A Unified Encoder-Decoder Framework with Entity Memory

Zhihan Zhang[1], **Wenhao Yu**[1], **Chenguang Zhu**[2], **Meng Jiang**[1]
[1]University of Notre Dame, Notre Dame, IN, USA
[2]Microsoft Cognitive Services Research, Redmond, WA, USA
[1]{zzhang23, wyu1, mjiang2}@nd.edu; [2]chezhu@microsoft.com

Abstract

Entities, as important carriers of real-world knowledge, play a key role in many NLP tasks. We focus on incorporating entity knowledge into an encoder-decoder framework for informative text generation. Existing approaches tried to index, retrieve, and read external documents as evidence, but they suffered from a large computational overhead. In this work, we propose an **E**ncoder-**D**ecoder framework with an entity **Mem**ory, namely EDMem. The entity knowledge is stored in the memory as latent representations, and the memory is pre-trained on Wikipedia along with encoder-decoder parameters. To precisely generate entity names, we design three decoding methods to constrain entity generation by linking entities in the memory. EDMem is a unified framework that can be used on various entity-intensive question answering and generation tasks. Extensive experimental results show that EDMem outperforms both memory-based auto-encoder models and non-memory encoder-decoder models.

1 Introduction

A large amount of real-world knowledge is related to entities, *e.g.*, persons, nations, and events. Entity knowledge is the information describing facts and attributes related to entities. Many entity-intensive NLP tasks require models obtain entity knowledge to generate informative outputs, such as answering factual questions (Kwiatkowski et al., 2019), explaining claims (Onoe et al., 2021), or making informative conversations (Dinan et al., 2019). Pre-trained encoder-decoder models can be directly applied on such entity-intensive tasks (Ye et al., 2020; Roberts et al., 2020), but their ability to store and use knowledge is still questionable (Lewis et al., 2021; Wang et al., 2021). A popular approach to incorporate knowledge into the generation process is retrieving evidence documents from external sources (Lewis et al., 2020b; Izacard and Grave, 2021; Oguz et al., 2020; Yu et al., 2022c). However, they suffer from significant computational overheads in indexing, retrieving, and reading a large number of extra documents (Lee et al., 2021; de Jong et al., 2022). Therefore, it is important to give encoder-decoder models access to entity knowledge without sacrificing too much efficiency.

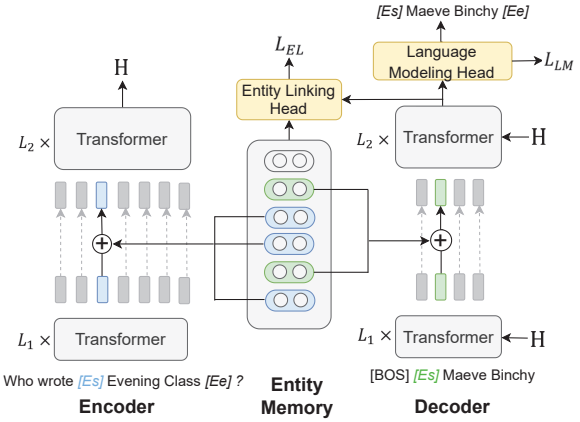

Figure 1: An overview of the EDMem framework. **H** denotes the final hidden states of the encoder.

Recently it has been proposed to use an in-model memory to augment auto-encoder models with entity knowledge on entity linking tasks (Févry et al., 2020; Verga et al., 2021; Sun et al., 2021). The entity memory stores entity knowledge as dense vectors which can be directly incorporated into the hidden states of Transformer models (Vaswani et al., 2017), with no need to encode extra text. However, the auto-encoder framework in previous approaches can only select entities from a pre-defined entity vocabulary. Hence, they are not able to give an entity outside the vocabulary, nor to *generate* answers or text beyond a single entity.

In this paper, we propose a novel **E**ncoder-**D**ecoder framework with an entity **Mem**ory (EDMem), as shown in Figure 1. EDMem is a unified framework on various entity-intensive QA and generation tasks, in which we train an entity memory for efficient knowledge incorporation. First,

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 689–705
December 7-11, 2022

EDMem is pre-trained on Wikipedia documents, where it learns entity embeddings in the memory along with an encoder-decoder model. EDMem learns to select relevant entities from the memory via an entity linking objective, and learns to generate answers using entity knowledge via a language modeling objective. Second, to precisely generate entity names, we design three decoding methods that utilize the entity linking ability of EDMem in its generation process, when we fine-tune it on downstream tasks. These include (1) free-form: left-to-right generation with entity identifiers; (2) static entity linking: first select entities by entity linking, build prefix trees for the selected entities, and then perform constrained entity generation using the trees; (3) dynamic entity linking: select entities on-the-fly for constrained entity generation.

We conduct experiments on two popular testbeds of entity knowledge: open-domain QA and entity-intensive generation. With the incorporation of entity knowledge, EDMem outperforms non-memory encoder-decoder models on both tasks, and it retains the efficiency advantage of closed-book (*i.e.*, non-retrieval) models. Compared to memory-based auto-encoders, EDMem achieves both higher overall accuracy (+9%) and better entity precision (+8%) on open-domain QA datasets, and it generates high-quality text from the memory-supported decoder on generation datasets when auto-encoders fail to do so. To summarize, EDMem is the first knowledge-augmented closed-book framework to perform both tasks in a *unified* manner.

2 Related Work

Closed-Book Models Closed-book models are pre-trained models that store knowledge in their own parameters. For example, COMET (Bosselut et al., 2019) fine-tuned GPT2 (Radford et al., 2018) to construct knowledge graphs by generating commonsense triples. Recently, fine-tuned BART (Lewis et al., 2020a) or T5 (Raffel et al., 2020) models are proved to be competitive on open-domain QA (Ye et al., 2020; Roberts et al., 2020). Therefore, closed-book models are able to memorize some entity knowledge after pre-trained on massive data. However, studies showed that closed-book models just recalled similar inputs and answers in their pre-training corpus (Wang et al., 2021), and their performances were behind open-book models.

Open-Book Models Open-book models first retrieve evidence documents from external corpora and read these documents to predict an answer (Chen et al., 2017). REALM (Guu et al., 2020) proposed a self-supervised approach to pre-train a retriever-reader model. DPR (Karpukhin et al., 2020) devised a contrastive objective to train a dense bi-encoder retriever on open-domain QA. Subsequent approaches combined DPR with a generative objective to build large, powerful models on open-domain QA and generation tasks (Lewis et al., 2020b; Izacard and Grave, 2021; Sachan et al., 2021; Yu et al., 2022a). However, open-book models have to process the raw text of all retrieved documents, which leads to extremely long inference time. Besides, additional overheads are brought by loading the document index and retrieving evidence documents for each example.

Entity Memory EaE (Févry et al., 2020) was the first to pre-train an entity memory with an auto-encoder framework to perform entity prediction on open-domain QA. FILM (Verga et al., 2021) followed EaE and added a fact memory containing representations of Wikidata triples. To better encode relational knowledge, OPQL (Sun et al., 2021) learned latent relational representations for arbitrary entity pairs. Recent work focused on learning a huge mention-level memory (~150M entries) with extensive pre-training (de Jong et al., 2022) or leveraging the entity memory in domain adaptive training (Kang et al., 2022). These models are all based on an auto-encoder framework. Thus, they are able to predict entities IDs but would fail to generate any non-entity answers or sentences. There is a preprint paper contemporaneous to our work which trained a memory with an encoder-decoder model (Chen et al., 2022). However, it used QA pairs as memory entries instead of entities, limiting its application to QA tasks. Besides, their memory is much heavier (60M entries) than ours (1M).

3 Proposed Framework

Suppose we have a pre-defined vocabulary of N entities $\mathcal{E} = \{e_1, \ldots, e_N\}$. A mention is the actual tokens in context which refer to an entity. The set of all mentions in the corpus is denoted as $\mathcal{M}$. Thus, there is a global alias table $\mathcal{T} : \mathcal{E} \rightarrow 2^{\mathcal{M}}$, where each entity is mapped to all its mentions. The input of EDMem is a sequence of tokens $\boldsymbol{x}$ of length S, and the target output is another sequence $\boldsymbol{y} = [y_1, \cdots, y_T]$ of length T. Both sequences

contain a pre-labeled set of mentions. Each mention refers to an entity in $\mathcal{E}$. We add two special tokens $[E_s]$ and $[E_e]$ to represent "entity start" and "entity end" boundaries of a mention, *e.g.*, "$[E_s]$ *Brett Hart* $[E_e]$ *is the president of the* $[E_s]$ *United Airlines* $[E_e]$". These special tokens come from either Wikipedia hyperlinks (in pre-training, §3.3) or an entity linking model (in fine-tuning, §3.4).

3.1 Architecture

An overview of EDMem is presented in Figure 1. The framework has a transformer encoder, a transformer decoder, an entity memory, and two prediction heads. Both the encoder and decoder have two parts: ($L_1\times$) lower layers and ($L_2\times$) upper layers. Transformer layers in EDMem have the same architecture with BART (Lewis et al., 2020a). At the end of lower layers, EDMem is allowed to use the hidden states as a query to access the entity memory. The knowledge representation obtained by each memory access is summed and normalized with the hidden states before performing further reasoning in upper layers. Two prediction heads use the final hidden states of the decoder for prediction: an LM head for token prediction and an entity linking head for entity prediction (Details are in §3.3). In practice, we follow EaE (Févry et al., 2020) to set $L_1 = 4$ and $L_2 = 8$.

3.2 Entity Memory

The entity memory contains a large embedding table, which stores the embeddings of entities in $\mathcal{E}$. Intuitively, an entity embedding contains the contextual information around all mentions of the entity in Wikipedia documents. During encoding and decoding, EDMem queries the entity memory whenever it encounters a mention. It recognizes mentions by identifying the $[E_s]$ token. EDMem takes the hidden state of the $[E_s]$ token as query to retrieve relevant knowledge from the entity memory by attending to the entity embedding table (bias terms are omitted):

$$\mathbf{h}_s^{ent} = \mathbf{W}_{out}(\sum_{i=1}^{N} \alpha_i \cdot \mathbf{e}_i), \tag{1}$$

$$\text{where } \alpha_i = \frac{\exp\left(\mathbf{e}_i^\intercal \mathbf{W}_{in} \mathbf{h}_s^{low}\right)}{\sum_{j=1}^{N} \exp\left(\mathbf{e}_j^\intercal \mathbf{W}_{in} \mathbf{h}_s^{low}\right)}. \tag{2}$$

$\mathbf{e}_i$ is the embedding of entity e_i. $\mathbf{h}_s^{low}$ denotes the hidden state of the $[E_s]$ token (from lower encoder/decoder layers). $\mathbf{h}_s^{ent}$ is the aggregated entity representation, which is summed and normalized with $\mathbf{h}_s^{low}$ to put into upper layers. $\mathbf{W}_{in}$ and $\mathbf{W}_{out}$ are linear projection layers for dimension matching. Following EaE, during inference, we aggregate the entity representaion of top 100 entities (sorted by α_i) instead of attending to all N entities.

3.3 Pre-Training

3.3.1 Pre-Training Corpus

We pre-train EDMem on the whole Wikipedia corpus. All documents are split into 128-token passages. In addition, we set a 10-token sliding window between passages to avoid an entity being split into two adjacent chunks. Such a setting yields a total of 39M passages, of which we hold out 0.5% of them as the validation set during pre-training. We leverage Wikipedia hyperlinks as gold annotations of 249M mentions and their linked entities. Since hyperlinks do not cover all mentions in text, we heuristically label missing mentions to create more training signals for the entity memory. We use the alias table $\mathcal{T}$ to label all mentions in a Wikipedia page if they match either (1) a linked entity in the same page, or (2) the title entity of this page. This leads to a total of 468M mentions in the pre-training corpus. We collect 1M most frequently linked entities to form the entity vocabulary $\mathcal{E}$. More details can be found in Appendix A.

3.3.2 Pre-Training Objective

Our pre-training objective is a combination of language modeling and entity linking. For language modeling objectives, we randomly corrupt parts of the input sequence and train EDMem to reconstruct the original sequence. We adopt two kinds of sequence corruption: *random token masking* and *salient span masking*. In random token masking, each token has a probability of P_{rtm} to be replaced by a [MASK] token. Salient span masking is adapted from (Guu et al., 2020), where each mention has a probability of P_{ssm} that all tokens within the mention are replaced by [MASK]. Such explicit masking of whole mention names encourages EDMem to rely on the entity memory in predicting mentions, which facilitates the learning of entity embeddings. The LM head performs token prediction through a linear-softmax layer, and the LM loss is the negative log-likelihood of the target sequence: $L_{LM} = -\sum_{j=1}^{T} \log P(y_j|\boldsymbol{x}, y_{1:j-1})$.

EDMem utilizes direct supervision signals to the entity memory for entity representation learning. The entity linking loss is applied each time it queries the entity memory. Besides in the middle of

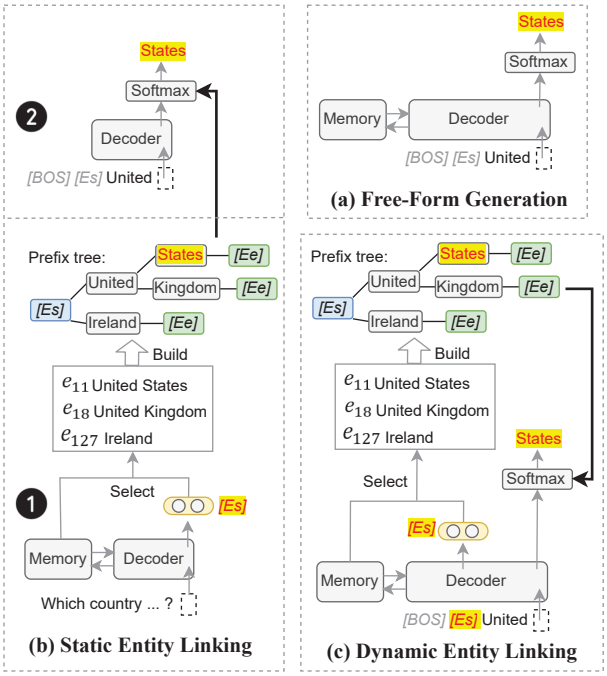

Figure 2: Three decoding methods in downstream tasks.

the encoder and decoder, EDMem queries the memory in the entity linking head, as shown in Figure 1. The entity linking head predicts the corresponding entity using the hidden states of each mention, the same as Equation (2). We use a cross-entropy loss to maximize the attention weights of the labelled entities: $L_{EL} = -\sum_m \log \alpha_i$, where m is a mention in the input or output sequence that is linked to the i-th entity in $\mathcal{E}$. The final loss function is $L_{LM} + \lambda_{EL} L_{EL}$, where the coefficient λ_{EL} is a hyper-parameter.

3.4 Fine-Tuning

EDMem is fine-tuned on downstream tasks via an LM objective and an entity linking objective. The LM objective is to maximize the probability of the task-specific output. The entity linking objective links mentions to entities in the memory, the same as pre-training. Mention boundaries are pre-labeled using an state-of-the-art entity linking model (Li et al., 2020). In entity-intensive downstream tasks, the entity memory assists sequence generation by not only providing entity knowledge but also generating entity names. Thus, we design three decoding settings to let the entity linking objective assist sequence generation. A sketch of different settings is given in Figure 2.

Free-Form Generation In this setting, the model generates the output sequence entirely based on the probability given by the LM head. This includes the special tokens $[E_s]$ and $[E_e]$ which indicate an access to the memory. There is no constraint on what tokens to generate between $[E_s]$ and $[E_e]$, *i.e.*, the subsequence $[E_s], y_i, \cdots, y_j, [E_e]$ may not be a valid entity name in the entity vocabulary. One advantage is that the model processes the entity knowledge in a latent manner, which does not explicitly affect the probability distribution of the language model. However, this may affect the model's performance in tasks where entity names are strictly required, *e.g.*, open-domain QA tasks where exact match is used as evaluation.

Static Entity Linking Static entity linking explicitly restricts the model to generate entity names for QA. Here, the decoding process is divided into two steps: *entity linking* and *constrained generation*. First, given a question, the model selects one or multiple entities as references. As shown in Figure 2(b), the question with an appended $[E_s]$ token as a placeholder is passed into the decoder, and the entity linking head is trained to predict the entity ID of the gold answer[1]. Then we have the selected top-k entities for each test question. We restrict the generation space to the top-k entities when the model is trying to generate an entity name. To achieve this, inspired by (Cao et al., 2021), we build a prefix tree for k entities for each test example. The prefix tree tells the model which tokens are allowed to generate given a prefix (*i.e.*, previous generated tokens). When the model generates an $[E_s]$ token, we restrict the following generated tokens to be one of the k entity names (*i.e.*, one of the paths in the prefix tree). In this way, the model can either generate an entity answer (by generating $[E_s]$ and traversing the pre-built prefix tree), or generate a non-entity answer (if no $[E_s]$ token is generated). Readers can refer to (Cao et al., 2021) for more implementation details.

Dynamic Entity Linking Static entity linking is applicable only when the downstream task can be converted into an entity linking objective. Another way to generate entities is to predict the entities *on-the-fly*. After each time the model generates an $[E_s]$ token, the entity linking head predicts top-k entities using the hidden state of $[E_s]$ based on previous generated tokens, as shown in Figure 2(c). This differs from static entity linking, where the model makes entity predictions solely dependent

[1]Training examples with non-entity answers are discarded.

on the input sequence. A prefix tree of the names of top-k entities is also built on-the-fly for constrained entity generation.

4 Experiments

We test our EDMem framework on two testbeds of entity knowledge: open-domain QA and entity-intensive generation tasks.

4.1 Open-Domain QA

4.1.1 Data

Open-domain QA is a task where models are required to answer questions without any provided evidence. Questions are usually related to real-world facts and entities. We test EDMem on three popular datasets: Natural Questions (NQ) (Kwiatkowski et al., 2019), TriviaQA (TQA) (Joshi et al., 2017) and WebQuestions (WQ) (Berant et al., 2013). We follow the in-house splits introduced by (Lee et al., 2019). We also report on dev set of the TQA official split to compare with EaE (Févry et al., 2020). We report exact match (EM) scores on these datasets.

We mainly compare with previous closed-book models (*i.e.*, models without evidence retrieval), including traditional encoder-decoder models like BART (Lewis et al., 2020a) and T5 (Raffel et al., 2020), and memory-based auto-encoder models like RELIC (Ling et al., 2020), EaE, and FILM (Verga et al., 2021). Besides, We pre-train two ablations of EDMem. EncMem is composed of an encoder and an entity memory, and is trained via the same objectives as EDMem. EncDec removes the entity memory from EDMem, and is trained via the same LM objectives. We also list the performance of state-of-the-art open-book models (*i.e.*, models with evidence retrieval to assist prediction) for reference, such as REALM (Guu et al., 2020), RAG (Lewis et al., 2020b), and FiD (Izacard and Grave, 2021). We test three variants of EDMem, *i.e.*, free-form generation (-free), static entity linking (-stat.) and dynamic entity linking (-dyn.).

4.1.2 Results

Experimental results on open-domain QA datasets are listed in Table 1. With the same architecture, EncDec outperforms BART due to the additional salient span masking pre-training. Memory-based auto-encoder models like EaE and EncMem perform entity linking to provide answers. They outperform traditional encoder-decoder models by a large margin on TQA and WQ. However, target answers are mainly entities on both datasets[2]. While on NQ where there are fewer entity answers, the performance of EncMem is similar to BART-Large. Compared to baselines, the free-form EDMem already outperforms memory-based auto-encoder models and traditional encoder-decoder models on TQA and NQ. EDMem-static and EDMem-dynamic explicitly copy entity names into the generated answers, which further improves EDMem's performance, especially on TQA and WQ datasets where a larger portion of answers are entities. Overall, EDMem improves the best of closed-book baselines by 9%/6%/6% on TQA/NQ/WQ, respectively. Although closed-book models are still behind open-book models in general, our approach shows that by combining the merits of encoder-decoder models like BART and entity linking models like EaE, the performance of EDMem is getting closer to open-book approaches and even outscores the open-book model REALM on WQ.

Model	TQA		NQ	WQ
	In-House Test	Dev	Test	Test
Closed-Book Models				
BART-Large*	25.02	27.28	24.82	29.23
T5-Large*	-	28.70	28.50	30.60
EncDec*	27.54	30.01	25.96	29.38
RELIC†	35.70	-	-	-
EaE†	-	43.20	-	39.00
FILM†	29.10	-	-	-
EncMem†	41.01	42.00	25.54	38.88
EDMem-free	42.24	43.31	<u>29.14</u>	36.47
EDMem-stat.	**46.19**	**47.23**	**30.19**	**41.44**
EDMem-dyn.	<u>43.82</u>	<u>44.44</u>	27.70	<u>39.52</u>
Open-Book Models				
REALM	-	-	40.40	40.70
RAG	56.80	-	44.50	45.20
FiD	67.60	-	51.40	47.64

Table 1: Exact match scores on open-domain QA datasets. **Bold** scores and <u>underlined</u> scores are the best and second best results among closed-book models. (*traditional encoder-decoder models, †memory-based auto-encoder models)

4.1.3 Entity/Non-Entity Answers

To further investigate the improvements of EDMem over previous closed-book models, we calculate EM scores on two subsets divided *w.r.t.* the answer type (*i.e.*, entity answers and non-entity answers).

[2]89% of the answers in WQ test set are Wikipedia entities, and the ratio is 84%/70% for TQA/NQ.

Model	TQA						NQ			WQ		
	In-House Test			Official Dev			Test			Test		
	Total	Ent.	Non-Ent.	Total	Ent.	Non-Ent.	Total	Ent.	Non-Ent.	Total	Ent.	Non-Ent.
BART	25.02	27.92	9.31	27.28	30.52	9.70	24.82	28.54	15.95	29.23	32.28	5.24
EaE	-	-	-	43.20	51.43	0.00	-	-	-	39.00	42.86	0.00
EncMem	41.01	48.58	0.00	42.00	49.74	0.00	25.54	36.24	0.00	38.88	43.82	0.00
EDMem-free	42.24	48.27	**9.59**	43.31	49.44	**10.10**	29.14	34.32	**16.19**	36.47	40.16	**7.42**
EDMem-stat.	**46.19**	**53.82**	4.88	**47.23**	**54.86**	5.85	**30.19**	**37.38**	13.04	**41.44**	**46.26**	3.49
EDMem-dyn.	43.82	50.64	6.81	44.44	51.10	8.29	27.70	33.41	14.07	39.52	44.09	3.49

Table 2: Exact match scores on entity answers ("Ent.") and non-entity answers ("Non-Ent.") in open-domain QA.

If an answer can be directly linked to a Wikipedia entity according to Google's SLING (Ringgaard et al., 2017) phrase table, it is counted as an entity answer, otherwise a non-entity answer. As shown in Table 2, as an entity linking model, EaE cannot predict non-entity answers; and as an encoder-decoder model, BART is able to generate a portion of non-entity answers while its accuracy on entity answers is much lower than EaE due to the lack of entity knowledge. EDMem incorporates entity knowledge into an encoder-decoder model, making it competitive in both entity answers and non-entity answers. However, the free-form generation variant is not as accurate on entity answers as EaE, because it may generate any form of answers while EaE always predicts a valid entity name. The entity linking variants, on the other hand, remedy this issue by setting constraints on generating entity names, either statically or dynamically. Such approaches improve the model performance on entity answers although sacrificing some performance on non-entity ones, and achieve the best overall performance on all answers. Besides, the best setting of EDMem outperforms EaE in entity answers as well, presumably due to a larger number of transformer layers trained in the encoder-decoder architecture, compared to its auto-encoder counterpart.

4.1.4 Inference Efficiency

We compare the time efficiency of different models during inference in Table 3. We run EDMem, BART and the open-book model FiD on the test set of TQA (11K questions) with 8×V100 GPUs. Compared to BART, EDMem needs to access a large entity memory multiple times, which slows down the inference time from 10s to 28s. However, such a time cost is much smaller than the gap between closed-book models and the open-book FiD (85min). In the open-book setting, the model needs to (1) load the pre-computed index from disk to the RAM, (2) retrieve evidence documents from the index, and (3) read all evidence documents to generate an answer. In addition to the overhead caused by accessing the index, the model needs to encode the raw text of all evidence documents (*i.e.*, 100 documents for FiD) before generating an answer with the decoder, but EDMem and BART only needs to encode the question itself. Thus, EDMem is able to achieve significant improvement over traditional encoder-decoder models while retaining the efficiency advantage of closed-book models.

Type	Model	T_{ind}	T_{ret}	T_{pred}	EM
Closed-book	BART	0	0	17s	25.02
	EDMem-free	0	0	28s	42.24
	EDMem-dyn.	0	0	48s	43.82
	EDMem-stat.	0	0	59s	46.19
Open-book	FiD	29min	15min	41min	67.60

Table 3: Inference time on TriviaQA test set. T_{ind} is the time for loading the index, which is a fixed amount of time; T_{ret} and T_{pred} denote time for document retrieval and answer prediction, which will linearly increase as the number of test examples increases. Inference time of EDMem-stat. is the accumulation of the entity linking step and the constrained generation step.

4.1.5 Size of Entity Memory

We compare the performance of EDMem and its auto-encoder variant EncMem based on different sizes of the entity memory. We randomly mask out entities from the original 1M vocabulary and re-train the model. Embeddings of masked entities do not participate in computing attention while accessing the memory. According to the curves in Figure 3, due to EDMem's ability of closed-book generation, it is less sensitive to the size of the entity memory, resulting in a smaller slope when less entities are visible. Particularly, EDMem is still able to generate many correct answers even

Dataset	Model	ROUGE-1	ROUGE-2	ROUGE-L	F1	BERTScore	Entity Coverage	
							Total	Unseen
MSMARCO	BART-Large	56.72	37.62	53.26	53.86	89.34	43.87	22.40
	EDMem-free	**57.67**	**39.45**	**54.45**	**55.08**	**89.40**	45.53	25.33
	EDMem-dyn.	55.96	37.42	52.84	52.94	88.49	**51.28**	**28.49**
	FiD (open-book)	60.54	42.96	57.22	58.12	79.79	49.63	32.58
CREAK	BART-Large	32.87	14.93	30.34	30.20	81.21	46.87	14.98
	EDMem-free	**33.81**	**16.49**	**31.78**	**31.34**	**86.11**	49.06	16.65
	EDMem-dyn.	32.70	15.75	30.68	30.32	85.76	**49.90**	**18.76**
	FiD (open-book)	35.96	18.11	33.54	33.57	81.02	51.02	21.09
ELI5	BART-Large	25.87	5.89	22.99	19.38	**81.45**	29.18	14.83
	EDMem-free	27.14	5.70	23.24	20.19	80.78	38.76	18.61
	EDMem-dyn.	**27.48**	**7.14**	**23.97**	**20.66**	81.17	**45.66**	**23.31**
	FiD (open-book)	25.06	5.98	22.12	18.48	80.96	28.06	16.96
WoW	BART-Large	19.52	3.42	**17.22**	15.37	81.92	11.78	4.05
	EDMem-free	18.92	3.50	16.52	15.28	83.19	16.71	6.83
	EDMem-dyn.	**19.54**	**4.00**	17.01	**15.51**	**83.24**	**17.29**	**7.81**
	FiD (open-book)	22.72	6.43	20.16	18.48	81.06	17.91	11.99

Table 4: Results on entity-intensive generation datasets. **Bold** scores are best results among closed-book models.

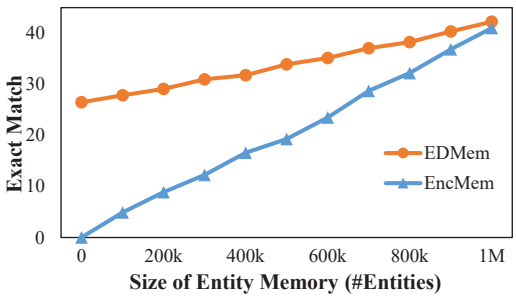

Figure 3: TQA performance of EDMem and EncMem on different memory sizes.

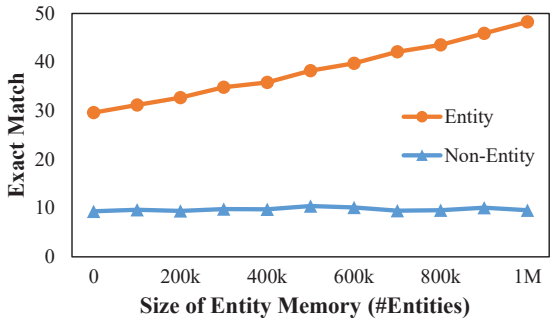

Figure 4: TQA performance of EDMem on entity and non-entity answers, trained with different memory sizes.

when we remove the whole memory. In contrast, EncMem can only predict random entities when the entire memory is masked, which leads to a score close to zero. These results show the advantage of encoder-decoder models over auto-encoder models when jointly trained with an entity memory, especially on low-resource scenarios.

In addition, we also illustrate the performance trend of EDMem on entity answers and non-entity answers in TQA. When all entities are masked, the model deteriorates to its non-memory variant EncDec. As more entity knowledge are available, EDMem performs better in predicting entity answers, while its generation performance remains consistent. These results show the advantage of memory-based EDMem over traditional encoder-decoder models on entity-intensive task comes from the incorporation of entity knowledge from the entity memory.

4.2 Entity-Intensive Generation

4.2.1 Data

To test EDMem's ability in generating longer sentences with entities, we perform experiments on several generation datasets with rich entity mentions. We choose Wizard of Wikipedia (WoW) (Dinan et al., 2019) for knowledge-aware dialogue, MSMARCO-NLGen (Nguyen et al., 2016) and ELI5 (Fan et al., 2019) for abstractive question answering, and CREAK (Onoe et al., 2021) for claim explanation[3]. For MSMARCO and CREAK, we report results on the official dev set while holding out 10% training data for validation. For ELI5, to keep a reasonable density of entities, we filter a subset of 85K data where the input and output are

[3]Given a claim and a true/false judgment, the model generates an explanation about why the claim is true/false.

Model	MSMARCO			CREAK			ELI5			WoW		
	Flu.↑	Rel.↑	Cor.↑	Flu.↑	Rel.↑	Cor.↑	Flu.↑	Info.↓	Rea.↓	Flu.↑	Info.↓	Rea.↓
BART-Large	2.90	1.93	1.57	2.97	1.82	1.63	2.77	2.26	2.34	2.70	2.45	2.51
EDMem-free	2.90	**2.35**	**2.38**	2.99	**2.35**	**2.05**	2.86	<u>1.97</u>	<u>2.06</u>	<u>2.83</u>	**1.86**	**2.14**
EDMem-dynamic	2.87	**2.45**	**2.49**	2.97	**2.32**	**2.04**	2.86	**1.75**	**1.92**	2.80	**1.75**	**2.03**
Human	-	-	-	-	-	-	2.75	2.01	2.19	2.68	1.66	1.86

Table 5: Human evaluation results. "Flu.", "Rel", "Cor." stand for fluency, relevance and correctness, of which scores are given on a 1-3 scale. "Info." and "Rea." stand for informativeness and reasonability, of which 4 sentences are ranked #1 – #4. ↑ indicates higher values are better results. ↓ indicates lower values are better results. We run paired sample t-test comparing EDMem with BART. **Bold** scores indicate significant difference with p-value $<$ 0.01, and <u>underlined</u> scores indicate p-value $<$ 0.05.

both no longer than 75 tokens. Detailed dataset settings are provided in Appendix D.2.

We report ROUGE (Lin, 2004) and unigram F1 scores, as well as BERTScore (Zhang et al., 2020a) for semantic-based evaluation. We also include metrics on evaluating entity generation. Given the entities in the ground-truth as reference, we calculate the coverage ratio of reference entities in the model-generated output. We also consider the mentions of these entities as correct matches, according to the alias table $\mathcal{T}$. To avoid cases where entities in the output can be directly copied from the input[4], we report the coverage ratio of *unseen entities*, *i.e.*, entities in the ground-truth output that do not exist in the input.

4.2.2 Results

Auto-encoder models like EaE are not applicable on these datasets, thus we compare our EDMem to the traditional encoder-decoder model BART. As shown in Table 4, the free-form EDMem outperforms BART on both reference-based metrics (ROUGE, F1, BERTScore) and entity coverage scores. This indicates that the entity knowledge in the entity memory helps generate sentences with desired entities and correct entity-related information. Since these datasets cannot be directly converted to an entity linking setting, EDMem-static is not applicable here. The dynamic entity linking variant outperforms the free-form variant and BART in entity coverage scores on all datasets, while it does not sacrifice much language fluency in reference-based metrics. We find that both EDMem variants outscore BART on entity coverage by a large margin (up to 56% on overall and up to 93% on unseen), which indicates much stronger ability of EDMem models in entity generation.

[4]For example, in CREAK dataset, the topic entity of the input claim usually appears in the explanation as well.

4.2.3 Human Evaluation

To test whether the model generations are reasonable to humans, we leverage Amazon's MTurk platform to conduct human evaluation. For each dataset, we sample 50 data examples with generations from BART and EDMem. We ask three annotators to evaluate each example on *fluency* and two knowledge-related metrics. For CREAK and MSMARCO, knowledge-related metrics are topic *relevance* and factual *correctness*, given the ground-truth as reference. For ELI5 and WoW, since the ground-truth is not the only possible answer to the context, we use *informativeness* and *reasonability* as knowledge-related metrics, and we also evaluate the human-written answers. Detailed descriptions of these metrics are in Appendix F. When evaluating *informativeness* and *reasonability*, annotators are asked to **rank** these generations from #1 – #4, thus lower rankings indicate better results.

As shown in Table 5, EDMem generates more informative and factually correct sentences, compared to BART which lacks knowledge incorporation from the entity memory. Besides, such knowledge incorporation does not harm the fluency of model generations. In 3 out of 4 datasets, EDMem-dynamic achieves the best results on knowledge-based metrics. This indicates that integrating entity linking with text generation is beneficial for generating informative sentences with rich entity knowledge. Interestingly, annotators even prefer EDMem's generations over human answers on ELI5. One possible reason is that human answers are usually longer than model-generated ones, so not all clauses are closely related to the question. Also, the quality of some Reddit responses (*i.e.*, the source of ELI5 data) may not be reliable.

Claim: Chicago Symphony Orchestra started in Indiana. This is false because _____
Ground truth: Chicago is in Illinois so it did not start in Indiana.
BART: It was not started here.
EDMem-free: The $[E_s]$ Chicago Symphony Orchestra $[E_e]$ started in $[E_s]$ Utah $[E_e]$. **Attended entities**: "Illinois", "Chicago", "Cook County, Illinois", "Wisconsin", "United States"
EDMem-dynamic: $[E_s]$ Chicago Symphony Orchestra $[E_e]$ was founded in $[E_s]$ Illinois $[E_e]$. **Attended entities**: "Illinois", "Cook County, Illinois", "Chicago", "Wisconsin", "United States"

Table 6: Case study from the CREAK dataset. We list the top-5 entities that EDMem attends to when it generates the underlined entity.

4.2.4 Case Study

In Table 6, we show an example from CREAK with generations of different models. Without knowledge augmentation, BART fails to generate an informative explanation on why the starting place of the orchestra is not Indiana. Although EDMem-free steps closer to the correct explanation, it falsely predicts that the orchestra started in Utah. However, "Utah" does not exist in the top-5 linked entities during memory access. After we constrain the generation space of EDMem-dynamic to the top-5 predicted entities, "Utah" is no longer valid to generate, and the model finds "Illinois" as the correct location. Examples from other datasets can be found in Appendix G.

4.2.5 Impact of Entity Richness on Generation Improvement

In Table 7, we show detailed ROUGE-L scores according to the number of entity mentions in the ground-truth. Examples with larger numbers of mentions require more entity knowledge to generate. We list scores for CREAK dataset where the outputs are short factual claims, and ELI5 dataset where the outputs are long and diverse answers. In both datasets, the improvement of EDMem over BART occurs on entity-rich generations. For examples which do not need entity knowledge to solve (contain 0 mentions), there is not much difference between the performance of two models. This further demonstrates the effectiveness of incorporating knowledge from the entity memory on entity-intensive generation tasks.

#Mentions	CREAK		ELI5	
	BART	EDMem	BART	EDMem
0	8.89	8.86	13.75	14.28
1	26.28	27.47	15.96	16.38
2	35.33	37.10	16.41	17.50
3	34.63	36.55	17.91	19.76
4	34.05	34.78	17.14	19.42
5+	30.84	31.56	18.84	20.23

Table 7: ROUGE-L scores based on different number of mentions in the ground-truth reference.

5 Conclusions

In this work, we proposed EDMem, an encoder-decoder framework with entity memory. The entity memory was pre-trained on Wikipedia to provide entity knowledge for the encoder-decoder model. EDMem also performed entity linking with the memory to assist entity generation in downstream tasks. As a unified framework, EDMem outperformed previous closed-book models on various entity-intensive QA and generation tasks, and still retained the efficiency advantage over open-book models. Further analysis proved that the proposed EDMem was enabled for entity linking with the entity memory and for generation with the encoder-decoder framework.

6 Limitations

First, if applying EDMem to other datasets, its performance may correlate to the density of entity mentions in data examples. EDMem may not be able to acquire sufficient entity knowledge from the memory if there are few mentions in the specific task. Another limitation of our work is that the pre-trained entity memory may not be generalized to special domains, *e.g.*, biomedical text. A lot of specific terminology is not included in our pre-trained entity memory, which may require additional training on domain-specific corpora.

Acknowledgement

This work was supported in part by NSF IIS-1849816, IIS-2119531, IIS-2137396, IIS-2142827, CCF-1901059, and ONR N00014-22-1-2507. We would like to thank Yuwei Fang (Microsoft), Jinfeng Lin (Meta), Mingxuan Ju (University of Notre Dame), and Qian Liu (Nanyang Technology University) for their valuable suggestions to this work.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013*.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019*.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *9th International Conference on Learning Representations, ICLR 2021*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017*.

Wenhu Chen, Pat Verga, Michiel de Jong, John Wieting, and William Cohen. 2022. Augmenting pre-trained language models with qa-memory for open-domain question answering. *ArXiv preprint 2204.04581*.

Michiel de Jong, Yury Zemlyanskiy, Nicholas FitzGerald, Fei Sha, and William Cohen. 2022. Mention memory: incorporating textual knowledge into transformers through entity mention attention. In *International Conference on Learning Representations, ICLR 2022*.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of wikipedia: Knowledge-powered conversational agents. In *7th International Conference on Learning Representations, ICLR 2019*.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: long form question answering. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019*.

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

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

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021*.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017*.

Minki Kang, Jinheon Baek, and Sung Ju Hwang. 2022. KALA: knowledge-augmented language model adaptation. *ArXiv preprint 2204.10555*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*.

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

Jinhyuk Lee, Mujeen Sung, Jaewoo Kang, and Danqi Chen. 2021. Learning dense representations of phrases at scale. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021*.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*.

Patrick S. H. Lewis, Pontus Stenetorp, and Sebastian Riedel. 2021. Question and answer test-train overlap in open-domain question answering datasets. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021*.

Belinda Z. Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih. 2020. Efficient one-pass end-to-end entity linking for questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

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*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019*.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory F. Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed precision training. In *6th International Conference on Learning Representations, ICLR 2018*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016),*.

Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. 2020. Unik-qa: Unified representations of structured and unstructured knowledge for open-domain question answering. *ArXiv preprint 2012.14610*.

Yasumasa Onoe, Michael J. Q. Zhang, Eunsol Choi, and Greg Durrett. 2021. CREAK: A dataset for commonsense reasoning over entity knowledge. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021*.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick S. H. Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. Language models are unsupervised multitask learners. *Technical Report, OpenAI*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*

Michael Ringgaard, Rahul Gupta, and Fernando C. N. 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? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*.

Devendra Singh Sachan, Siva Reddy, William L. Hamilton, Chris Dyer, and Dani Yogatama. 2021. End-to-end training of multi-document reader and retriever for open-domain question answering. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*.

Haitian Sun, Patrick Verga, Bhuwan Dhingra, Ruslan Salakhutdinov, and William W. Cohen. 2021. Reasoning over virtual knowledge bases with open predicate relations. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, NeurIPS 2017*.

Pat Verga, Haitian Sun, Livio Baldini Soares, and William W. Cohen. 2021. Adaptable and interpretable neural memoryover symbolic knowledge. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021*.

Cunxiang Wang, Pai Liu, and Yue Zhang. 2021. Can generative pre-trained language models serve as knowledge bases for closed-book qa? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021*.

Qinyuan Ye, Belinda Z. Li, Sinong Wang, Benjamin Bolte, Hao Ma, Wen-tau Yih, Xiang Ren, and Madian Khabsa. 2020. Studying strategically: Learning to mask for closed-book QA. *ArXiv preprint 2012.15856*.

Donghan Yu, Chenguang Zhu, Yuwei Fang, Wenhao Yu, Shuohang Wang, Yichong Xu, Xiang Ren, Yiming Yang, and Michael Zeng. 2022a. Kg-fid: Infusing knowledge graph in fusion-in-decoder for open-domain question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, ACL 2022*.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2022b. Generate rather than retrieve: Large language models are strong context generators. *ArXiv preprint arXiv:2209.10063*.

Wenhao Yu, Chenguang Zhu, Zaitang Li, Zhiting Hu, Qingyun Wang, Heng Ji, and Meng Jiang. 2022c. A survey of knowledge-enhanced text generation. *ACM Computing Surveys (CSUR)*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020a. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020*.

Zhihan Zhang, Xiubo Geng, Tao Qin, Yunfang Wu, and Daxin Jiang. 2021. Knowledge-aware procedural text understanding with multi-stage training. In *WWW '21: The Web Conference 2021*.

Zhihan Zhang, Zhiyi Yin, Shuhuai Ren, Xinhang Li, and Shicheng Li. 2020b. DCA: diversified co-attention towards informative live video commenting. In *Natural Language Processing and Chinese Computing - 9th CCF International Conference, NLPCC 2020*.

Zhihan Zhang, Wenhao Yu, Mengxia Yu, Zhichun Guo, and Meng Jiang. 2022. A survey of multi-task learning in natural language processing: Regarding task relatedness and training methods. *ArXiv preprint 2204.03508*.

A Pre-Training

A.1 Pre-Training Data

We pre-train our model on the Wikipedia corpus of over 5 million documents. All documents are split into 128-token passages. The last passage is round up to 128 tokens by appending tokens from the beginning of the same document, so there are no cross-document passages. In addition, we set a 10-token sliding window between passages to avoid an entity being split into two adjacent chunks. Such a setting yields a total of 39 million passages, of which we hold out 0.5% of them as the validation set during the pre-training process. For supervision signals on the entity memory, we leverage Wikipedia hyperlinks as gold annotations. Each hyperlink provides the boundaries of a mention, and also the corresponding entity[5] that the mention is linked to.

However, the average density of Wikipedia hyperlinks is only one in 21 words, which means 6 mentions per passage. This is because in a specific page, (1) only the first mention of an entity is linked and (2) the title entity is not linked since a hyperlink always redirects to a different page. To provide more supervision signals for entity embedding learning, we label the missing mentions using heuristic rules. We use the alias table $\mathcal{T}$ to label all mentions in a Wikipedia page if they match either (1) a linked entity in the same page, or (2) the title entity of this page. After such heuristic labeling, the hyperlink density increases to one in 11 words, with a passage having 12 mentions on average. We manually checked 50 passages and found the precision of such heuristic labeling to be 92%, a pretty acceptable rate.

A.2 Pre-Training Settings

We pre-train our model on the Wikipedia corpus containing 39 million passages for 1 million steps using a batch size of 2048. AdamW (Loshchilov and Hutter, 2019) optimizer is used with maximal learning rate 1×10^{-4} and the weight decay coefficient is 0.01. The learning rate is scheduled to be warmed up for 10% of the training steps and then linearly decay. The mask rate for random token masking is $P_{rtm} = 0.3$, and the mask rate for salient span masking is $P_{ssm} = 0.5$ (ablations in Appendix E.1). The maximum length of input sequence is set to 128. The coefficient of the entity linking objective is set to $\lambda_{EL} = 1.0$ and the dropout rate is 0.1. The whole model is trained from scratch. We tried to initialize the encoder-decoder model with BART (Lewis et al., 2020a) and derive entity embeddings from BART embeddings, but the model showed up to be unstable in further training. We use the mixed precision floating point arithmetic (Micikevicius et al., 2018) to speed-up training. The full setting of EDMem takes about two weeks to train on 16×A100 GPUs.

[5] In Wikipedia, each page corresponds to a unique entity which has the same name as the page title.

B Fine-Tuning

Different from pre-training on Wikipedia, in open-domain QA and generation tasks, there are no gold annotations of mention boundaries in the input and output. Therefore, we annotate mention boundaries as well as the linked entities using a state-of-the-art neural entity linker ELQ (Li et al., 2020). For generation datasets, we pass the source sequence and the target sequence into the ELQ model respectively, and obtain their mention annotations. For open-domain QA datasets, since the answers are usually short, we concatenate the question and the answer as input to the ELQ model. During fine-tuning, we tune the hyperparameters within the following ranges: learning rate $\in$ {5e-6, 1e-5, 2e-5, 3e-5}, $\lambda_{EL} \in \{0.5, 1.0, 2.0\}$, dropout rate $\in \{0.1, 0.2, 0.3\}$, beam size $\in \{1, 3, 5\}$, #candidate entities (in static/dynamic entity linking) $\in \{1, 3, 5\}$. They are tuned based on on the main evaluation metric of the specific task (open-domain QA: EM; WoW: F1; other generation datasets: ROUGE-L) on the dev set. Batch size is fixed to 256 unless it exceeds GPU memory or the dataset is too small (*e.g.*, CREAK and WQ). Early stopping is used with 20 waiting steps on the dev set. Fine-tuning EDMem usually costs a few hours (*e.g.*, ~3 hours on TQA) on 8×V100 GPUs.

C Entity Memory Settings

We collect the 1-million most frequent entities in Wikipedia documents as our entity vocabulary $\mathcal{E}$. The frequency of an entity is calculated based on how many hyperlinks are linked to the Wikipedia page of that entity. The dimension of entity embeddings learned in the memory is set to 256. The model attends to all 1 million entities during training. During inference, top-100 entities are selected according to the dot product similarity, and we only

Dataset	Train	Dev	Test
TQA (In-House)	78,785	8,837	11,313
TQA (Official)	87,622	11,313	Not Used
NQ	79,168	8,757	3,610
WQ	3,417	361	2,032

Table 8: Statistics of open-domain QA datasets.

Dataset	Train	Dev	Test
MSMARCO	138,352	15,373	12,467
CREAK	9,158	1,018	1,371
ELI5	75,220	8,361	1,384
WoW	54,330	3,054	2,944

Table 9: Statistics of generation datasets.

integrate the embeddings of these 100 entities when performing attention.

D Datasets

D.1 Open-Domain QA Datasets

Statistics of the open-domain QA datasets are listed in Table 8. In TQA, most previous works used the in-house split provided by (Lee et al., 2019), while we also test EDMem on the official dev set to compare with the scores reported by EaE.

D.2 Generation Datasets

Here we provide detailed descriptions of the generation datasets used in our experiments. Statistics of these datasets are listed in Table 9.

MSMARCO MSMARCO (Nguyen et al., 2016) is originally collected for the abstractive QA task. We use the NLGen split where answers are sentences carefully written by human workers. Since the official leaderboard has been closed, we test our model on the official dev set and hold out 10% training examples for validation.

CREAK CREAK (Onoe et al., 2021) is a recent dataset for claim verification and explanation. In our experiments, we target on the explanation subtask. The model is given a factual claim and a true/false judgment, and is expected to generate an explanation about why the claim is true or false. Since the official test set does not have explanations, we report results on its dev set and hold out 10% of the training set for validation.

ELI5 ELI5 (Fan et al., 2019) is a dataset for generating long-form responses for factual questions. To keep a reasonable density of entities, we filter a subset where the input question and the output response are both no longer than 75 tokens. We also remove those examples which have no entity mentions in the output. This result in a total of 85K data examples.

Wizard of Wikipedia (WoW) WoW (Dinan et al., 2019) is a dialogue dataset where entity knowledge is included in speakers' responses. We use the open-domain setting of this dataset provided by the KILT benckmark (Petroni et al., 2021), where no candidate knowledge pieces are given to make the response. We additionally remove the training examples where no knowledge piece is used to generate the response.

E Additional Experiments

E.1 Pre-Training Mask Rates

We test the performance of EDMem on different mask rates during pre-training, and list the results in Table 10. In pre-training, we adopt two masked language modeling objectives: random token masking (RTM) and salient span masking (SSM). When using smaller mask rates, there is more visible contextual information to the model. Therefore, when evaluating the model on the validation set during pre-training, smaller mask rates lead to lower language model perplexity and better entity linking performances. However, larger mask rates finally lead to better performances on the downstream task. With larger mask rates, more training signals are applied to the model and thus the model is more sufficiently trained. Specifically, in SSM, the model is encouraged to leverage the entity knowledge from the entity memory to predict the masked mention. Therefore, a larger SSM rate leads to more sufficient learning of the entity memory, where the contextual information of masked mentions is integrated into the corresponding entity embedding.

E.2 Pre-Training Steps

We fine-tune EDMem on TQA using pre-trained checkpoints of different number of training steps. As is shown in Figure 5, longer pre-training leads to better performance on the downstream task. Although there is no sign of overfitting, as the learning rate gradually decays and the model gradually converges, there is not much improvement of the model performance after 500K steps.

P_{rtm}	P_{ssm}	Pre-Train		TQA	
		PPL↓	ACC↑	EM↑	Entity↑
0.1	0.3	1.14	73.33	41.01	46.86
0.2	0.4	1.31	72.94	41.52	47.03
0.3	0.5	1.51	71.37	42.24	48.27

Table 10: Performance of EDMem with different pre-training mask rates. PPL: perplexity of the masked tokens on validation set (lower is better); ACC: entity linking accuracy of masked mention spans on validation set; EM: overall exact match scores; Entity: exact match scores on entity answers.

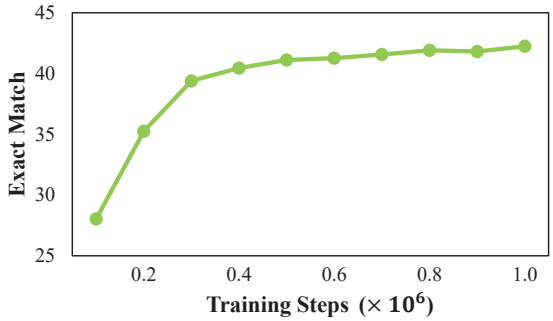

Figure 5: TQA performance of EDMem with different pre-training steps.

F Human Evaluation Details

Here we provide the actual questions that we asked Amazon Mturk annotators in human evaluation, along with their rubrics. For **fluency**, **relevance** and **correctness** metrics, annotators are asked to give scores on a 1-3 scale. For **informativeness** and **reasonability**, ranking evaluation is applied. This is because in ELI5 and WoW datasets, the human-written answer is not the only possible one to the context, so we do not compare model generations to the ground-truth. Instead, we let annotators evaluate the human written answer along with the model-generated ones. Since **informativeness** and **reasonability** are hard to set clear rubrics if no ground-truth is given, we adopt a ranking-based evaluation. The annotator is asked to rank all sequences (three model generations and the human-written answer) from #1 – #4, with lower rankings indicating better results.

- **Fluency**: How is the fluency of the machine-generated explanation? (Do not consider its correctness)
  - ◇ 3 – Fluent English
  - ◇ 2 – Readable with grammar errors or typos
  - ◇ 1 – Not fluent at all
- **Relevance**: Does the machine-generated explanation contain the same concepts as the human-written reference? (Synonyms are allowed, and do not consider its factual correcteness)
  - ◇ 3 – Contains the same concepts as the reference
  - ◇ 2 – Misses some concepts in the reference, or contains redundant concepts
  - ◇ 1 – Does not contain any concept in the reference
- **Correctness**: Does the machine-generated explanation express similar meanings with the human-written reference? (Paraphrases are allowed)
  - ◇ 3 – Expresses similar meanings with the reference
  - ◇ 2 – Expresses partial meanings of the reference
  - ◇ 1 – Expresses totally different meanings with the reference
- **Informativeness**: Rank these answers based on the amount of information they contain. #1: most informative, #4: least informative. Ties are allowed (e.g., 1/1/3/4 or 1/2/2/2). You do not need to consider whether the information is relevant to the question.
- **Reasonability**: Rank these answers based on whether they are reasonable answers to the question. #1: most reasonable, #4: least reasonable. Ties are allowed (e.g., 1/1/3/4 or 1/2/2/2).

G Case Study

We present an example on TQA with a non-entity answer in Table 11. We use our auto-encoder variant EncMem to represent memory-based auto-encoder models. We also provide additional examples on generation datasets in Tables 12 – 14.

Question: At the equator, in miles per hour, what speed of the ground beneath your feet, as a result of the Earth's rotation?
Ground truth: [18,000 mph, eighteen thousand speed, 18,000 speed]
BART: 8,000 **EncMem**: Speed **EDMem-free**: 18,000 mph **EDMem-static**: 18,000 speed

Table 11: Case study of an example with a non-entity answer from the TQA dataset. Entity mentions in the question are underlined. Since the auto-encoder model EncMem can only perform entity linking to provide an answer, it selects an entity "Speed" from the entity memory. In contrast, EDMem has the ability of generating non-entity answers thanks to the encoder-decoder framework, and successfully predicts the correct amount of speed.

Question: Where is Niagara?
Ground truth: Niagara is in New York, United States.
BART: Niagara is in Ontario, Canada.
EDMem-free: $[E_s]$ Niagara $[E_e]$ is in $[E_s]$ Niagara County, Pennsylvania $[E_e]$, $[E_s]$ United States $[E_e]$. **Attended entities**: "Niagara County, New York", "Regional Municipality of Niagara", "Niagara Falls, New York", "Niagara Falls", "Wayne County, New York"
EDMem-dynamic: $[E_s]$ Niagara Falls $[E_e]$ is in $[E_s]$ Niagara County, New York $[E_e]$, $[E_s]$ United States $[E_e]$. **Attended entities**: "Niagara County, New York", "Niagara Falls", "Regional Municipality of Niagara", "New York (state)", "Upstate New York"

Table 12: Case study from the MSMARCO dataset. We list the top-5 entities that EDMem attends to when it generates the underlined entity. EDMem-free correctly predicts the county name and country name of Niagara. However, the state name "Pennsylvania" is not expected. Although the model generates $[E_s]$ and $[E_e]$ tokens around "Niagara County, Pennsylvania", it is not a valid entity name which leads to false information. In contrast, EDMem-dynamic generates the correct answer by constraining the model to generate "Niagara County, New York", one of the top-5 selected entity names.

Question: The history of Canada
Human-Written Answer: Canada was initially settled by France and later Britain invaded. Later some of the Southern British colonies rebelled against the King but the Northern colonies chose not to. A hundred years after that Southern rebellion the Northern colonies were granted their independence peacefully.
BART: Canada is a country of about 5 million people. It's a very large country, and it's been a part of the world for a long time.
EDMem-free: $[E_s]$ Canada $[E_e]$ is a $[E_s]$ country $[E_e]$ in the $[E_s]$ North America $[E_e]$. It was a $[E_s]$ colony $[E_e]$ of $[E_s]$ English $[E_e]$ and $[E_s]$ French $[E_e]$. **Attended entities**: "Great Britain", "United Kingdom", "England", "British Empire", "British people"
EDMem-dynamic: $[E_s]$ Canada $[E_e]$ is a country in the $[E_s]$ Americas $[E_e]$. It was a part of the $[E_s]$ British Empire $[E_e]$, which was a $[E_s]$ nation $[E_e]$ of $[E_s]$ Europe $[E_e]$. **Attended entities**: "British Empire", "Commonwealth of Nations", "United Kingdom", "Canada", "United States"

Table 13: Case study from the ELI5 dataset. We list the top-5 entities that EDMem attends to when it generates the underlined entity. BART generates a fluent sentence about Canada. However, it barely contains any useful information. Canada's population is much larger than 5 million, and the whole generation is not related to the history of Canada. The generation of EDMem-free is more informative than BART, but when it is supposed to generate country names, it generates "English" and "French", which hurts its performance. EDMem-dynamic avoids this problem since "English" is not in the top attended entities. It is able to generate a more informative answer by correctly selecting entities from the memory.

Dialogue: – I was a cheerleader in high school. Were you ever a cheer leader? – Yeah! Remember back then, we do encourage our college basketball team to success. – What is your favorite cheer leading move? I loved the pyramid.
Human-Written Response: Yeah I loved the components of tumbling, dance, jumps, cheers, and stunting.
BART: I love the pyramid. It is a large, circular structure that is used to hold the flag.
EDMem-free: I like the $[E_s]$ hula hoop $[E_e]$. It is a $[E_s]$ dance $[E_e]$ performed by $[E_s]$ hula hoops $[E_e]$. **Attended entities**: "Haka", "Human pyramid", "Jumping", "Push-up", "Tumbling (gymnastics)"
EDMem-dynamic: I love the $[E_s]$ human pyramid $[E_e]$. $[E_s]$ Cheerleading $[E_e]$ is a form of $[E_s]$ performance art $[E_e]$ that combines $[E_s]$ gymnastics $[E_e]$ and $[E_s]$ dance $[E_e]$. **Attended entities**: "Human pyramid", "Jumping", "Haka", "Cheering", "Cheerleading"

Table 14: Case study from the WoW dataset. We list the top-5 entities that EDMem attends to when it generates the underlined entity. The second generated sentence of BART does not correlate with the context, and does not contain any correct information either. The generation of EDMem-free talks about the hula hoop, but it is not a typical cheerleading move. EDMem-dynamic generates a reasonable response by first selecting a correct cheerleading move "human pyramid", and then continues to generate an informative sentence by selecting relevant entities from the memory.