increase in computational complexity. This is usually done through routing, where only a subset of the network is used to process each input. To facilitate such routing, different approaches have been developed, including large mixture of experts (Shazeer et al., 2017) or gating (Eigen et al., 2013; Cho and Bengio, 2014). Our method uses entities as experts, using several experts per sentence. It uses mention detection as a form of gating to limit expert access.

4 Knowledge Probing Tasks

We follow previous work in using cloze tests and question answering tasks to quantify the declarative knowledge captured in the parameters of our model (Petroni et al., 2019; Roberts et al., 2020).

In cloze tests, a model must recover the words in a blanked out mention by correctly associating the mention with its surrounding sentence context. We report performance on the recently introduced LAMA knowledge probe task (Petroni et al., 2019), and on our model's ability to recover the textual surface form and entity identity of a blanked out Wikipedia hyperlink.

Roberts et al. (2020) show that a very large transformer model can correctly answer a number of open-domain questions, and surmise that the model must therefore be capturing a significant amount of real-world knowledge. We follow their evaluation setup, and report results on the entity-centric TriviaQA question answering task.

4.1 Predicting Wikipedia Hyperlinks

We explore the ability of our model to predict entities and their text when they are masked. This is the training task described in Section 2.3, and we create development and test sets that have the same form as our training data. None of the text in the evaluation sets is included in the training data, and 25% of the hyperlinked mentions in these evaluation sets have been masked out.[4]

[4]This is slightly different from masking 20% of all mentions, as we do for training.

4.2 LAMA

LAMA (Petroni et al., 2019) contains cloze tasks from three different knowledge base sources, and one QA dataset. LAMA aims to probe the knowledge contained in a language model, with a focus on the type of knowledge that has traditionally been manually encoded in knowledge bases. As an evaluation-only probing task, it does not involve any task specific model training, and we do not apply any LAMA specific modeling enhancements. As a result, EAE is not very well suited to the LAMA ConceptNet sub-task, which include non-entity answers such as “fly”, “cry”, and “happy”. For Google-RE, a third of the answers are dates, which we also do not predict well. Nonetheless, we include these sub-tasks for completeness.

4.3 Open domain QA

TriviaQA (Joshi et al., 2017) was introduced as a reading comprehension task, in which questions are paired with documents. More recently, the questions and answers from this dataset have been used to assess the performance of question answering systems in the open domain setting where no evidence documents are given. Most approaches (Lee et al., 2019; Min et al., 2019b,a) have relied on a text corpus at inference time, retrieving passages that match the question, and then reading these passages to extract an answer. However, Roberts et al. (2020) has applied a language model (henceforth, T5) directly to this task. T5 simply reads the question, and then outputs an answer string.

The TriviaQA data contains (question, answer) pairs q_i, e_{a_i}.[5] Then, the task is to predict $e_{a_i} \in \mathcal{E}$ from q_i. Unlike the cloze tasks described above, we tune EAE's parameters on the TriviaQA training set. The fine-tuning objective has only the mention detection loss and a single `EntityPrediction` head loss component: to predict the answer entity e_{a_i}, from the single masked token concatenated at the end of the question.

5 Models Evaluated

In this section we introduce all models, new and from prior work, that are compared in Section 7.

[5]TriviaQA answers tend to be common entities. Answering open domain questions with free-form text, as is required for Natural Questions (Kwiatkowski et al., 2019), is out of the scope of this current paper.

5.1 Models with Entity Representations

The models described here use the novel components introduced in Section 2.

EaE Our primary model, **E**ntities **A**s **E**xperts embeds the input using the token embedding layer, then passes it through four layers of transformer. This output is used for the entity memory layer, then passed through an additional eight transformer layers. Finally, the model has two heads: one `TokenPrediction` head and one EntityPrediction head. In EAE, hyper-parameters for the transformer layers are identical to those in BERT-base (Devlin et al., 2018), described below.

Sparse Activation in EaE In total, EAE has 367m parameters which is similar to the BERT-large model described below. However, because EAE only accesses the entity embedding matrix for mention spans, and because it can quickly find the top-k entities using fast nearest maximum inner product search (Ram and Gray, 2012; Shen et al., 2015; Shrivastava and Li, 2014; Johnson et al., 2017), EAE uses far fewer parameters (110.5m) for each example at inference time. Unless otherwise mentioned, we use $k = 100$ at inference for all of the results presented in Section 7. We investigate the impact of this choice in Section 8.

EaE-unsup The EAE-unsup model is similar in architecture to EAE, but its entity representations are not supervised. This implies that the entity memory has no entity-linking loss and there is no entity prediction head. This model isolates the usefulness of supervising memory slots and access over the fully unconstrained setting. In our experiments with this model, we use full attention at inference when doing memory access.

No-EaE The No-EAE baseline seeks to isolate the impact of the entity-memory layer. No EAE has a token embedding layer, twelve transformer layers, and an `EntityPrediction` and a `TokenPrediction` head. This approach has similar number of parameters as EAE,[6] but only uses the entity embedding matrix at the final entity linking layer. In contrast with EAE, this baseline cannot model interactions between the entity representations in the entity embedding matrix. Also, the entity embeddings cannot be directly used to inform masked language modelling predictions.

[6]With the exception of projection matrices totalling less than one million parameters.

RELIC Ling et al. (2020) is a dual encoder with a BERT-Base architecture that compares a representation of a mention to an entity representation. It is similar to our No-EAE architecture. Its training is however different, as only linked mentions are masked and only one mention is masked at a time. In addition, RELIC does not have mention detection or masked language modelling losses. Finally, it is also initialized with the BERT checkpoint whereas we train our models from scratch.

5.2 Sequence Modeling Baselines

BERT We follow Petroni et al. (2019) and report the performance of BERT (Devlin et al., 2018) on the LAMA tasks introduced in Section 4. We report results for BERT-base, which has 110m parameters, and BERT-large, which has 340m parameters. The transformer architecture used by BERT-base is identical to the 12 transformer layers in EAE. BERT-large uses a much larger transformer, and has a similar number of parameters overall to EAE.

BERT-MM To ascertain which changes are due to the training data and mention masking function introduced in Section 2.3.2, and which are due to the entity-specific modeling and supervision, we also report performance for BERT-MM. This has the same architecture as BERT, but is trained on the same data as EAE, with the same input token masking function and masked language modeling loss. As above, we report results for the BERT-base and BERT-large model architectures.

5.3 Question Answering Baselines

We compare our system to two paradigms for QA: *Open-Book* systems (ORQA and Graph Retriever), which first retrieve a set of documents from a large corpus with the question and predict a span in the retrieved document as an answer and *Closed-Book* system (RELIC and T5): model answers the question from its parameters only, without retrieving question specific documents as an input.

ORQA (Lee et al., 2019) is a retrieval-based system with two components: a retriever and a reader. Both components use the BERT-base architecture. The retriever encodes the question and passages totalling the entire English Wikipedia. The top retrievals are then concatenated to the question and fed to the reader component. The reader then predicts an answer span.

Graph Retriever (Min et al., 2019b) is also a retrieval based system comprising a retriever and a reader. In contrast with ORQA, it can perform Graph-based, Entity-based and Text-match retrieval. These different retrieval modalities are then fused in representations that serve as input to the span-selection reader.

T5 T5 is an encoder-decoder transformer introduced in Raffel et al. (2019). It has been fine-tuned for open domain question answering in Roberts et al. (2020). In that setup, the model is trained to generate the answer to a question without any context. T5 does not explicitly model entities or have any form of memory. We compare to models of different sizes, from 220m parameters to 11B.

6 Experimental Setup

6.1 Wikipedia Pre-training

Wikipedia Processing We build our training corpus of contexts paired with entity mention labels from the 2019-04-14 dump of English Wikipedia. We first divide each article into chunks of 500 bytes, resulting in a corpus of 32 million contexts with over 17 million entity mentions. We restrict ourselves to the one million most frequent entities (86% of the linked mentions). These are processed with the BERT tokenizer using the lowercase vocabulary, limited to 128 word-piece tokens. In addition to the Wikipedia links, we annotate each sentence with unlinked mention spans using the mention detector from Section 2.3. These are used as additional signal for our mention detection component and allow the model to perform retrieval even for mentions that are not linked in Wikipedia. We set aside 0.2% of the data for development and 0.2% for test and use the rest to pre-train our model.

Pre-training hyper-parameters We pre-train our model from scratch. We use ADAM (Kingma and Ba, 2014) with a learning rate of 1e-4. We apply warmup for the first 5% of training, decaying the learning rate afterwards. We also apply gradient clipping with a norm of 1.0. We train for one million steps using a large batch size of 4096. We use a `TokenPrediction` head for all our models and an `EntityPrediction` head for the EAE and No-EAE models.

6.2 Task Specific Fine-tuning

Open Domain QA Preprocessing We annotate each question with proper-name mentions[7] using the mention detector from Section 2.3.

When the answer is an entity, in our entity vocabulary, we link the answer string to an entity ID using the SLING phrase table[8] (Ringgaard et al., 2017). If the answer is not an entity in our vocabulary, we discard the question from the training set, though we keep it in the development and test set to ensure fair comparisons with prior work. On the training set, we keep 77% of the training examples, most of the discarded examples being cases where the answer is not an entity.

Hyper-parameters We fine-tune the entire model using a learning rate of 5e-6, a batch size of 64 and performing 50,000 training steps.

7 Results

7.1 Predicting Entities in Context

We explore the ability of our model to predict entities and their text when they are masked, as described in Section 4.1. Table 1 shows the results for all models that are trained using the data and language modeling losses from Section 2.3.

We see that the MM-base and No-EAE models perform similarly on the token prediction task. These two models have exactly the same architecture up until the point of token prediction. This result indicates that the signal coming from the entity linking loss (Section 2.3.2) does not benefit language modeling when it is applied at the top of the transformer stack.

Introducing the entity memory layer in the middle of the transformer stack (EAE) improves performance on both language modeling and entity linking, compared to the No-EAE model. This indicates that the entity representations are being used effectively, and that the model is learning inter-entity relations, using the output from the entity memory layer to improve predictions at the downstream entity prediction layer.

When the memory layer is not supervised, performance on token prediction accuracy and per-

[7]Nominal mentions in questions typically refer to the answer entity. This is unlike Wikipedia text, where nominal mentions refer to entities that have previously been named. We only link proper name mentions in questions so that the model is not forced to hallucinate links for entities that have not been properly introduced to the discourse.

[8]https://github.com/google/sling

Model	Params	Entity Acc	Token Acc	Token PPL
MM-Base	110m	-	45.9	18.0
MM-Large	340m	-	53.4	10.3
EAE-unsup	366m	-	46.9	16.9
No EAE	366m	58.6	45.0	19.3
EAE	367m	61.8	56.9	11.0

Table 1: Results for our model and baselines on cloze-style entity prediction accuracy, token prediction accuracy and perplexity on the held-out set of our Wikipedia pre-training data.

plexity is significantly worse than for EAE. This underlines the importance of entity linking supervision in teaching EAE how to best allocate the parameters in the entity memory layer.

Finally, EAE performs better at predicting mention tokens correctly than the similar sized MM-large, but does marginally worse in terms of perplexity. We believe that EAE is overconfident in its token predictions when wrong, and we leave investigation of this phenomenon to future work. However, we find it encouraging that EAE performs favourably against MM-Large despite only using a third of the parameters at inference time.

7.2 LAMA

We follow Petroni et al. (2019)'s setup when evaluating on the LAMA knowledge probe (described in Section 4.2).

Table 2 shows results for the models described in Sections 5.1 and 5.2. Adding the entity memory layer to the BERT-base model improves performance across the board on this task. Not supervising this layer with entity linking is worse, with the exception of SQuAD where it is slightly better. EAE achieves a similar average accuracy to BERT-large. However, the LAMA sub-task accuracies show that the two models are actually performing very differently. EAE is significantly better than BERT-large when predicting the mention-like words in the SQuAD and T-Rex probes. It is marginally worse for the RE probe, and very significantly worse for the ConceptNet probe, though. As discussed in Section 4.2, ConceptNet is not constrained to mention-like predictions and RE contains many dates. As a result, all of the models that use the mention masking strategy from Section 2.3 perform less well on these tasks.

7.3 Open domain QA

Table 3 shows results for the TriviaQA task. The table is broken down into open-book approaches, which retrieve evidence documents from a corpus, and then select answer spans, and closed-book approaches that store all of the information needed in a set of learned parameters. Overall, open-book approaches outperform closed-book approaches. This is not surprising—open-book approaches have access to an enormous and redundant evidence corpus at inference time.

Comparing EAE to No-EAE, we see that adding entity memory to the transformer encoder adds 5.5 points. Both of these models outperform the RELIC model which also uses a large entity embedding table to represent answers, but does not have any entity specific supervision of the question encoder. Even more interestingly, EAE outperforms all of the T5 models including one that has $30\times$ the parameters. This indicates that the entity specific model architecture is more efficient in capturing the sort of information required for this knowledge probing task than the general Transformer architecture used by T5. Furthermore, EAE only uses a small proportion of its parameters at inference time—only the top 100 entity embeddings are accessed for each mention in the question—whereas T5's transformer architecture uses almost all of its parameters for each example. The only conditional activation in T5 comes from the deterministic selection of word-piece embeddings, which only accounts for a tiny share of its parameters.

Finally, it should be noted that EAE is only capable of answering the TriviaQA questions that have entity answers. We successfully resolve 85% of the answers in Unfiltered-Dev to Wikipedia entities and measure the accuracy of EAE, Graph Retriever, and T5 on this slice. All three models improve in performance. T5's accuracy goes up to 47.4%, EAE's accuracy goes up to 52.3%, and Graph Retriever's accuracy goes up to 63.0%. This indicates that the entity-answer slice is easier for all three models, but the gains (9.1%) are larger for EAE. In Section 8 we provide a detailed analysis of model accuracy on TriviaQA, broken down according to the frequency of the answer in Wikipedia, and the number of properly named entities in the question.

8 Analysis

We provide qualitative and quantitative analysis of our model. First, we study the impact of the num-

Model	Params	ConceptNet	RE	SQuAD	T-REx	Avg.
BERT-base	110m	15.6	9.8	14.1	31.1	17.7
BERT-large	340m	19.2	10.5	17.4	32.3	19.9
MM-Base	110m	11.0	9.6	17.2	30.6	17.1
MM-Large	340m	12.4	6.5	24.4	31.4	18.7
EAE-unsup	366m	10.6	8.4	23.1	30.0	18.0
No EAE	366m	10.3	9.2	18.5	31.8	17.4
EAE	367m	10.7	9.4	22.4	37.4	20.0

Table 2: Results on the LAMA probe. Adding entity memory improves performance for the SQuAD and T-Rex probes, which typically require prediction of mention words. Our mention masking strategy reduces performance on the ConceptNet sub-task, which require prediction of non-mention terms such as "happy".

Name	# Params	Unfiltered-Dev	Wiki-Test
Open-Book Approaches: Retrieval + Span Prediction			
BM25 + BERT Lee et al. (2019)	110m	47.2	-
ORQA Lee et al. (2019)	330m	45.1	-
GraphRetriever Min et al. (2019b)	110m	**55.4**	-
Closed-Book Approaches: Generation (Roberts et al., 2020)			
T5-Base	220m	-	29.1
T5-Large	770m	-	35.9
T5-3B	3B	-	43.4
T5-11B	11B	42.3	50.1
Closed-Book Approaches: Entity Prediction			
RELIC Ling et al. (2020)	3B	35.7	-
No EAE	366m	37.7	-
EAE	367m	43.2	**53.4**

Table 3: Exact Match open-domain factoid question answering on TriviaQA. Open-Book approaches retrieve documents to answer the question, while closed-book approaches rely solely on learned parameters. The open-book approaches all reserve 10% of the TriviaQA training data for development, and the T5 systems merge the Unfiltered-Dev set into their training set before running on Wiki-Test. For a thorough description of these choices, see Appendix 8.2.

ber of retrieved entity embeddings per mention in the entity memory layer, which has a significant impact on inference speed. Second, we compare the predictions of our entity prediction QA model with generation-based and retrieval-based QA models.

8.1 Top-K over Entity Embeddings

Table 4 shows how varying the number of retrieved entity embeddings in the entity memory layer at inference time impacts accuracy. Even for $K = 10$, performance does not deteriorate meaningfully.

This is a key advantage of our modular approach, where entity information is organized in the Entity Memory layer. Despite only accessing as many parameters as BERT Base, our model outperforms BERT Large on token prediction. Similarly in TriviaQA, we outperform T5's model, while assessing about 1% of the parameters.[9] While naive implementation would not bring significant computational gain, fast nearest neighbor methods methods on entity embeddings enable retrieving the Top K entities in sub-linear time and storage, enabling efficient model deployment.

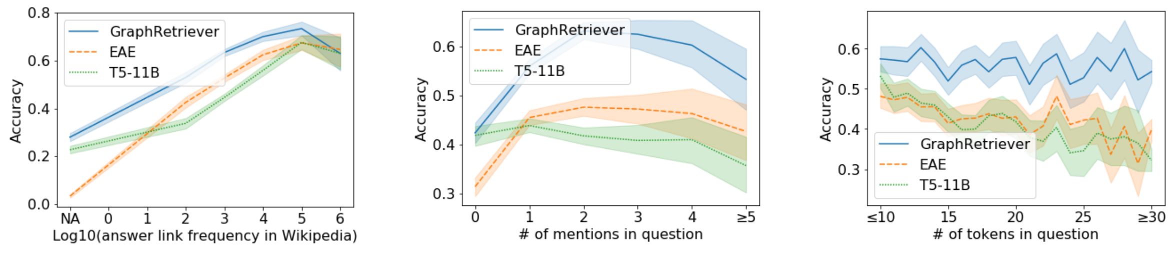
K	Entity acc	Tok acc	Tok PPL	TQA
1	59.2	56.7	18.0	40.1
10	61.7	57.2	11.1	43.1
100	61.8	57.1	11.0	43.2
Full (10^6)	61.8	56.9	11.0	43.4

Table 4: Impact of varying the number of retrieved entity embeddings (K) in the Entity Memory layer at inference on the entity prediction and TriviaQA tasks.

8.2 Analysis of TriviaQA Results

Comparing QA paradigms We compare the performance of four systems (ORQA, GraphRetriever (GR), T5, EAE) on the TriviaQA Unfiltered-

[9]We acknowledge this comparison is a bit unfair, as T5's model is also pre-trained for other tasks such as machine translation.



(a) (b) (c)

Figure 3: Performance on TriviaQA's Unfiltered Dev set for GraphRetriever, T5 and our EAE model by (a) the frequency of the answer entity in the Wikipedia corpus (NA if not an entity), (b) the number of named entity mentions in the question, (c) the number of tokens in the question.

Systems	Oracle Acc.	Pred Overlap (%)
T5 & EAE	55.9	29.3
T5 & GR	**66.4**	30.1
EAE & GR	64.6	33.6
ORQA & GR	63.8	**39.6**

Table 5: Comparing prediction overlap and oracle accuracy on TriviaQA. Oracle accuracy considers a prediction correct if at least one of the model prediction is correct. While ORQA and GR outperform EAE and T5, their predictions overlap more with GR and offer less complementary value.

Dev set. GR achieves an accuracy of 55.4, ORQA 45.1, EAE 43.2 and T5 42.3. As the open-book paradigm differs significantly from the closed-book one, we intuit they might complement each other.

To test this hypothesis, we measure prediction overlap and oracle accuracy. Oracle accuracy considers a prediction correct if either system is correct (see Table 5). Unsurprisingly, the two open book approaches show the most similar predictions, overlapping in nearly 40% of examples. While ORQA outperforms T5 and EAE, the oracle accuracy of ORQA & GR is lower than ORQA & T5 or ORQA & EAE. This suggests some questions might be better suited to the closed book paradigm. In addition, the oracle accuracy of the two closed book systems is higher than that of the best performing open book system. We leave designing approaches that better combine these paradigms to future work.

Entity-Based Analysis We compare the performances of retrieval-based GR, generation-based T5-11B, and our EAE model on the TriviaQA Unfiltered-dev set. Figure 3 (a) compares performance based on the frequency of the answer entity as a hyperlink in the Wikipedia corpus. Unsurprisingly, all models perform better on frequent entities. EAE shows lower performance for entities seen fewer than one hundred times during training, likely because their entity embedding does not contain enough information. Figure 3 (b) shows an interesting trend: as the number of named entity mentions grows in the question, T5 performance decreases whereas EAE and GR performance increases. We presume more entity mentions makes it easier to retrieve relevant documents, thus contributing to better performance for GR. For our model, having more entity mentions allows the model to access more entity knowledge. We conduct further qualitative analysis in Section 6. Figure 3 (c) compares performance based on the length of the question. Closed-book models seem to perform worse when questions are long, whereas the trend disappears for open-book models.

Manual Analysis We randomly sampled 100 examples from unfiltered-dev set, and analysed the ability of EAE's to correctly identify and link the question's entities. We find that 87% of questions do not have any incorrectly predicted entities.[10]

We find that when there is at least one incorrectly linked entity, the performance of EAE is considerably reduced. On this small sample, the performance is even lower than for examples in which there are no named entities in the question.

Table 6 illustrates a representative sample of questions and predictions from EAE and T5. We break this sample down into questions that contain no named entities, questions that contain only correctly linked named entities, and questions that contain incorrectly linked named entities.

In the first two examples, questions contain dates but no proper names. Since we do not have rep-

[10]In this analysis, we do not define a strict bracketing to decide which of the named entities in nested phrases like [[1966 [FIFA World Cup]] Final] should be predicted, and we leave the investigation of the impact of this choice to future work.

	Questions with no proper name mentions	**Answer**	**T5 Prediction**	**EaE Prediction**
i	Next Sunday, Sept 19, is International what day?	Talk like a pirate day	talk like a pirate day	Pearl Harbor Remembrance Day
ii	What links 1st January 1660 and 31st May 1669?	First and last entries in Samuel Pepys's diaries	they were the dates of the henry viii's last bath	Anglo-Dutch Wars
iii	Which radioactive substance sometimes occurs naturally in spring water?	radon	radon	radon
	Questions with only [correctly] linked entities			
iv	Which [Dr. Who] villain has been played by [Roger Delgado], [Anthony Ainley], [Eric Roberts], etc?	The Master	mr. daleks	The Master
v	Who directed the 2011 [Palme d'Or] winning film '[The Tree Of Life]'?	Terence Malick	ang lee	Terence Malick
vi	Name the year: [Hirohito] dies; The [Exxon Valdez] runs aground; [San Francisco] suffers its worst earthquake since 1906.	1989	1989	1990s
	Questions with {incorrectly} linked entities			
vii	Which car manufacturer produces the **{Jimmy}** model?	Suzuki	suzuki	Brixham
	Jimmy $\rightarrow$ Marcos Engineering Q1637323			
viii	Where do you find the **{Bridal Veil}**, **[American]**, and **[Horseshoe Falls]**?	Niagara falls	niagra Falls	Niagara Falls
	Bridal Veil $\rightarrow$ Veil (Garment) Q6497446			
ix	Which early aviator flew in a plane christened **{Jason}**?	Amy Johnston	jean batten	Icarus
	Jason $\rightarrow$ Jason (Greek Mythology) Q176758			

Table 6: Examples of question, answer and model predictions for the TriviaQA Unfiltered dev set. Questions are annotated with the entity prediction from our EaE model: correctly predicted entity mention are in square brackets and incorrectly predicted mention in curly brackets. Our model suffers when entity mention are linked incorrectly, while it is more successful when entity linking is successful.

resentation for dates in the entity memory, these are challenging and our predictions are incorrect. Nonetheless, our answers are of feasible types—an annual event, and a 17th century event. While T5 also has no distinct representation for dates, its 11B parameters managed to memorize the esoteric connection between the phrases 'Sept 19' and 'talk like a pirate day'. Example (ii) shows an amusing failure mode of generation based approaches. They hallucinate syntactically correct, but nonsensical answers. In the third example without named entities, the answer entity is sufficiently frequent in Wikipedia (top 50k entities), and EaE seems to have learned its connection with the specific categorical information in the question.

When there are many entities mentioned in the question, like in example (iv) EaE is able to predict very rare answer entities. In this case, 'The Master' only occurs 38 times in our training data, but in those occurrences it is likely that at least one of the character's actors are also mentioned. EaE learns the character-actor relationship, while T5 makes up a character name based on a common category of Dr. Who villain. Similarly, EaE has learned the connection between 'The Tree of Life' and 'Terrence Malick', while T5 predicts a different director's name. Finally, example (vi) shows how EaE fails at predicting date answers, even when there is abundant information in the question, predicting the entity representing the 1990's.

When EaE predicts the wrong entity in the question, this is often because the correct entity is not available (example (ix)) or there may be a much more frequent entity with the same name (example (viii)). Example's (vii) typo (from 'Jimny' to 'Jimmy') is particularly interesting: when fixed, EaE links 'Jimny' correctly and predicts the right answer. In example (ix) EaE mistakes the question

ID for the Greek mythological figure Jason and in turn predict Icarus, another Greek mythological figure as the answer. Interestingly, Icarus is the one Greek mythological figure that could reasonably be called an aviator.[11]. Finally, In example (viii), an incorrectly predicted question entity does not cause a failure. This is likely thanks to the two other correctly predicted question entities.

Overall, linking errors are very detrimental, suggesting better linking accuracy can improve question answering performance.

[11]https://en.wikipedia.org/wiki/Icarus

References

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

Kyunghyun Cho and Yoshua Bengio. 2014. Exponentially increasing the capacity-to-computation ratio for conditional computation in deep learning. *arXiv preprint 1406.7362*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *NeurIPS*.

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

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. 2013. Learning factored representations in a deep mixture of experts. *arXiv preprint 1312.4314*.

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

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint 1702.08734*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *TACL*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint 1705.03551*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint 1412.6980*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *TACL*.

Guillaume Lample, Alexandre Sablayrolles, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2019. Large memory layers with product keys. In *NeurIPS*.

Anne Lauscher, Ivan Vulić, Edoardo Maria Ponti, Anna Korhonen, and Goran Glavaš. 2019. Informing unsupervised pretraining with external linguistic knowledge. *arXiv preprint 1909.02339*.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. *arXiv preprint 1906.00300*.

Yoav Levine, Barak Lenz, Or Dagan, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. 2019. Sensebert: Driving some sense into bert. *arXiv preprint 1908.05646*.

Jeffrey Ling, Nicholas FitzGerald, Zifei Shan, Livio Baldini Soares, Thibault Févry, David Weiss, and Tom Kwiatkowski. 2020. Learning cross-context entity representations from text. *arXiv preprint 2001.03765*.

Robert Logan, Nelson F Liu, Matthew E Peters, Matt Gardner, and Sameer Singh. 2019. Baracks wife hillary: Using knowledge graphs for fact-aware language modeling. In *ACL*.

Sewon Min, Danqi Chen, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019a. A discrete hard em approach for weakly supervised question answering. *arXiv preprint 1909.04849*.

Sewon Min, Danqi Chen, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019b. Knowledge guided text retrieval and reading for open domain question answering. *arXiv preprint 1911.03868*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint 1802.05365*.

Matthew E Peters, Mark Neumann, Robert L Logan, IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A Smith. 2019. Knowledge enhanced contextual word representations. *arXiv*.

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

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 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 1910.10683*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Parikshit Ram and Alexander G Gray. 2012. Maximum inner-product search using cone trees. In *International conference on Knowledge discovery and data mining*.

Michael Ringgaard, Rahul Gupta, and Fernando CN Pereira. 2017. Sling: A framework for frame semantic parsing. *arXiv preprint 1710.07032*.

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

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint 1701.06538*.

Fumin Shen, Wei Liu, Shaoting Zhang, Yang Yang, and Heng Tao Shen. 2015. Learning binary codes for maximum inner product search. In *ICCV*.

Anshumali Shrivastava and Ping Li. 2014. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *NeurIPS*.

Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *NeurIPS*.

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 *Neurips*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding.

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

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

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

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

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2016. Reference-aware language models. *arXiv preprint 1611.01628*.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. Ernie: Enhanced language representation with informative entities. *arXiv preprint 1905.07129*.

Appendices

TriviaQA Evaluation Data Configuration

As the TriviaQA dataset was originally introduced for reading comprehension task, prior work adapted the initial splits and evaluation to reflect the open domain setting. We describe the setup used by prior approaches and introduce ours, to enable fair comparison. The dataset comes in two blends, Wikipedia and Web documents. While TriviaQA's official web evaluation uses (question, document, answer) triplets, all open domain approaches average over (question, answer) pairs when using the Web data.

Open-book Approaches: Lee et al. (2019); Min et al. (2019a,b) use only the web splits. They use 90% of the original train data for training, performing model selection on the remaining 10% and reporting test numbers on the original development set since the original test set is hidden.

Previous Closed-book Approaches: Roberts et al. (2020) also use the web data to train and validate their model. However, they use the Wikipedia[12] test split as a test set. After hyper-parameter tuning, they re-train a model on both the training and development sets of the web data. Ling et al. (2020) uses the original web splits and reports performance on the development set.

Our approach: To compare our approach more closely to T5, we follow their setup, with the exception that we do not re-train a model on both the train and development splits after hyper-parameter selection.

[12]https://competitions.codalab.org/competitions/17208