Entities as Experts: Sparse Memory Access with Entity Supervision

Thibault Févry∗ Livio Baldini Soares Nicholas FitzGerald
thibaultfevry@gmail.com liviobs@gmail.com nfitz@gmail.com

Eunsol Choi†
The University of Austin at Texas
eunsol@cs.utexas.edu

Tom Kwiatkowski
Google Research
tomkwiat@gmail.com

Google Research

Abstract

We focus on the problem of capturing declarative knowledge about entities 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. We show that EAE’s learned representations capture sufficient knowledge to answer TriviaQA questions such as “Which Dr. Who villain has been played by Roger Delgado, Anthony Ainley, Eric Roberts?”, outperforming an encoder-generator Transformer model with 10× the parameters. According to the LAMA knowledge probes, EAE contains more factual knowledge than a similarly sized BERT, as well as previous approaches that integrate external sources of entity knowledge. Because EAE associates parameters with specific entities, it 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.

1 Introduction

Neural network sequence models, pre-trained as language models, have recently revolutionized text understanding (Dai and Le, 2015; Peters et al., 2018; Howard and Ruder, 2018; Devlin et al., 2018), and recent work has suggested that they 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).

In this paper, we focus on developing neural sequence models that capture the knowledge required to answer questions about real world entities. 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 learns entity representations from text along with all the other model parameters. We call our model Entities as Experts (EAE), since it divides the parameter space according to entity identity. This name also reflects EAE’s similarities with the Massive Mixture of Experts (Shazeer et al., 2017), as well as other work that integrates learned memory stores into sequence models (Weston et al., 2014; Lample et al., 2019).

To understand the motivation for distinct and independent entity representations, consider Figure 1. A traditional Transformer (Vaswani et al., 2017) needs to build an internal representation of Charles Darwin from the words “Charles” and “Darwin”, both of which can also refer to 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 memory can also be accessed for other mentions of Darwin, such as “Charles
layers
layers
Robert Darwin” or “the father of natural selection”. Retrieving and re-integrating this memory makes it easier for EAE to find the answer.

We train EAE to predict masked-out spans in English Wikipedia text (Devlin et al., 2018); to only access memories for entity mention spans; and to access the correct memory for each entity mention. Mention span supervision comes from an existing mention detector, and entity identity supervision comes from 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, and only the mentioned memories need to be retrieved.

We evaluate EAE’s ability to capture declarative knowledge using the LAMA knowledge probes introduced by Petroni et al. (2019), as well as the open-domain variants of the TriviaQA and WebQuestions question answering tasks (Joshi et al., 2017; Berant et al., 2013). On both tasks, EAE outperforms related approaches with many more parameters. An in-depth analysis of EAE’s predictions on TriviaQA shows that the correct identification and reintegration of entity representations is essential for EAE’s performance.

We further demonstrate that EAE’s learned entity representations are better than the pre-trained embeddings used by Zhang et al. (2019); Peters et al. (2019) at knowledge probing tasks and the TACRED relation extraction task (Zhang et al., 2017; Alt et al., 2020). We show that training EAE to focus on entities is better than imbuing a similar-sized network with an unconstrained memory store, and explain how EAE can outperform much larger sequence models while only accessing a small proportion of its parameters at inference time.

2 Approach

Let $\mathcal{E} = \{e_1 \ldots e_N\}$ be a predefined set of entities, and let $\mathcal{V} = \{\text{MASK}, w_1 \ldots w_M\}$ be a vocabulary of tokens. A context $x = [x_0 \ldots x_L]$ is a sequence of tokens $x_i \in \mathcal{V}$. Each context comes with the list of the mentions it contains, $m = [m_0 \ldots m_M]$, 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 mentions might not be linked to a specific entity in $\mathcal{E}$, thus $e_{m_i} \in \mathcal{E} \cup \{\emptyset\}$, where $\emptyset$ refers to the null entity.

2.1 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. Figure 2 illustrates our model. Our model is:

$$
X^0 = \text{TokenEmbed}(x)
$$

$$
X^1 = \text{Transformer}(X^0, \text{num_layers} = l_0)
$$

$$
X^2 = \text{EntityMemory}(X^1)
$$

$$
X^3 = \text{LayerNorm}(X^2 + X^1)
$$

$$
X^4 = \text{Transformer}(X^3, \text{num_layers} = l_1)
$$

$$
X^5 = \text{TaskSpecificHeads}(X^4)
$$

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, normalized, then processed by additional transformer layers. Throughout this work we use $l_0 = 4$ and $l_1 = 8$.

**Entity Memory Layer** Let $E$ be a matrix of learned entity embeddings of shape $(N, d_{ent})$. $\text{EntEmbed}(e_i)$ maps an entity $e_i$ to its row in $E$. The entity memory layer takes the output sequence from the preceding transformer layer $(X^l)$ and outputs a new sequence $(X^{l+1})$, 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 $E$, at position $s_{m_i}$.

$$
x_i^{l+1} = W_b E_{m_k} \quad \text{if} \quad i = s_{m_k}
$$

where $W_b$ 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.

$$
h_{m_i} = W_f [x_{s_{m_i}}^l || x_{t_{m_i}}^l]
$$

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

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

$$
E_{m_i} = \sum_{e_j \in \text{topk}(E, h_{m_i}, k)} \alpha_j \cdot (\text{EntEmbed}(e_j))
$$

$$
\alpha_j = \frac{\exp(\text{EntEmbed}(e_j) \cdot h_{m_i})}{\sum_{e \in \text{topk}(E, h_{m_i}, k)} \exp(\text{EntEmbed}(e) \cdot h_{m_i})}
$$
Figure 2: The Entities as Experts 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 \( \text{topK}(E, h_{m_i}, k) \) returns the \( k \) entities that yield the highest score \( \text{EntEmbed}(e_j) \cdot h_{m_i} \). We use \( k = N \) to train and use \( k = 100 \) at inference (see Section 4.1 and 6.3).

The entity memory layer can be applied to any sequence output without loss of generality. We apply it 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: TokenPred and EntityPred.

The TokenPred head predicts masked tokens for a cloze task. Each masked token’s final representation \( x^4_i \) is fed to an output softmax over the token vocabulary, as in BERT.

The EntityPred head predicts entity ids for each entity mention span (i.e., entity linking). We build the pseudo entity embedding \( h_{m_i} \) from the last sequence output \( X^4 \). Then, the model predicts the entity whose embedding in \( E \) is the closest to the pseudo entity embedding.

**Inference-time Mention Detection** We introduce a mention detection layer to avoid dependence at inference on an external mention detector. The mention detection layer applies a BIO\(^1\) classifier to the first transformer block’s output. We decode the entire BIO sequence, ensuring that inconsistent sequences are disallowed. We use inferred mention spans at inference for all our experiments.

2.2 Training

2.2.1 Data and Preprocessing

We assume access to a corpus \( D = \{(x_i, m_i)\} \), where all entity mentions are detected but not necessarily all linked to entities. We use English Wikipedia as our corpus, with a vocabulary of \( 1 \)m entities. Entity links come from hyperlinks, leading to \( 32 \)m \( 128 \) byte contexts containing \( 17 \)m entity links. Non-overlapping entity mention boundaries come from hyperlinks and the Google Cloud Natural Language API\(^2\) leading to \( 140 \)m mentions.

We remove 20% of randomly chosen entity mentions (all tokens in the mention boundaries are replaced with [MASK]) to support a masked language modeling objective. See Appendix B for full details of our pre-processing and pre-training hyperparameters.

2.2.2 Learning Objective

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

**Mention Detection** The BIO classification of tokens is supervised with a cross-entropy loss over the labels. Assuming the mention boundaries are complete, we apply this supervision to all tokens.

**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

\(^{1}\)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
$e_i \neq e_∅$, the pseudo embedding $h_{m_i}$ (Equation 2) should be close to the entity embedding of the annotated entity, $\text{EntEmbed}(e_{m_i})$.

$$\text{ELoss} = \sum_{m_i} \alpha_i \cdot \mathbb{1}_{e_{m_i} \neq e_∅} \left( \text{EntEmbed}(e_{m_i}) \cdot h_{m_i} \right)$$

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

Note that this loss is not applied to 88% of mentions, which do not have hyperlinks. Memory access for those mentions is unsupervised. The same loss is used for the EntityPred head.

**Masked Language Modelling**  We follow BERT, and train the TokenPred head to independently predict each of the masked out tokens in an input context, using a cross-entropy loss over $\mathcal{V}$.

### 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). Our focus is 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). Closest to our work, KNOWBERT (Peters et al., 2019) introduce an entity memory layer that is similar to the one in EAE. In contrast with our work, KNOWBERT starts from the BERT checkpoint, does not train with a knowledge-focused objective such as our mention-masking input function and uses pre-computed entity representations when integrating the information from knowledge bases. In addition, KNOWBERT relies on a fixed, pre-existing candidate detector (alias table) to identify potential candidates and entities for a span while our model learns to detect mentions. We compare to their approach in Section 7.

**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 unconstrained, 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, approaches such as large mixture of experts (Shazeer et al., 2017) or gating (Eigen et al., 2013; Cho and Bengio, 2014) have been used. Our method proposes entities as experts, which allows us to supervise memory access at two levels. We only access memories for entity mentions, and we only need to access memories for the entities that were mentioned.

### 4 Models Evaluated

We evaluate EAE on close knowledge probes, open-domain question answering and relation extraction. Here, we describe baselines from previous work and ablations.

**4.1 The Entities as Experts models**

Our primary model, Entities As Experts embeds the input using the token embedding layer, then passes it through 4 transformer layers. This output is used for the entity memory layer, which uses embeddings of size 256, then passed through an additional 8 transformer layers. Finally, the model has a TokenPred head and an EntityPred head. In EAE, hyper-parameters for the transformer layers are identical to those in BERT-base (Devlin et al., 2018).

**Sparse Activation in EaE** EAE only accesses the entity embedding matrix for mention spans, and we only retrieve $k=100$ entity memories for each mention (see Appendix 6.3 for analysis of this
choice). This type of conditional computation can facilitate massive model scaling with fixed computational resources (Shazeer et al., 2017; Lample et al., 2019) and, while our current implementation does not yet include an efficient implementation of top-k routing (Section 2), we note that it is possible with fast maximum inner product search (Ram and Gray, 2012; Shen et al., 2015; Shrivastava and Li, 2014; Johnson et al., 2017). We leave the implementation and investigation of this to future work.

4.2 Ablations

EaE-unsup In EaE-unsup, the entity memory is not supervised to isolate the usefulness of supervising memory slots with entity linking. We use full attention at inference when doing memory access.

No-EaE This ablation seeks to isolate the impact of the entity-memory layer. No EAE has a token embedding layer, twelve transformer layers, and an EntityPred and a TokenPred head. This approach has similar number of parameters as EAE, but only uses the entity embedding matrix at the EntityPred head. 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.

BERT / MM We compare to the BERT model. To ascertain which changes are due to EAE’s data and masking function (Section 2.2.2), and which are due to the entity-specific modeling, we report performance for BERT-MM which uses BERT’s architecture with EAE’s masking function and data. We present results for Base and Large model sizes.

4.3 Question Answering Models

RELIC learns entity embeddings that match BERT’s encoding of the contexts in which those entities were mentioned (Ling et al., 2020). T5 is an encoder-decoder trained on an enormous web corpus. We compare to the version fine-tuned for open-domain question answering (Roberts et al., 2020). We also compare to the nearest neighbour baselines introduced by Lewis et al. 2020; and we compare to three recent QA models that use a retriever and a reader to extract answers: BM25+BERT and ORQA from Lee et al. 2019 and GraphRetriever (GR) is introduced by Min et al. 2019b. All are described fully in Appendix A.

5 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).

5.1 Predicting Wikipedia Hyperlinks

We explore the ability of our model to predict masked out hyperlink mentions from Wikipedia, similar to the pre-training task (Section 2.2). We calculate accuracy on a 32k test examples separate from the training data (Appendix B).

Table 1 shows the results for all our models. The MM-base and No-EAE models perform similarly on the token prediction task. These two models have the same architecture up until the point of token prediction. This indicates that the signal coming from the entity linking loss (Section 2.2.2) does not benefit language modeling when it is applied at the top of the transformer stack only.

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 and token prediction layers.

If the memory layer is not supervised (EAE-unsup), performance on token prediction accuracy and perplexity 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 than the 24-layer MM-large, but does marginal 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.

5.2 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 a zero-shot probing task, it does not involve any

4With the exception of projection matrices totalling less than one million parameters.

5In pre-training, we also mask non-hyperlinked mentions.
Table 1: Results on cloze-style entity prediction accuracy, token prediction accuracy and perplexity on the test set of our masked hyperlink data.

<table>
<thead>
<tr>
<th>Model</th>
<th>Params</th>
<th>Entity Acc</th>
<th>Acc</th>
<th>PPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>MM-base</td>
<td>110m</td>
<td>45.0</td>
<td>19.6</td>
<td></td>
</tr>
<tr>
<td>MM-large</td>
<td>340m</td>
<td>53.4</td>
<td>10.3</td>
<td></td>
</tr>
<tr>
<td>EAE-unsup</td>
<td>366m</td>
<td>46.9</td>
<td>16.9</td>
<td></td>
</tr>
<tr>
<td>No EAE</td>
<td>366m</td>
<td>58.6</td>
<td>45.0</td>
<td>19.3</td>
</tr>
<tr>
<td>EAE</td>
<td>367m</td>
<td>61.8</td>
<td>56.9</td>
<td>11.0</td>
</tr>
</tbody>
</table>

Table 2: Results on the LAMA probe. Adding entity memory improves performance for the probes that focus on entities. Mention masking reduces performance on ConceptNet sub-task which requires prediction of non-mention terms such as “happy”.

<table>
<thead>
<tr>
<th>Model</th>
<th>Concept Net RE</th>
<th>SQuAD T-REx</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT-base</td>
<td>15.6</td>
<td>9.8</td>
<td>14.1</td>
</tr>
<tr>
<td>BERT-large</td>
<td>19.2</td>
<td>10.5</td>
<td>17.4</td>
</tr>
<tr>
<td>MM-base</td>
<td>10.4</td>
<td>9.2</td>
<td>16.0</td>
</tr>
<tr>
<td>MM-large</td>
<td>12.4</td>
<td>6.5</td>
<td>24.4</td>
</tr>
<tr>
<td>EAE-unsup</td>
<td>10.6</td>
<td>8.4</td>
<td>23.1</td>
</tr>
<tr>
<td>No EAE</td>
<td>10.3</td>
<td>9.2</td>
<td>18.5</td>
</tr>
<tr>
<td>EAE</td>
<td>10.7</td>
<td>9.4</td>
<td>22.4</td>
</tr>
</tbody>
</table>

Task specific model training, and we do not apply any LAMA-specific modeling. Table 2 shows that adding the entity memory layer to the MM-base model improves performance across the board on this task. Not supervising this layer with entity linking (EAE-unsup) is worse overall.

EAE’s average accuracy is similar to BERT-large. However, the LAMA sub-task accuracies show that the two models are complementary. Mention focused approaches are much better than the BERT baselines at predicting the mention like words in the SQuAD and T-REx probes, but they are marginally worse for the RE probe and very significantly worse for the ConceptNet probe. This is because the ConceptNet sub-task mostly includes non-entity answers such as “fly”, “cry”, and “happy”, and a third of the answers in the Google-RE sub-task are dates. We leave modeling of non-entity concepts and dates to future work. Finally, the entity specific memory in EAE is most beneficial for T-Rex, which focuses on common entities that are likely in our 1m entity vocabulary.

5.3 Open domain Question Answering

Setup TriviaQA and WebQuestions were introduced as reading comprehension and semantic parsing tasks, respectively (Joshi et al., 2017; Berant et al., 2013). More recently, these datasets have been used to assess the performance of QA systems in the open domain setting where no evidence documents or database is given. In this setup, TriviaQA contains 79k train examples and WebQuestions 3.1k. Most approaches rely on a text corpus at test time, extracting answers from evidence passages returned by a retrieval system. However, T5 and RELIC used neural networks to answer questions directly. We follow Roberts et al. 2020 in describing approaches as open-book (test time access to corpus) and closed-book (no test time access to corpus), and we report the nearest neighbour results from Lewis et al. 2020.

We follow RELIC and resolve answer strings to Wikipedia entity identifiers. We follow the training procedure from Section 2.2, with a round of task specific training that applies the entity linking and mention detection losses to the question answering data. Each question is appended with a special ‘answer position’ token and EAE is trained to predict the correct answer entity in this position, using the entity linking loss from Section 2.2.2. Mention spans are identified within the question (Section 2.2.1) and the mention detection loss from Section 2.2.2 is applied to encourage EAE to access the entity memory for entities in the question.

See Appendix C for additional information on task setup and fine-tuning hyper-parameters.

Results Table 3 shows results on two open domain QA datasets. Entity prediction methods, RELIC and EAE, significantly outperform nearest neighbor baselines, showing that model generalizes beyond train / development overlap and entity representations contains information about answer entities. No-EAE and RELIC both encode text with a transformer and retrieve answer entities. Compared to RELIC, No-EAE is trained to identify all entities in a piece of text, instead of just one. This leads to small but significant gains on TriviaQA. A much larger gain (almost 6 points on both datasets) comes from adding an entity memory inside the transformer encoder (EAE). We also show that it is possible to improve performance on TriviaQA by doubling the size of the entity embeddings to 512d (EAE-emb-512). While this almost doubles the model size, it does not significantly increase

---

We resolve 77% of TriviaQA train; 84% of TriviaQA dev; 84% of WebQuestions train; 95% of WebQuestions dev; 91% of WebQuestions test. See Appendix C for full details. Answering with free-form text is out of this paper’s scope.
Table 3: Exact Match accuracy on TriviaQA and Web-Questions. Open-book approaches reserve 10% of the training data for development; entity prediction approaches only train on linked entity answers; and T5 merges Unfiltered-Dev into training for Wiki-Test. For more description of these choices, see Appendix E.

Though closed-book approaches are still behind open-book approaches on TriviaQA, we believe even higher performances could be attained by ensembling diverse approaches and a preliminary study (Appendix F), indicates that ensembling open-book with closed-book approaches is preferable to ensembling within a single paradigm.

6 Analysis of TriviaQA Results

6.1 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) shows that all models perform better on frequent entities. EAE’s performance is much lower for entities seen fewer than one hundred times, likely because their entity embedding do not contain enough information. Figure 3 (b) shows that 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 EAE, having more entity mentions allows the model to access more entity knowledge. We conduct further qualitative analysis below. Figure 3 (c) shows that closed-book models under-perform on long questions. The trend disappears for open-book models.

6.2 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. 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 4 illustrates three representative examples from EAE and T5. In the first example, the question contains a date but no proper names. Since EAE does not have a representation for dates in the entity memory, this question is challenging and the model predicts an incorrect answer of the correct type (annual event). The second example demonstrates EAE’s ability to model connections between entities. In this case, ‘The Master’ only occurs 38

7 We do not define a strict bracketing to decide which entities in nested phrases like [[1966 [FIFA World Cup]] Final] should be predicted.
Figure 3: Performance on TriviaQA by: answer frequency in our Wikipedia training corpus (NA if not linked); proper names in the question; tokens in the question. Standard deviations obtained through bootstrapping.

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>T5 Prediction</th>
<th>EaE Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next Sunday, Sept 19, is International what day?</td>
<td>Talk like a pirate day</td>
<td>talk like a pirate day</td>
<td>Pearl Harbor Remembrance Day</td>
</tr>
<tr>
<td>Which [Dr. Who] villain has been played by [Roger Delgado], [Anthony Ainley], [Eric Roberts], etc?</td>
<td>The Master</td>
<td>mr. daleks</td>
<td>The Master</td>
</tr>
<tr>
<td>Which early aviator flew in a plane christened [Jason]?</td>
<td>Amy Johnston</td>
<td>jean batten</td>
<td>Icarus</td>
</tr>
</tbody>
</table>

Table 4: Illustrative examples of predictions for the TriviaQA dev set. Questions are annotated with [correct] and [incorrect] entity predictions from EAE, which is most successful when question entities are linked successfully.

<table>
<thead>
<tr>
<th>$K$</th>
<th>Entity acc</th>
<th>Tok acc</th>
<th>Tok PPL</th>
<th>TQA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>59.2</td>
<td>56.7</td>
<td>18.0</td>
<td>40.1</td>
</tr>
<tr>
<td>10</td>
<td>61.7</td>
<td>57.2</td>
<td>11.1</td>
<td>43.1</td>
</tr>
<tr>
<td>100</td>
<td>61.8</td>
<td>57.1</td>
<td>11.0</td>
<td>43.2</td>
</tr>
<tr>
<td>Full (10$^5$)</td>
<td>61.8</td>
<td>56.9</td>
<td>11.0</td>
<td>43.4</td>
</tr>
</tbody>
</table>

Table 5: Impact of varying the number of retrieved entity embeddings ($K$) in the Entity Memory layer at inference on the entity prediction and TriviaQA tasks.

6.3 Top-K over Entity Embeddings

As described in Section 2, EAE uses the top 100 entity memories during retrieval for each mention. Here, we empirically analyse the influence of this choice. Table 5 shows how varying the number of retrieved entity embeddings in the entity memory layer at inference time impacts accuracy of entity prediction and TriviaQA. 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 T5-3B model, while accessing about 3% of the parameters. 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.

7 Comparison to Alternative Entity Representations

EAE is novel in learning entity representations along with the parameters of a Transformer model. Here, we provide a direct comparison between EAE’s learned representations and the externally trained, fixed representations used by ERNIE (Zhang et al., 2019) and KNOWBERT (Peters et al., 2019). Table 6 presents results for the EAE architecture trained with entity memories initialized
from three different sources and two different training scenarios. The initial embeddings are either the TransE-Wikidata embeddings used by ERNIE (Bordes et al., 2013); the Deep-Ed embeddings used by KNOWBERT (Ganea and Hofmann, 2017); or the random embeddings used by EAE. The embeddings are either frozen, following ERNIE and KNOWBERT, or trained along with all other network parameters.\(^8\) Along with the knowledge probing tasks from Section 5, we report performance on the on the 9-way entity typing task from Choi et al. 2018 (Appendix D).

It is clear that learning entity representations is beneficial. However, initializing the entity memory with Deep-Ed embeddings leads to gains on most of the knowledge probing tasks, suggesting that there are potential benefits from combining training regimens. Meanwhile, the best entity typing results come from initializing with Wikidata embeddings, possibly because Wikidata has high quality coverage of the types (‘person’, ‘organization’, ‘location’, etc.) used. Finally, we point out that both ERNIE and KNOWBERT differ from EAE in other ways (see Appendix A). KNOWBERT in particular incorporates WordNet synset embeddings as well as entity embeddings, leading to entity typing accuracy of 76.1. Future work will explore different ways of combining learned embeddings with knowledge from diverse external sources.

As discussed in Section 3, there are significant differences between EAE and KNOWBERT other than the choice of entity representation. In particular, KNOWBERT has an explicit entity-entity attention mechanism. To determine whether this has a significant effect on a model’s ability to model entity-entity relations, we evaluate EAE on TACRED (Zhang et al., 2017) using the cleaned dataset introduced by (Alt et al., 2020).\(^9\) Table 7 shows that EAE outperforms KNOWBERT on the revised and weighted splits introduced by (Alt et al., 2020), although it slightly under-performs on the original setting.\(^10\) This result indicates that EAE, without explicitly entity-entity attention, can capture relations between entities effectively.

### 8 Conclusion

We introduced a new transformer architecture, EAE, which learns entity representations from text along with other model parameters. Our evaluation shows that EAE is effective at capturing declarative knowledge and can be used for a wide variety of tasks – including open domain question answering, relation extraction, entity typing and knowledge probing tasks. Our entity representations influence the answer predictions for open-domain question answering system and are of high quality, compared to prior work such as KNOWBERT.

Our model learns representations for a pre-fixed vocabulary of entities, and cannot handle unseen entities. Future work can explore representations for rare or unseen entities, as well as developing less memory-intensive ways to learn and integrate entity representations. Furthermore, integrating information from knowledge-bases can further improve the quality of entity representation.

---

\(^8\) All experiments use the same 200k entities, at the intersection of the sets used by ERNIE and KNOWBERT. For efficiency, these three comparison systems are trained for 500k steps, rather than the full 1m steps used in Section 5.

\(^9\) Our method follows (Baldini Soares et al., 2019) in using special tokens to mark subject and object, and concatenating their representations to model the relation.

\(^10\) The weighted split weights examples by its difficulty, focusing on correctly predicting difficult examples.
Acknowledgments

We thank many members of Google Research for helpful feedback and discussions, especially Kenton Lee, Dipanjan Das, Pat Verga and William Cohen. We also thank Adam Roberts and Sewon Min for sharing their system outputs.

References


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


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.


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 Ish (alsih) for sublinear time maximum inner product search (mips). In NeurIPS.


A Other Models Evaluated

In this section we describe the models mentioned in Section 4.3 and compared to in Section sections 5 and 7.

A.1 Open-Book Question Answering Systems

Open-Book Open Domain Question Answering Systems are usually comprised of two components: a retriever and a reader. The retriever reads a set of documents from a corpus or facts from a knowledge base. Top retrievals are then fed to the reader which predicts an answer, often through span selection.

BM25+BERT (Lee et al., 2019) uses the non-parametric BM25 as its retriever and a BERT-based reader to predict an answer through span selection. ORQA (Lee et al., 2019) uses two BERT-based models to retrieve relevant passages and a BERT-base reader to predict the answer span.

Graph Retriever (Min et al., 2019b)’s retriever can perform Graph-based, Entity-based and Text-match retrieval. These different retrieval modalities are then fused in representations that serve as input to a span-selection reader.

A.2 Other Models

BERT (Devlin et al., 2018) is a transformer, pre-trained using masked language modelling. 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.

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.

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. ‘SSM’ refers to salient span masking, indicating that prior to fine-tuning on open domain question answering the model was fine-tuned using salient span masking, which bears resemblances to our mention masking.

KnowBERT (Peters et al., 2019) KnowBERT is a BERT-base transformer that embeds multiple knowledge bases to improve performance in a variety of tasks. The integration of this information is done through a Knowledge Attention and Recontextualization component, which can be seen as a small transformer that is run on the pooled mention representations of potential entities. KnowBERT uses this layer to embed entity representations from Wikipedia as well as Wordnet graph information. In contrast with our work, KnowBERT starts from the BERT checkpoint, does not train with a knowledge-focused objective such as our mention-masking input function and uses pre-existing representations when integrating the information from knowledge bases. In addition, KnowBERT relies on a fixed, pre-existing candidate detector (alias table) to identify potential candidates and entities for a span while our model learns mention detection.

ERNIE (Zhang et al., 2019) ERNIE is a BERT-base transformer that takes as additional input the list of entities in the sentence. Multi-head attention is performed on those entities before they are introduced in they are aggregated with the token representations. In addition to BERT’s pre-training objective, ERNIE also masks entities and trains the model to predict them. In contrast with both KnowBERT and EAE, ERNIE takes entity linking as an input rather than learning to do it inside the model. In contrast with our approach, ERNIE uses pre-existing entity representations that are fixed during training.

B 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 2.2. 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. We did not run extensive hyper-parameter tuning for pre-training due to computational cost. We train on 64 Google Cloud TPUs for all our pre-training experiments. All pre-training experiments took between 2 days and a week to complete.

Open Domain Question Answering

Open Domain QA Preprocessing We annotate each question with proper-name mentions using the mention detector from Section 2.2.

When the answer is an entity, in our entity vocabulary, we link the answer string to an entity ID using the SLING phrase table (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. Table 8 shows the share of answers that were linked by our linking procedure for the TriviaQA and WebQuestions datasets. The test set for TriviaQA is not public, hence the missing number.

Hyper-parameters For TriviaQA, we fine-tune the entire model using a learning rate of 5e-6, a batch size of 64 and performing 50,000 training steps. For WebQuestions, we set the learning rate to 3e-5, the batch size to 32 and train for 700 steps. Fine-tuning was done on 4 Google Cloud TPUs. In both cases, we searched the learning rate over $\{5 \times 10^{-5}, 3 \times 10^{-5}, 10^{-5}, 5 \times 10^{-6}, 3 \times 10^{-6}\}$ and the batch size in $\{32, 64, 128\}$ and selected the model based on validation performance.

D Entity Typing

We describe the procedure to obtain the Typing results of Table 6.

Open Entity Processing We use the Ultra-fine entity typing dataset introduced in (Choi et al., 2018). As is done in Zhang et al. (2019); Peters et al. (2019) we limit the task to the 9 generic types (‘person’, ‘group’, ‘organization’, ‘location’, ‘entity’, ‘time’, ‘object’, ‘event’ and ‘place’) and use the Micro-F1 metric. The dataset is comprised of 5994 examples, equally divided between train, development and test. We pass the sentences to our model without extra annotations for mention annotations or entity linking, relying instead on the model’s own predictions. To predict a type for the span, we take the span representation and project it to the 9 classes.

Hyper-parameters Since we have no mention boundary or linking information, we freeze the entity embeddings and the mention detection parameters in fine-tuning. We used a learning rate of 3e-5, a batch size of 32 and trained for 700 steps. We also used label smoothing with a value of 0.1. We searched the learning rate and the batch size between the same bounds as for the open-domain question answering tasks. For every model, we ran with five different seeds and selected the best model based on validation performance before running on test. We selected the threshold to compute F1 based on validation scores.
Table 9: Comparing prediction overlap and oracle accuracy onTriviaQA. 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.

<table>
<thead>
<tr>
<th>Systems</th>
<th>Oracle Acc.</th>
<th>Pred Overlap (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T5 &amp; EAE</td>
<td>55.9</td>
<td>29.3</td>
</tr>
<tr>
<td>T5 &amp; GR</td>
<td>66.4</td>
<td>30.1</td>
</tr>
<tr>
<td>EAE &amp; GR</td>
<td>64.6</td>
<td>33.6</td>
</tr>
<tr>
<td>ORQA &amp; GR</td>
<td>63.8</td>
<td>39.6</td>
</tr>
</tbody>
</table>

Table 10 illustrates additional 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.

Example (i) shows another case where our model fails to handle dates. 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’. In the second 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.

Example (iii) shows another case where the correctly linked entities enable EAE to correctly answer the prompt while T5 predicts a different director’s name. 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.

F  Comparing QA paradigms

We compare the performance of four systems (ORQA, GraphRetriever (GR), T5, EAE) on the TriviaQA Unfiltered-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 9). 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.

G Additional Examples of TriviaQA Predictions

Table 10 illustrates additional 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.

Example (i) shows another case where our model fails to handle dates. 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’. In the second 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.

Example (iii) shows another case where the correctly linked entities enable EAE to correctly answer the prompt while T5 predicts a different director’s name. 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.

13https://competitions.codalab.org/competitions/17208
<table>
<thead>
<tr>
<th>Questions with no proper name mentions</th>
<th>Answer</th>
<th>T5 Prediction</th>
<th>EaE Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>i What links 1st January 1660 and 31st May 1669?</td>
<td>First and last entries in Samuel Pepys's diaries</td>
<td>they were the dates of the henry viii’s last bath</td>
<td>Anglo-Dutch Wars</td>
</tr>
<tr>
<td>ii Which radioactive substance sometimes occurs naturally in spring water?</td>
<td>radon</td>
<td>radon</td>
<td>radon</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Questions with only [correctly] linked entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>iii Who directed the 2011 [Palme d’Or] winning film ‘The Tree Of Life’?</td>
</tr>
<tr>
<td>iv Name the year: [Hirohito] dies; The [Exxon Valdez] runs aground; [San Francisco] suffers its worst earthquake since 1906.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Questions with {incorrectly} linked entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>v Which car manufacturer produces the [Jimmy] model?</td>
</tr>
<tr>
<td>Jimmy → Marcos Engineering Q1637323</td>
</tr>
<tr>
<td>vi Where do you find the [Bridal Veil], [American], and [Horseshoe Falls]?</td>
</tr>
<tr>
<td>Bridal Veil → Veil (Garment) Q6497446</td>
</tr>
</tbody>
</table>

Table 10: Additional examples of question, answer and model predictions for the TriviaQA Unfiltered dev set.

Examples (v) and (vi) shows other failure modes when EAE fails to correctly predict the entities in the question. In example (vi) the entity is not available, though this not cause the model to err, likely thanks to the two other correctly predicted question entities. Example’s (vi) typo (from ‘Jimny’ to ‘Jimmy’) is particularly interesting: when fixed, EAE links ‘Jimmy’ correctly and predicts the right answer.