MEM2MEM: LEARNING TO SUMMARIZE LONG TEXTS WITH MEMORY-TO-MEMORY TRANSFER

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce the *Mem2Mem mechanism*, a conditional memory-to-memory mechanism that can be appended to general sequence-to-sequence frameworks, and demonstrate its effectiveness in improving long text neural abstractive summarization. Mem2Mem seamlessly transfers "memories" via readable/writable external memory modules that augment both the encoder and decoder. By enabling a memory transfer, Mem2Mem uses representations of highly salient input sentences and performs an implicit sentence extraction step. By allowing the decoder to read and write over encoded input memories, the models learn to store information about the input sequence while keeping track of what has been generated by the decoder. We evaluate Mem2Mem on abstractive text summarization and surpass the current state-of-the-art with less model capacity than competing models and with a full end-to-end training setup. To our knowledge, Mem2Mem is the first mechanism that can effectively use and update memory cells filled with different contextual information.

1 INTRODUCTION

Automatic summarization is the automated process of reducing the size of an input text while preserving its most relevant information content and its core semantics. Abstractive text summarization requires a source document to be understood, the most important parts to be prioritized and key concepts to be paraphrased into coherent sentences. Most recent approaches to solving abstractive summarization tasks have relied on the sequence-to-sequence (seq2seq) paradigm (Sutskever et al., 2014), with an encoder that generates vector representations of the input tokens and a decoder that generates an output text conditioned on the encoded representations. Advances in the conditional Natural Language Generation (NLG) of seq2seq architectures have typically focused on improving the decoder and have been particularly successful when applied to problems such as machine translation (Cho et al., 2014; Bahdanau et al., 2014), abstractive summarization (Rush et al., 2015; Nallapati et al., 2016; See et al., 2017) and semantic parsing (Dong & Lapata, 2016).

Several attention-based encoder-decoders were introduced to tackle varying text generation issues of standalone seq2seq models. Gu et al. (2016) proposed a copy mechanism to address difficulties in generating out-of-vocabulary words and named entities. See et al. (2017) used both a pointer mechanism (Vinyals et al., 2015) and a coverage mechanism (Tu et al., 2016) to address repetitive generation of word sequences in abstractive summarization. For very large input documents, however, such attention based techniques can still suffer from uninformative encoding. It is well studied that Recurrent Neural Networks (RNN) trained with stochastic gradient descent have difficulty learning long-term dependencies encoded in the input sequences due to vanishing gradients (Hochreiter, 1998; Pascanu et al., 2013). While attention mechanisms (Bahdanau et al., 2014; Luong et al., 2015) in the decoder provide more precise conditioning and facilitate gradient flow into the encoder, it is still unclear how trained encoders improve over word embeddings and how linguistic information of the input text is effectively represented (Wieting & Kiela, 2019; Park et al., 2019). Memory-Augmented Neural Networks (MANN) and Memory-Augmented Encoder-Decoder (MAED) have also been seen as potential solutions to long term dependency issues in RNNs and seq2seqs respectively (Wang et al., 2016; Yogatama et al., 2018; Ma & Príncipe, 2018; Le et al., 2018). Using differentiable read and write operations to an external module, MAED can represent non-local context of RNNs using its larger memory capacity. Such models are able to store temporally distant information of large input sequences. However, the role in memory for input encoding and sequence generation had not yet been fully understood or exploited.

Geared towards generating concise summaries and focusing summary generation on the most relevant information, alternative ideas introduced an intermediate sentence extraction step. Most recently, hybrid abstractive and extractive architectures (Chen & Bansal, 2018; Subramanian et al., 2019) were proposed and have proven to outperform competing methods on a number of summarization datasets. In such set-ups, a downstream attention based model first selects salient sentences and then rewrites them abstractively without relying on the encoded input representation. However, the two-step approach not only makes strong assumptions that all of the information necessary to generate a summary is contained within a subset of sentences, but it also relies on suboptimal labels (i.e. individual sentence level ROUGE scores) to identify high saliency sentences during the extraction step. Such targets for the extractive step do not necessarily ensure minimal content overlap and maximal ROUGE scores on the whole set of extracted sentences (Narayan et al., 2018). Furthermore, it is assumed that the performances of the extractive model is independent of the behavior of the abstractive model. In experiments by Chen & Bansal (2018); Subramanian et al. (2019) for instance, most if not all of the training cycles were performed separately for the abstractive and extractive models.

In this paper, we propose a memory based end-to-end approach trained from scratch that implicitly performs an extraction step without needing extraction labels. The architecture, that is called Mem2Mem, was inspired by the reconsolidation hypothesis, which states that the incorporation of new information can happen when memory is recalled in the brain (Sara, 2000; Coccoz et al., 2011). To our knowledge, Mem2Mem is the first MAED that creates a long term memory of the encoded input and that allows the decoder to modify its "memories" during the summary generation process. We empirically demonstrate the merits of this approach by setting a new state of the art on long text abstractive summarization results on the Pubmed dataset (Cohan et al., 2018). Our results show large improvements on all ROUGE scores, especially on ROUGE-L, the longest common subsequence and "a means of assessing fluency" (Narayan et al., 2018) over competing models. We further provide evidence that our new memory-based technique performs an implicit extraction step by choosing sentence representations to add to memory and to use during summary generation. Our contributions are three folds:

1. We introduce a novel mechanism, called Mem2Mem, to transfer memories created during the encoding process to the decoder, that is able to read and modify memory contents.
2. We show that Mem2Mem performs an implicit information retrieval step akin to extractive summarization, without needing extraction labels.
3. We achieve state-of-the art results with a fully differentiable model trained end-to-end.

2 Background

The methodology proposed in this paper can be attached to general seq2seq models that have either RNN or Convolutional encoder/decoders. Since we are interested in solving very long text summarization, we adopt a baseline seq2seq model based on the hierarchical recurrent encoder-decoder structure (HRED) from Nallapati et al. (2016) and Cohan et al. (2018).

The hierarchical recurrent encoder has two encoder GRUs: a sentence encoder and a document encoder. Given an input sentence of length N, the sentence encoder takes a sequence of token embeddings $\mathbf{x}$ and transforms it into a sequence of hidden states.

$$\mathbf{h}_1^{(w)}, \mathbf{h}_1^{(w)}, \ldots \mathbf{h}_N^{(w)} = \text{GRU}_{sen}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) \tag{1}$$

The last hidden state of the sentence encoder is used as a corresponding sentence embedding $\mathbf{s}$. Note that the sentence encoder is shared across every sentence in the input document. The sequence of sentence embeddings $\mathbf{s}$ are then processed by the document encoder.

$$\mathbf{h}_1^{(s)}, \mathbf{h}_1^{(s)}, \ldots \mathbf{h}_L^{(s)} = \text{GRU}_{doc}(\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_L) \tag{2}$$

where $\mathbf{s}_j$ is the j th sentence embedding, $\mathbf{h}_j^{(s)}$ is the associated document encoder hidden state and L is the total number of sentences in the document.

The decoder GRU generates a target summary one token at a time. At each decoding step t, the decoder creates a decoder hidden state $\mathbf{h}_t^{(d)}$. The decoder then obtains the context vector $\mathbf{c}_t$:

$$\mathbf{c}_t = \sum_{i=1}^{N_d} \gamma_{ti} \mathbf{h}_i^{(w)} \tag{3}$$

where γ_{ti} is the attention weight that represents the alignment between $\mathbf{h}_i^{(w)}$ and $\mathbf{h}_t^{(d)}$. γ_{ti} is computed by combining token level attention α and sentence level attention β.

$$\alpha_{ti} = \text{Attn}(\mathbf{h}_t^{(d)}, \mathbf{h}_i^{(w)}) \tag{4}$$

$$\beta_{tj} = \text{Attn}(\mathbf{h}_t^{(d)}, \mathbf{h}_j^{(s)}) \tag{5}$$

$$\gamma_{ti} = \frac{\beta_{tm(i)}\alpha_{ti}}{\sum_{l=1}^{N_d} \beta_{tm(l)}\alpha_{tl}} \tag{6}$$

$m(l)$ denotes the index of the sentence corresponding to the l th word, and N_d is the total number of tokens in the input document. Attn in equation (4) and (5) is defined as in Bahdanau et al. (2014):

$$\text{Attn}(\mathbf{q}, \mathbf{k}) = \text{softmax}(v^\top \tanh(\mathbf{W_q q} + \mathbf{W_k k})) \tag{7}$$

Finally, the pointer generator and decoder coverage method in See et al. (2017) are used for the baseline HRED. More details of the baseline architecture can be found in Appendix.

3 Conditional Memory-to-Memory Mechanism (Mem2Mem)

In this section, we describe the external memory modules that extend the baseline HRED to create Mem2Mem. Mem2Mem has three main features. (1) An encoder memory bank that compresses a large set of encoder hidden representations into a smaller subset of mutually exclusive representations. (2) Read/write operations that allow the decoder to update the encoder memory bank with new information during summary generation. (3) Word generation that is conditioned on the encoded sentence representations accessed from the dynamic memory. In essence, the whole process can be seen as *extraction* followed by *generation*. The architecture is depicted in Figure 1.

3.1 Neural Encoder Memory Bank

The aim of having an external memory on the encoder is to create a fixed size representation that reduces the set of L hidden representations from GRU$_{doc}$ to a subset of r representations. From a sequence of sentence-level document encoder hidden representations $\mathbf{h}_1^{(s)}, \mathbf{h}_1^{(s)}, \ldots \mathbf{h}_L^{(s)}$, we construct an intermediate 2-D matrix $\mathbf{H} \in \mathbb{R}^{L \times d}$, where d is the document encoder hidden size.

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_1^{(s)} \\ \mathbf{h}_2^{(s)} \\ \ldots \\ \mathbf{h}_L^{(s)} \end{bmatrix} \tag{8}$$

A smaller sized memory bank is generated by taking a linear combination of each row in $\mathbf{H}$. The weight vector for the linear combination $\mathbf{a}$ is computed with self-attention mechanism.

$$\mathbf{a} = \text{softmax}(w_{a1}^T \text{tanh}(W_{a2}\mathbf{H}^T)) \tag{9}$$

where $w_{a1} \in \mathbb{R}^{d_a}$, $W_{a2} \in \mathbb{R}^{d_a \times d}$, and d_a is the size of the hidden layer which is a hyperparameter. To capture various aspects of the input document, $\mathbf{a}$ is extended to a multi-headed attention matrix $\mathbf{A}$ with r heads.

$$\mathbf{A} = \text{softmax}(W_{a1} \text{tanh}(W_{a2} H^T)) \tag{10}$$

where $W_{a1} \in \mathbb{R}^{r \times d_a}$ and r is a hyperparameter. This results in r different weighted sums of $\mathbf{H}$, which gives us the final multi-headed encoder memory matrix $\mathbf{M^{(e)}} \in \mathbb{R}^{r \times d}$.

$$\mathbf{M^{(e)}} = \mathbf{AH} \tag{11}$$

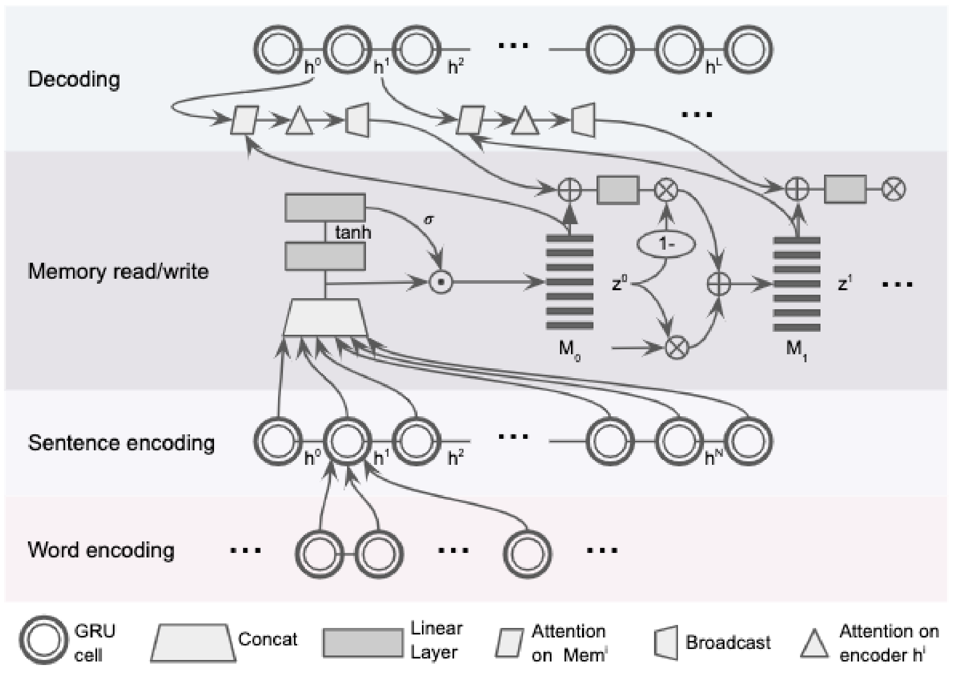


Figure 1: The proposed Mem2Mem architecture for abstractive summarization. Sentence-level representations from the document encoder hidden states are concatenated and reduced to a fix sized memory M_0. M_i is read using a RAM like mechanism. The memory readout vector is used to condition word-level attention. The decoder hidden states modulated by word-level attention is then used to update the memory state M_{i+1} via a gated write operation (z^i).

Lin et al. (2017) used a similar approach to create multiple types of sentence embeddings for one particular sentence with a ”structured self-attentive sentence embedding” matrix. Different from their work, we use the external memory block with r number of slots to store a smaller set (much less than L) sentence representations. We further integrate the memory module onto the baseline HRED architecture. Additionally, in our approach, the memory representation $\mathbf{M}$ gets updated dynamically by the decoder while the ”structured self-attentive sentence embedding” matrix remains static.

To ensure that attention weights of different heads are not redundant, a regularization term P is added to the cross entropy training objective to encourage diversity of memorized encoded states. Similar to Lin et al. (2017), we use the following regularization loss to constrain the attention weight matrix $\mathbf{A}$:

$$P = ||(\mathbf{A}\mathbf{A}^T - I)||_F{}^2 \tag{12}$$

where $||\bullet||_F$ is the Frobenius norm. The regularization loss P achieves two goals simultaneously: (1) It promotes diversity over the r representations stored in memory, thereby reducing the risk of storing redundant information. (2) It ”hardens” the attention probabilities by reducing variance across the categorical attention distribution, thereby assuring that each memory slot is associated to approximately one sentence representation. As a result, the encoder memory $\mathbf{M}^{(\mathrm{e})}$ essentially performs implicit extraction over the encoder hidden states. Note that no supervision exists on this extractive step. In Section 5, we will demonstrate that the encoder memory bank learns to choose amongst a large set of input sentences representations, using only the back-propagated error signals from the target summary generation objective as a guide.

3.2 Readable/Writable Decoder Memory

Most MAED architectures are not equipped with a Decoder memory but only query the Encoder Memory. Our MAED stores decoder hidden states with a write operation that is different than the one described in the previous section 3.1. As a first step, the encoder memory matrix $\mathbf{M}^{(\mathrm{e})}$ is transferred to the decoder and used as an initial state of the decoder memory $\mathbf{M^{(d)}}$. At every time step t, the decoder reads from the memory and generates a memory read vector $\mathbf{m}_t$. Specifically, the decoder takes the weighted sum over r memory slots via a RAM like attention mechanism:

$$\psi_{tk} = \text{Attn}(\mathbf{h}_t^{(d)}, \text{M}_k^{(d)}) \tag{13}$$

$$\mathbf{m}_t = \sum_{k=1}^{r} \psi_{tk} \text{M}_k^{(d)} \tag{14}$$

where $\text{M}_k^{(d)}$ is the vector representation of the k th slot of $M^{(d)}$. To condition the decoder attention mechanism on the memory read, $\mathbf{h}_t^{(d)}$ is combined with $\mathbf{m}_t$ and generate a memory augmented decoder hidden state $\mathbf{h}_t^{(m)}$.

$$\mathbf{h}_t^{(m)} = W_m[\mathbf{h}_t^{(d)}; \mathbf{m}_t] \tag{15}$$

$\mathbf{h}_t^{(d)}$ in the equation (4) and (5) of the baseline system is then replaced with $\mathbf{h}_t^{(m)}$. Similarly, the probability distribution of the next target word y_t is estimated using the memory augmented decoder hidden state $\mathbf{h}_t^{(m)}$ and the context vector $\mathbf{c}_t$, such that:

$$P(y_t|y_{1:t-1}) = \text{softmax}(V(W_v[\mathbf{h}_t^{(m)}, \mathbf{c}_t])) \tag{16}$$

As a consequence, the attention over the source text and the prediction of the target token are conditioned on the memory read $\mathbf{m}_t$, a weighted sum of the memorized sentence representations. There is a direct link between the contents of the memory and text generation. The training loss L is the average negative log likelihood of the target word y_t over the whole ground truth summary of length T. The semantics of the source sequence that is kept in the decoder memory can also be modified during the conditional text generation process. The decoder memory write operation outlined below removes and adds information using a gated mechanism to "forget" past memories and "update" each memory slot. Essentially, the memory write operation enables the memory to log the history of what has been attended and generated. The gating mechanism is conditioned on the the memory content $\mathbf{M^{(d)}}$, the sentence level context vector $\mathbf{c}_t^{(s)}$, and the decoder hidden state $\mathbf{h}_{t+1}^{(d)}$:

$$\mathbf{c}_t^{(s)} = \sum_{j=1} \beta_{tj} \mathbf{h}_j^{(s)} \tag{17}$$

$$z_k^t = \sigma(W_{z_1}\mathbf{h}_{t+1}^{(d)} + W_{z_2}\mathbf{c}_t^{(s)} + W_{z_3}\text{M}_k^{(d)}) \tag{18}$$

$$\mathbf{u}^t = \tanh\Big(W_{u_1}\mathbf{h}_{t+1}^{(d)} + W_{u_2}\mathbf{c}_t^{(s)} + W_{u_3}\text{M}_k^{(d)}\Big) \tag{19}$$

$$\text{M}_k^{(d)} := z_k^t \odot \text{M}_k^{(d)} + (1 - z_k^t) \odot \mathbf{u}^t \tag{20}$$

A similar memory write mechanism, called *scratchpad*, has been proposed in Benmalek et al. (2019). The write operation updates all encoder hidden states given the decoder hidden state and the context vector. In our work, we condition updates on memory-augmented context and content. Moreover, the *scratchpad* mechanism scales linearly with the input document length and output summary length. To be more precise, the complexity of *scratchpad* is $O(NT)$ where N and T are the length of the input text and the target summary respectively. Our Mem2Mem, however, is $O(rT)$ where r is the number of memory slots. Given that r is a fixed hyperparameter and typically $r \ll N$, our method is an ideal candidate for large output text generation conditioned on even larger input documents.

4 Related Work

Memory based models have been used in several NLG tasks. Sukhbaatar et al. (2015), using a continuous memory representation similar to Graves et al. (2014)'s Neural Turing Machines, show the importance of allowing multiple reads and writes to memory between inputs in language modeling experiments. Yogatama et al. (2018) further improved on such experiments by using an LSTM language model equipped with a multihop adaptive continuous stack memory. When experimenting with the multihop adaptive continuous stack memory for abstractive summarization in a HRED context[1], we have found two issues (1) only the top 2 slots of the encoder's stack memory were filled (2) the decoder memory stack rarely used or mixed memories transferred from the encoder.

Other MAED have been proposed for conditional NLG, such as machine translation (Kaiser et al., 2017), image captioning (Park et al., 2017), QA (Sukhbaatar et al., 2015; Kumar et al., 2015; Miller et al., 2016; Na et al., 2017; Han et al., 2019). In summarization, MAED have been used on extractive summarization (Singh et al., 2017) and on short text abstractive summarization (Kim et al., 2018). These approaches excel at tasks where it is necessary to store some parts of a sequential input in a representation that can later be precisely queried. We found however that most MAED innovations focused on improving long term memories of the encoded input or read/write operations. In this work, we suggest solutions in these areas: (1) a memory creation mechanism that condenses the encoded input akin to an extraction step; (2) a conditional read/write mechanism that allows the decoder to use/update encoder memories. Further, we propose another feature — Mem2Mem transfers memories from different context. The encoder memory stores input sentence-level information while the decoder updates are conditioned on output word-level information.

5 Results and Discussion

We evaluate Mem2Mem on PubMed (Cohan et al., 2018) which is a large scale summarization dataset comprised of biomedical literature. The average lengths of source documents and target summaries are 3016 and 203 respectively. In comparison, the widely used CNN/DailyMail summarization dataset (Hermann et al., 2015; See et al., 2017) is almost 4 times smaller with average lengths of source documents and target summaries at 781 and 56 respectively. Our pre-processing and training details are identical to Cohan et al. (2018). The readers are encouraged to refer to the appendix for training and evaluation details. For the quantitative evaluation, we use the ROUGE metric (Lin, 2004) and report full-length F-1 ROUGE scores.

Table 1 recaps the ROUGE scores of Mem2Mem along with a baseline model and previous works. We first note that the baseline HRED outperforms similar hierarchical recurrent encoder summarization model suggested in Cohan et al. (2018)[2]. The results also show that Mem2Mem surpasses the state of the art for abstractive summarization on the PubMed dataset. Mem2Mem even outperforms the hybrid extractive-abstractive *TLM-I+E* (Subramanian et al., 2019) on all scores except for ROUGE-1. We may say that Mem2Mem is much more "fluent" than *TLM-I+E* as evidence by the 13.2 percentage point gain in ROUGE-L scores. Interestingly, Mem2Mem achieves such performances over *TLM-I+E* while needing 6 times less parameters (14.7M vs. 93.3M). *TLM-I+E* also benefits from a language modeling step on its input text that can be considered a form of pre-processing. We train our model from scratch. Finally, we reiterate that Mem2Mem is trained completely end-to-end while TLM requires separate training for the extractive model and the conditional transformer language model.

5.1 Ablation Study

To assess the importance of different components in Mem2Mem, we conduct an ablation study on PubMed dataset results. Table 2 shows the affects of adding various Mem2Mem features on ROUGE scores . *Encoder Mem* is the baseline HRED augmented with the neural encoder memory bank described in Section 3.1. The result demonstrates that the memory context indeed enhances the quality of generated summaries. Furthermore, it can be observed that discouraging redundancies in the encoder memory cells via the regularization loss P leads to additional improvements on all ROUGE scores. Figure 2 shows the effect of regularization on the encoder memory selection of encoded sentence representations. We observe that each memory head attends to a single sentence of the

[1]We used a GRU with a stack memory on the sentence encoder and decoder. More details can be found in the appendix.

[2]The difference may arise from employing a different hierarchical structure: Ours used the word-sentence hierarchy while Cohan et al. (2018) used the word-section discourse. We also trained more epochs (15) than their work (10) to build a strong baseline model.

Model	Type	# of params.	ROUGE			
			1	2	3	L
SumBasic	Extractive	-	37.15	11.36	5.42	33.43
LexRank	Extractive	-	39.19	13.89	7.27	34.59
Ptr-Gen-Seq2Seq (Cohan et al., 2018)	Abstractive	-	35.86	10.22	7.60	29.69
Discourse-aware (Cohan et al., 2018)	Abstractive	14.3M	38.93	15.37	**9.97**	35.21
TLM-I (Subramanian et al., 2019)	Abstractive	71.2M	36.63	11.67	5.40	21.00
TLM-I+E (Subramanian et al., 2019)	Hybrid	93.3M	**41.43**	15.89	8.62	24.32
Baseline HRED	Abstractive	14.0M	40.02	15.82	9.30	36.28
Mem2Mem (our model)	Abstractive	14.7M	41.35	**16.09**	9.00	**37.53**

Table 1: Results on the PubMed dataset. TLM uses a GPT-like transformer language models (Radford et al., 2019) conditioned on introduction (I) and with extracted sentences (E). The highest ROUGE scores for abstractive methods are bold-faced. *Hybrid* refers to models that use two-step extractive-abstractive summarization.

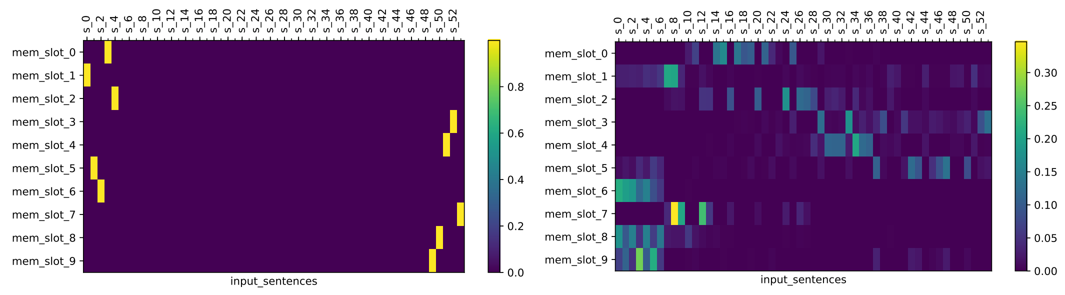
Model	ROUGE			
	1	2	3	L
Baseline HRED	40.02	15.82	9.30	36.28
+ Encoder Mem	40.39	16.00	9.39	36.58
+ Regularize Mem	40.72	16.06	**9.53**	36.90
+ Decoder Mem	41.09	16.00	9.02	37.34
+ Mem Transfer	**41.35**	**16.09**	9.00	**37.53**

Table 2: Model ablation study on the PubMed dataset. With *Encoder Mem*, the baseline HRED is augmented with a neural encoder memory bank as described in Section 3.1. We then regularize the encoder memory, *Regularize Mem*, using the equation (12). We also experiment with the *Decoder Mem* that performs read/write mechanism described in Section 3.2 without memory transfer. Finally, with *Mem Transfer*, we obtain the whole Mem2Mem setup.

input. This is due to the regularization term P that encourages sparse, non-redundant attention distributions over different memory heads or slots. Analysing multiple test samples, a trend emerges. The encoder memory typically attends to first and last few encoded sentences representations. This aligns with known statistic that existing summarization datasets tend to have key information at the front (i.e. the introduction) and last parts (i.e. the conclusion) of the document (Kim et al., 2018). *Decoder Mem* is a writable decoder memory but without memory transfer. In this case, the encoder memory is no longer used to initialize the decoder memory state at the start of the summary generation process. The decoder memory is simply initialized with zeros. With over one percentage point increases in both ROUGE-1 and ROUGE-L scores, the result shows that read/write mechanism on the decoder brings substantial improvements even without leveraging encoded sentence representations. This shows that the summary generation process largely benefits from accessing long term contextual information of the output text. The sparsity of the chosen content as seen Figure 2 (a) encourages us to think that presenting the decoder with a small subset of input sentence information helps summary generation in similar way to models that use a hybrid extractive-abstractive approach. Adding the memory transfer further improves ROUGE scores, which could imply that the soft extractive step of the encoder memory remains effective even with write operations on the decoder.

5.2 Implicit Extraction in the Encoder Memory

Our initial hypothesis was that the neural encoder memory bank would automatically pick a subset of the most salient sentences for a given task. To confirm this hypothesis, we analyze the quality of the sentences chosen by the encoder memory. As seen in equation (16), the Markovian probability of output tokens is conditioned on both the previous token and the decoder memory read m_t. In Mem2Mem, the memory content is initialised with a sparse set of encoded sentence representations (see Figure 2). As explained previously, the encoder memory attention $\mathbf{A}$ is the mechanism used to



(a) With regularization (b) Without regularization

Figure 2: An example of encoder memory attention $\mathbf{A}$. Rows denote different memory slots and columns indicate input sentence representations with indices. Note that the regularization loss removes the redundancy on different memory slots and helps each slot to focus on a single sentence.

Model	ROUGE 1	2	3	L
Lead-10	37.45	14.19	8.26	34.07
LexRank	39.19	13.89	7.27	34.59
Mem2Mem Encoder Memories	**40.11**	**15.56**	**8.99**	31.90
Gold Ext	47.76	20.36	11.52	39.19

Table 3: Rouge scores of unsupervised extracted input sentences with respect to the ground truth summaries. For *Mem2Mem Encoder Memories*, we used the sentences that had encoder memory attention $\mathbf{A}$ above 80%. *Gold Ext* is the gold extracted ROUGE scores of all sentences selected by a greedy selection from the input article that have the highest per-sentence ROUGE scores, as in Subramanian et al. (2019).

choose a subset of sentence representations. To better understand the link between encoder memory creation and the summary generation process, we compute the ROUGE scores of input sentence tokens that had above 80% probability encoder memory attention $\mathbf{A}$. That is, we calculated in Table 3 the ROUGE scores of input sentences whose representation was the most likely stored in the encoder memory bank.

Table 3 shows the ROUGE scores of different unsupervised extractive summarization methods on the PubMed dataset. The summaries generated with extracted sentences outperform existing unsupervised extractive summarization baselines on Pubmed. The result demonstrates that the encoder memory bank is able to prioritize amongst a large set of input sentences.

6 Conclusion

In this paper, we proposed a novel framework called Mem2Mem, which augments existing hierarchical seq2seq architectures and validated its efficacy on abstractive summarization task. From a long source document, the encoder memory learns to locate salient information without explicit labels for the extraction. The decoder memory, which is transferred from the encoder memory, enhances the quality of generated summaries by means of memory-augmented attention and hidden representations. The proposed Mem2Mem established new state of the art results on the challenging long-text summarization PubMed dataset. We also provided in-depth analyses of different features of Mem2Mem and show that the decoder memory along with memory transfer has the largest impact on performance.

The Mem2Mem's notion of implicit *extraction* followed by *generation* can be generalized to a wide range of seq2seq tasks such as question generation and answering. In future work, we would like to extend this approach and validate the strength of memory transfer on a variety of sequential learning tasks.

REFERENCES

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Ryan Benmalek, Madian Khabsa, Suma Desu, Claire Cardie, and Michele Banko. Keeping notes: Conditional natural language generation with a scratchpad encoder. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pp. 4157–4167, 2019.

Yen-Chun Chen and Mohit Bansal. Fast abstractive summarization with reinforce-selected sentence rewriting. *CoRR*, abs/1805.11080, 2018. URL `http://arxiv.org/abs/1805.11080`.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Vilma Coccoz, Hector J. Maldonado, and Alejandro Delorenzi. The enhancement of reconsolidation with a naturalistic mild stressor improves the expression of a declarative memory in humans. *Neuroscience*, 185:61–72, 2011.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. A discourse-aware attention model for abstractive summarization of long documents. *CoRR*, abs/1804.05685, 2018. URL `http://arxiv.org/abs/1804.05685`.

Li Dong and Mirella Lapata. Language to logical form with neural attention. *CoRR*, abs/1601.01280, 2016. URL `http://arxiv.org/abs/1601.01280`.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014. URL `http://arxiv.org/abs/1410.5401`.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. Incorporating copying mechanism in sequence-to-sequence learning. *CoRR*, abs/1603.06393, 2016. URL `http://arxiv.org/abs/1603.06393`.

Moonsu Han, Minki Kang, Hyunwoo Jung, and Sung Ju Hwang. Episodic memory reader: Learning what to remember for question answering from streaming data. *CoRR*, abs/1903.06164, 2019. URL `http://arxiv.org/abs/1903.06164`.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pp. 1693–1701, 2015.

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02): 107–116, 1998.

Lukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. Learning to remember rare events. *CoRR*, abs/1703.03129, 2017. URL `http://arxiv.org/abs/1703.03129`.

Byeongchang Kim, Hyunwoo Kim, and Gunhee Kim. Abstractive summarization of reddit posts with multi-level memory networks. *arXiv preprint arXiv:1811.00783*, 2018.

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

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. *CoRR*, abs/1506.07285, 2015. URL `http://arxiv.org/abs/1506.07285`.

Hung Le, Truyen Tran, Thin Nguyen, and Svetha Venkatesh. Variational memory encoder-decoder. *CoRR*, abs/1807.09950, 2018. URL `http://arxiv.org/abs/1807.09950`.

Chin-Yew Lin. Looking for a few good metrics: Automatic summarization evaluation-how many samples are enough? In *NTCIR*, 2004.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

Ying Ma and José C. Príncipe. A taxonomy for neural memory networks. *CoRR*, abs/1805.00327, 2018. URL `http://arxiv.org/abs/1805.00327`.

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. *CoRR*, abs/1606.03126, 2016. URL `http://arxiv.org/abs/1606.03126`.

Seil Na, Sangho Lee, Jisung Kim, and Gunhee Kim. A read-write memory network for movie story understanding. *CoRR*, abs/1709.09345, 2017. URL `http://arxiv.org/abs/1709.09345`.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*, 2016.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Ranking sentences for extractive summarization with reinforcement learning. *CoRR*, abs/1802.08636, 2018. URL `http://arxiv.org/abs/1802.08636`.

Cesc Chunseong Park, Byeongchang Kim, and Gunhee Kim. Attend to you: Personalized image captioning with context sequence memory networks. *CoRR*, abs/1704.06485, 2017. URL `http://arxiv.org/abs/1704.06485`.

Jaehong Park, Jonathan Pilault, and Christopher Pal. On the impressive performance of randomly weighted encoders in hierarchical encoder decoder models. In *Association for Computational Linguistics*, 2019.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. pp. 1310–1318, 2013.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8), 2019.

Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *CoRR*, abs/1509.00685, 2015. URL `http://arxiv.org/abs/1509.00685`.

Susan J Sara. Commentaryreconsolidation: Strengthening the shaky trace through retrieval. *Nature Reviews Neuroscience*, 1(3):212, 2000.

Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. *CoRR*, abs/1704.04368, 2017. URL `http://arxiv.org/abs/1704.04368`.

Abhishek Kumar Singh, Manish Gupta, and Vasudeva Varma. Hybrid memnet for extractive summarization. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 2303–2306. ACM, 2017.

Sandeep Subramanian, Raymond Li, Jonathan Pilault, and Christopher Pal. On extractive and abstractive neural document summarization with transformer language models. *arXiv preprint arXiv:1909.03186*, 2019.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. Weakly supervised memory networks. *CoRR*, abs/1503.08895, 2015. URL `http://arxiv.org/abs/1503.08895`.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. pp. 3104–3112, 2014.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Coverage-based neural machine translation. *CoRR*, abs/1601.04811, 2016. URL `http://arxiv.org/abs/1601.04811`.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pp. 2692–2700, 2015.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. Memory-enhanced decoder for neural machine translation. *CoRR*, abs/1606.02003, 2016. URL `http://arxiv.org/abs/1606.02003`.

John Wieting and Douwe Kiela. No training required: Exploring random encoders for sentence classification. *CoRR*, abs/1901.10444, 2019. URL `http://arxiv.org/abs/1901.10444`.

Dani Yogatama, Yishu Miao, Gabor Melis, Wang Ling, Adhiguna Kuncoro, Chris Dyer, and Phil Blunsom. Memory architectures in recurrent neural network language models. 2018.

A Appendix

A.1 Baseline Architecture

In addition to the hierarchical recurrent encoder-decoder (HRED) architecture described in Section 2, following features are used for both baseline and Mem2Mem architecture.

A.1.1 Pointer Generator network

In order to handle out-of-vocabulary (OOV) token predictions, pointer generator in See et al. (2017) is used to copy words directly from the input document. At each step t, the decoder decides whether to predict the next target word from the generation the pointer generator computes z_t which denotes the probability of choosing P_g for sampling the next target word.

$$z_t = \sigma(w_c^T \mathbf{c}_t + w_d^T \mathbf{h}_t^{(d)} + w_x^T \mathbf{x}_t^{'}) \tag{21}$$

The probability z_t is used as a variable for soft switch between generating a word from the vocabulary (P_g) or directly copying from the source document (P_c). The probability of copying a word w from the source text is calculated based on the attention weights γ.

$$P_c(y_t = w|y_{1:t-1}) = \sum_{i:x_i=w} \gamma_{ti} \tag{22}$$

Note that $P_c(y_t = w|y_{1:t-1}) = 0$ if w does not exist in the source document. Likewise, $P_g(y_t = w|y_{1:t-1}) = 0$ if w is an out of vocabulary word. Combining two probability distributions, the final probability of the next word y_t being w is as follows.

$$P(y_t = w|y_{1:t-1}) = z_t P_g(y_t = w|y_{1:t-1}) + (1 - z_t) P_c(y_t = w|y_{1:t-1}) \tag{23}$$

A.1.2 Decoder coverage

It is well known that sequence-to-sequence models tend to suffer from repeated phrases when generating long target sequences. See et al. (2017) tackled this issue by keeping track of the attention coverage. More concretely, the coverage vector $\mathbf{cov}_t$ at decoding step t is computed by taking the summation of the token-level attention weights α until the last step $t-1$.

$$\mathbf{cov}_t = \sum_{t'=0}^{t-1} \alpha_{t'} \tag{24}$$

To inform the decoder of the history of attention weights, the coverage vector is fed into the token-level attention mechanism, which modifies the equation (4) to the following equation.

$$\alpha_{ti} = \text{softmax}(v^\top \tanh\Big(W_e \mathbf{h}_k^{(e)} + W_d \mathbf{h}_t^{(d)} + w_c\, \mathbf{cov}_t^T\Big)) \tag{25}$$

A.2 Training Details

For pre-processing, we constrain the maximum number of sections to 4 and the maximum number of tokens for each section to 500. The length of the target summary is limited to 200 tokens.

Single-layer bidirectional GRUs (Cho et al., 2014) are used for sentence and document encoders. The decoder is also a single layer GRU. All GRUs have hidden size of 256. The dimensionality of token embeddings is 128 and embeddings are trained from scratch. The vocabulary size is limited to 50,000. Batch size is 16 and Adam (Kingma & Ba, 2014) with learning rate $2e^{-4}$ is used for training. Maximum gradient norm is set to 2. We train all models for 15 epochs. At the test time, beam search with the beam size 4 is used for decoding.

For Mem2Mem hyperparameters, the number of heads for the encoder memory module is 10 and the self-attention hidden size is 64. The weight for the regularization P is 0.01.