MemSum: Extractive Summarization of Long Documents using Multi-Step Episodic Markov Decision Processes

ETH zürich

Center for Law & Economics Working Paper Series

Number 12/2021

MemSum: Extractive Summarization of Long Documents using Multi-Step Episodic Markov Decision Processes

Nianlong Gu
Elliott Ash
Richard H.R. Hahnloser

August 2021

MemSum: Extractive Summarization of Long Documents using Multi-step Episodic Markov Decision Processes

Nianlong Gu
Institute of Neuroinformatics,
University of Zurich and
ETH Zurich
niangu@ethz.ch

Elliott Ash
Department of Humanities,
Social and Political Sciences,
ETH Zurich
ashe@ethz.ch

Richard H.R. Hahnloser
Institute of Neuroinformatics,
University of Zurich and
ETH Zurich
rich@ini.ethz.ch

Abstract

We introduce MemSum (Multi-step Episodic Markov decision process extractive SUMmarizer), a reinforcement-learning-based extractive summarizer enriched at any given time step with information on the current extraction history. Similar to previous models in this vein, MemSum iteratively selects sentences into the summary. Our innovation is in considering a broader information set when summarizing that would intuitively also be used by humans in this task: 1) the text content of the sentence, 2) the global text context of the rest of the document, and 3) the extraction history consisting of the set of sentences that have already been extracted. With a lightweight architecture, MemSum nonetheless obtains state-of-the-art test-set performance (ROUGE score) on long document datasets (PubMed, arXiv, and GovReport). Supporting analysis demonstrates that the added awareness of extraction history gives MemSum robustness against redundancy in the source document.

1 Introduction

Automatic text summarization is the task of automatically summarizing a long document into a relatively short text while preserving most of the information [1]. Text summarization methods can be categorized into abstractive and extractive summarization [2, 3]. Extractive summaries directly select sentences from the original text as the summary [4] and tend to be both grammatically and semantically more reliable [5] than abstractive summaries [6, 7, 8].

The extractive summarization task can be defined as follows: Given a document d consisting of an ordered list of N sentences $\{s_i | i = 1, \ldots, N\}$, the task is to pick up $M (M \ll N)$ sentences as the summary of the document.

The standard evaluation metric for summarization is the ROUGE score [9] between the extracted sentences and the gold summary. However, the ROUGE score is not differentiable and therefore cannot be directly used as the training loss [10, 11, 12]. One workaround is to view the extraction task as a sentence labelling problem, where a neural-network-based model is trained to predict the true label of each sentence with a cross-entropy loss [13], and the true label of each sentence is obtained by assigning label 1 to the set of sentences that jointly maximize the ROUGE score with respect to the gold summary, and assigning label 0 to the rest of the sentences.

Alternatively, to overcome the non-differentiability of the ROUGE computation, reinforcement-learning (RL) based approaches have been proposed that directly optimize the ROUGE metric [10, 14, 5, 12]. In policy gradient RL such as REINFORCE [15], a policy network is trained to extract from a document a subset of sentences that yield a summary with high ROUGE score. Current RL-based approaches usually model the extraction process as a one-step episodic Markov Decision Process (MDP) [14, 16], in which the extraction of multiple sentences is treated as a single action. The policy network produces an affinity score for each sentence based on the initial state in which no sentences had been been extracted yet. From these sentence scores the action probability is computed in either an instantaneous [10, 12] or a sequential [14, 5] manner, and the sentence scores usually remain unchanged during the extraction process. Therefore, the final learned policies do not depend on the extraction history, i.e., the set of unordered sentences that at any given time have been selected into the candidate summary.

Extractive summarizers that are not aware of the extraction history have two potential drawbacks. First, unlike the way humans extract documents, such extractive summarizers cannot stop extraction in time without knowing which sentences have been extracted into the summary. In recent one-step MDP-based RL approaches, a common strategy is to stop extraction until the preconceived maximum number of extracted sentences is reached, which may lead to lengthy summaries. Second, due to the lack of knowledge of the extraction history, the extraction model is susceptible to redundancy in the summaries, because sentences with high scores tend to be repeatedly selected for summarization, regardless of whether similar sentences have been selected before. Both these drawbacks can lead to performance decreases in ROUGE F1 scores.

In this paper, we address these drawbacks by making the extraction process fully aware of the extraction history. We model extractive summarization as a multi-step episodic MDP, where a summary is generated in an espisode of observing states and performing actions, which are to either select a sentence into the summary or to stop the extraction; the reward in this process is received at the end of the episode [16].

For each sentence in the document and at each time step in the process, we define a *sentence state* composed of three sub-states: 1) the content of the sentence, 2) the context of the sentence within the document, and 3) information on the extraction history, including the previously selected set of unordered sentences versus the remaining sentences. At each time step, the policy network (agent) takes the current sentence states as the input and transforms them into scores used to select an action of either stopping the extraction process or selecting one of the remaining sentences into the candidate summary.

Unlike one-step episodic MDP based models [10, 14, 5] that encode the state information only once at the beginning of the episode, in our multi-step policy the agent updates at each time step the extraction history before selecting an action. Such a step-wise state updating strategy enables the agent to consider the content of the partial summary when selecting a sentence.

To efficiently encode local and global sentence states, we design an extraction agent based on LSTM networks [17]; to encode the extraction history and to select actions, we use a reduced number of attention layers [18] and relatively low dimensionality. These choices enable our model to be easily trainable and to summarize long documents such as scientific papers [19, 20] or government reports [20].

The contributions of our work are as follows:

- We propose to treat extractive summarization as a multi-step episodic MDP that is aware of the extraction history.
- We show that the awareness of the extraction history allows our model to extract more compact summaries and behave more robustly to redundancies in documents than models without history awareness.
- Our model outperforms both extractive and abstractive summarization models on PubMed, arXiv [19] and GovReport [20] datasets.
- We provide an open source package for replicating our results, as well as usable extractive summarizers trained on each of the three datasets.[1]

2 Related Work

Current extractive summarization algorithms [10, 14, 21, 22, 23] usually contain two steps: 1) In the sentence scoring step, neural networks are adopted to compute an affinity score for each sentence. 2) In the sentence selection step, sentences are selected into a summary based on their scores.

In HIBERT [22], BERTSUMEXT [23], and the long-document extractive summarizer [21], sentence selection is treated as a sequence labeling process by predicting a binary label (0 or 1) of each sentence using the affinity score, and selecting sentences with label 1 as the summary; In the RL-based method REFRESH [10], sentences are ranked and the top K sentences with highest scores are selected as the summary; In BanditSum [14], the sentence selection is modeled as a process of sequential sampling without replacement, where the normalized scores of the remaining sentences are used as the sampling likelihood. In these approaches, sentence scores are generally not updated based on the current partial summary of previously selected sentences, indicating a lack of knowledge about extraction history.

One way to make the extraction process history-aware is to select actions by considering a combination of multiple sentences as the candidate summary. For example, a two-stage extractive summarizer is provided in Zhong et al. [24]: First, a BERT-based summarizer [23] is used to obtain a pruned subset of salient sentences. Second, all combinations of sentences from that pruned subset are ranked by a fine-tuned Siamese-BERT [25]. These BERT-based extractors achieve state-of-the-art performance on datasets such as CNN/DM [26], but the quadratic computational and memory complexities [20] of such models limit their scalability for extractive summarization of long documents with thousands of tokens.

Although large pre-trained Transformers with efficient attention mechanism [20] have been proposed to abstractively summarize long documents, we argue that extractive summarization tends to be more faithful. Furthermore, because MemSum achieves state-of-the-art performance on various long document summarization tasks, MDP approaches will be promising design choices for further research.

3 Model

This section outlines the multi-step episodic MDP policy for extractive summarization.

3.1 Policy Gradient Methods

Policy gradient methods aim to learn a parameterized policy of selecting actions that maximize a certain objective function $J(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ represents the parameters of the policy $\pi_{\boldsymbol{\theta}}$. In an episodic task with a terminal state (i.e. *end of summary*), the objective function J is the expected return for that policy, $J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[R_0]$, where the return $R_t = \sum_{k=t+1}^{T} r_k$ is the cumulative reward from time $t+1$ until the end of the episode. In applications of RL to extractive summarization, the instantaneous reward r_t is zero except at the end of the episode when the final reward r is computed in a single step according to Equation (1), so $R_t \equiv R_0 = r$. The reward r is usually expressed as [14]:

$$r = \frac{1}{3}(\text{ROUGE-1}_f + \text{ROUGE-2}_f + \text{ROUGE-L}_f) \tag{1}$$

According to the REINFORCE algorithm [15], the policy gradient is defined as:

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}_{\pi}[R_t \nabla \log \pi(A_t|S_t, \boldsymbol{\theta})], \tag{2}$$

where $\pi(A_t|S_t, \boldsymbol{\theta})$ denotes the likelihood that at time step t the policy $\pi_{\boldsymbol{\theta}}$ selects action A_t given the state S_t. The parameter update rule is [16] (α is the learning rate):

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \alpha R_t \nabla \log \pi(A_t|S_t, \boldsymbol{\theta}_t), \tag{3}$$

3.2 Multi-step Episodic MDP Policy

We regard the extraction of the summary as an episode and the selection of a single sentence into the candidate summary as an action. The objective, then, is to learn a policy that maximizes the expected return of an episode. This approach is different from one-step episodic MDP policies [10, 14, 5] that extract the whole summary of multiple sentences from a single action.

Formally, we assume that at each time step t, corresponding to the extracted sentence number t, the action A_t is either to stop the extraction or to select a sentence s_{a_t} from the remaining sentences. The agent's associated policy is:

$$\begin{aligned} \pi(A_t|S_t, \boldsymbol{\theta}_t) &= p(\text{stop}|S_t, \boldsymbol{\theta}_t) p(a_t|\text{stop}, S_t, \boldsymbol{\theta}_t) \\ p(a_t|\text{stop}, S_t, \boldsymbol{\theta}_t) &= \begin{cases} \frac{u_{a_t}(S_t, \boldsymbol{\theta}_t)}{\sum_{j \in I_t} u_j(S_t, \boldsymbol{\theta}_t)} & \text{if stop} = \text{false} \\ \frac{1}{|I_t|} & \text{if stop} = \text{true}, \end{cases} \end{aligned} \tag{4}$$

[1]Our code and datasets used in this paper will be provided online.

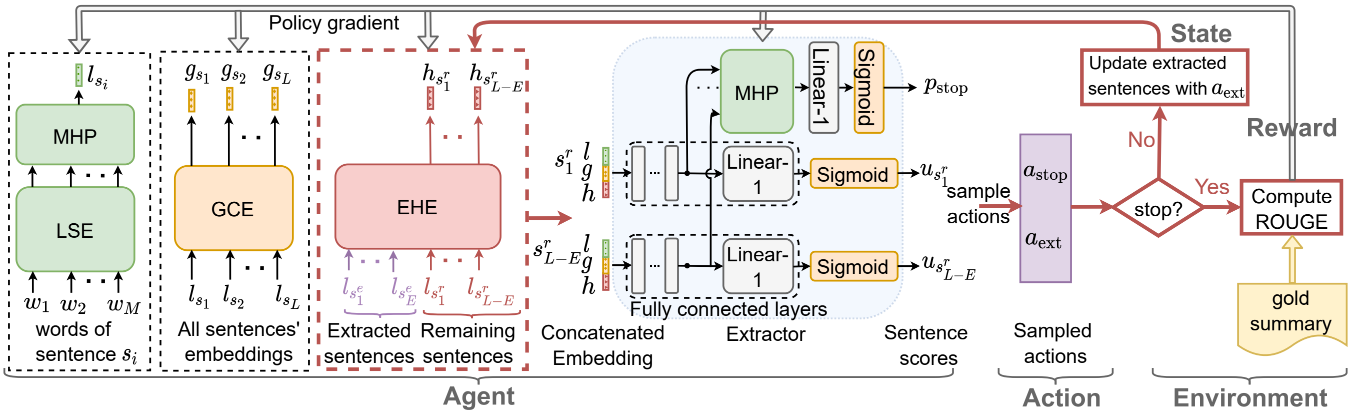

Figure 1: The architecture of our MemSum extractive summarizer with a multi-step episodic MDP policy. Because the extraction-history embeddings h are updated at each time step t, the scores u of remaining sentences and the stopping probability p_{stop} are also updated.

where I_t denotes the set of remaining sentence indices at time step t. If the agent determines not to stop, it first computes a score u_j for each of the remaining sentences and samples a sentence s_{a_t} based on the probabilistic distribution of normalized scores. When the agent stops the extraction, no sentence is selected and the conditional likelihood $p(a_t|\text{stop=false}, S_t, \boldsymbol{\theta}_t)$ is set to $\frac{1}{|I_t|}$ (where $|I_t|$ represents the number of remaining sentences at time t), which is independent of the policy parameters to prohibit the gradient from being passed to the policy parameters via the conditional likelihood. After obtaining the reward according to Equation (1), the policy parameters are updated at each time step according to Equation (3).

3.3 Policy Network

The state S_t in Equation (4) is designed to be informative on: 1) the local content of the sentence, 2) the global context of the sentence within the document, and 3) the current extraction history. To encode these three properties in the state, we use a local sentence encoder, a global context encoder, and an extraction history encoder respectively. Subsequently, the state is mapped by an extractor to an output score for each of the remaining sentences and the extraction stop signal. The overall framework of our model is depicted in Figure 1.

In the **Local Sentence Encoder** (LSE), ordered words $(w_1, w_2, \dots w_M)$ in a sentence s_i are first mapped onto word embeddings using a word embedding matrix. Subsequently, a N_l-layer bi-directional LSTM [17] transforms the word embeddings and maps them onto sentence embeddings l_{s_i} via a multi-head pooling layer (MHP) [8].

The **Global Context Encoder** (GCE) consists of a N_g-layer bi-LSTM that takes the L local sentence embeddings $(l_{s_1}, l_{s_2}, \dots l_{s_L})$ as inputs and produces for each sentence s_i an embedding g_{s_i} that encodes global contextual information such as the sentence's position in the document and information on neighboring sentences.

The **Extraction History Encoder** (EHE) encodes the extraction history information and produces the extraction history embedding $h_{s_i^r}$ for each remaining sentence s_i^r. At each time step t, the extraction history embedding $h_{s_i^r}$ consists of two main sources of information: 1) the context of the partially extracted sentences, and 2) the context of the remaining sentences. We encode the contextual information for each remaining sentence using the attention mechanism [18]: The EHE is composed of a stack of N_h itentical layers. Within one layer, there are two multi-head attention sublayers, similar to the Transformer's decoder part as described in Vaswani et al. [18]. One attention sublayer is used to perform multi-head attention on the extracted sentences to encode the contextual information of the extracted sentences, and the other is used to attend the remaining sentences' embeddings to encode the contextual information of the remaining sentences. The final outputs of the N_h^{th} layers of the EHE are treated as the extraction history embeddings for the remaining sentences.

There is no positional encoding and the EHE produces the extraction history embeddings non-autoregressively by attending to both precedent and subsequent positions. The EHE under such a setting is insensitive to the order of the previously selected sentences: Suppose that at time step t sentences A and B have been selected into the summary, no matter whether A was first selected or vice versa, the EHE will produce the same extraction history embedding $h_{s_i^r}$ for the remaining sentences. This is consistent with our previous definition of extraction history, as we believe that the sequential information of previously selected sentences is not needed in deciding whether to stop extraction and in avoiding redundancy.

The **Extractor** computes the score each remaining sentence and outputs an extraction stop signal. As input to the extractor, we form for each of the remaining sentences s_i^r an aggregated embedding by concatenating the local sentence embedding $l_{s_i^r}$, the global context embedding $g_{s_i^r}$, and the extraction history embedding $h_{s_i^r}$. As shown in Figure 1, to produce the score $u_{s_i^r}$, the concatenated embedding of remaining sentence s_i^r is passed to the fully connected layers with ReLU activation and then projected to a scalar by a Linear-1 layer followed by a sigmoid function. Note that the same fully connected layers are applied identically to all remaining sentences. To output a single vector, a multi-head pooling layer [8] is applied to the last hidden vectors of all remaining sentences. This vector is then passed to a linear layer with a sigmoid function, producing a stopping probability p_{stop}.

3.4 Training

We train the parameterized policy network according to the update rule in Equation (3). At each training iteration, an episode is sampled to compute the final return r and the action probabilities $\pi(A_t|S_t, \boldsymbol{\theta}_t)$ for all time steps t. An example episode with T extracted sentences looks like: $(S_0, s_{a_0}, \ldots, S_{T-1}, s_{a_{T-1}}, S_T, A_{\text{stop}}, r)$, where S_t represents the concatenated state information introduced in Section 3.3, s_{a_t} represents the selection of sentence a_t, A_{stop} represents the extraction stops at the final time step T, and r is the reward as defined in Equation (1). To encourage the agent to select compact summaries, we multiply the final reward r by a length penalty term $1/(T+1)$ [5]. Consequently, the return for each time step is $R_t \equiv \frac{r}{T+1}$.

Algorithm 1 The training algorithm.

```
Parameters: learning rate α
for each document-summary pair (D_i, G_i) do
    Compute local sentence embeddings l_{s_1},...,l_{s_L}
    Compute global context embeddings g_{s_1},...,g_{s_L}
    Sample an episode S_0,s_{a_0},...,S_{T-1},s_{a_{T-1}}, S_T,A_stop,r
        from the high-ROUGE episodes set E_p of document D_i
    for each time step: t = 0,1,...,T: do
        if t > 0 then
            Compute extraction history embeddings
                h_{s^r_1},...,h_{s^r_{L-E_t}} for remaining sentences
        else
            Initialize h_{s^r_1},...,h_{s^r_{L-E_0}} to 0
        Compute scores u_{s^r_1},...,u_{s^r_{L-E_t}} for remaining sentences
            and the stop likelihood p_stop
        Compute the action probability π(A_t|S_t, θ) according to Equation (4)
        θ ← θ + α (r/(T+1)) ∇ log π(A_t|S_t, θ)
```

Algorithm 1 summarizes the training procedure of our MemSum model. We initialize the extraction history embeddings to $\mathbf{0}$, since no sentences have been extracted at $t = 0$. E_t represents the number of sentences that have been extracted into the summary before time step t. Following the strategy in Narayan et al. [10] and Mohsen et al. [12], instead of sampling an episode following the current policy $\pi(\cdot|\cdot, \boldsymbol{\theta}_t)$, we sample an episode from a set $\mathbb{E}_p$ of episodes with high ROUGE scores, which enables the agent to quickly learn from optimal policies and to rapidly converge. Details on creating a set of high-ROUGE episodes for training are described further in the Appendix.

4 Experiments

In this section, we report implementation details of our model and describe the datasets used for training and for evaluation.

Datasets: The PubMed and arXiv datasets [19] were obtained from scientific repositories. In these two datasets the document to be summarized is the full body of a paper and the gold summary is the corresponding abstract. The GovReport dataset [20] contains U.S. government reports with gold summaries written by experts. Table 1 reports for each dataset the average document and summary length, as well as the number of document-summary pairs for training, validating, and testing.

Datasets	avg. doc. length		avg. summ. length		# of doc.-summ. pairs		
	# of words	# of sentences	# of words	# of sentences	Train	Valid	Test
PubMed	3016	88	203	7	116,937	6,633	6,657
arXiv	4938	206	220	10	202,880	6,436	6,440
GovReport	9409	307	553	18	17,517	974	973

Table 1: An overview of datasets used in this paper.

We simply treat the document as a list of sentences without considering section information, following a general setting adopted in recent works [14, 24] but unlike Cohan et al. [19], Xiao and Carenini [21], Huang et al. [20],

Implementation Details: We computed local sentence embeddings using pretrained Glove word embeddings [27] of dimension $d = 200$, keeping the word embeddings fixed during training. For the LSE, we used $N_l = 2$ bi-LSTM layers and for the GCE $N_g = 2$. For the EHE, we used $N_h = 3$ attention layers, and we set the number of attention heads to 8 and the dimension of the feed-forward hidden layer to 1024; during training we set the dropout rate to 0.1. The extractor consisted of 2 fully-connected hidden layers with output dimensions $2d$ and d, respectively. The total number of trainable parameters is 4.4 M, so our model is much smaller than BERT-based extractive summarization models in which merely the $\text{BERT}_{\text{BASE}}$ [25] contains 110 M trainable parameters.

We trained our model using the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$ [28], fixed learning rate $\alpha = 1e^{-4}$, and weight decay $1e^{-6}$. To store the network parameters for validating and inference, we used the exponential moving average strategy introduced in Karras et al. [29] with decay 0.999. During validating and testing, the agent extracted sentences in a deterministic way: after computing the scores $u_{s_i^r}$ for the remaining sentences and the stop likelihood p_{stop}, the agent stopped the extraction if $p_{\text{stop}} \geq p_{\text{thres}}$ or if the maximum admissible number $N_{\max}$ of extracted sentences was reached; otherwise, the agent selected the sentence with the largest score. The model was trained on 8 NVIDIA GeForce RTX 2080 Ti 11GB GPUs.

On the validating datasets we selected the best checkpoint of each model and determined the optimal $N_{\max}$ and stopping criterion p^*_{thres}. $N_{\max}$ was set to 7, 5, and 22 for the Pubmed, arXiv, and the GovReport, respectively. We set p^*_{thres} to 0.6, 0.5, and 0.6 for the PubMed, arXiv, and GovReport datasets, respectively.

Evaluation: We evaluated the performance of our model using F_1 ROUGE [9], including ROUGE-1, ROUGE-2, and ROUGE-L for measuring unigram, bigram, and longest common subsequence. We used a native python implementation of ROUGE for training and validating and used the official pyrouge package[2] for testing [5, 14].

5 Results and Discussion

Here we present the results on various extractive summarization tasks. In particular, we analyze the influences of history awareness and of the early stopping mechanisms.

[2] We use pyrouge, a python wrapper of ROUGE, with the parameters "-a -c 95 -m -n 4 -w 1.2".

Model	PubMed			arXiv		
	R-1	R-2	R-L	R-1	R-2	R-L
ORACLE	61.99	34.95	56.76	60.00	30.60	53.03
Extractive summarization models						
SumBasic	37.15	11.36	33.43	29.47	6.95	26.3
Lead-10	37.45	14.19	34.07	35.52	10.33	31.44
SummaRuNNer	43.89	18.78	30.36	42.81	16.52	28.23
Atten-Cont	44.85	19.70	31.43	43.62	17.36	29.14
Sent-CLF	45.01	19.91	41.16	34.01	8.71	30.41
Sent-PTR	43.30	17.92	39.47	42.32	15.63	38.06
Abstractive summarization models						
PEGASUS	45.97	20.15	41.34	44.21	16.95	38.83
BigBird	46.32	20.65	42.33	46.63	19.02	41.77
Dancer	46.34	19.97	42.42	45.01	17.60	40.56
Hepos-Sinkhorn	47.93	20.74	42.58	47.87	20.00	41.50
Hepos-LSH	48.12	21.06	42.72	48.24	20.26	41.78
Our Models						
Lead-10 (ours)	37.71	14.13	34.30	34.88	10.46	30.96
MemSum (ours)	**49.25**	**22.94**	**44.42**	**48.42**	**20.30**	**42.54**

Table 2: Results on the PubMed and arXiv test sets.

Model	R-1	R-2	R-L
ORACLE	75.56	45.91	72.51
Abstractive summarization baselines			
Hepos-LSH	55.00	21.13	51.67
Hepos-Sinkhorn	56.86	22.62	53.82
Our Models			
Lead-20 (ours)	50.94	19.53	48.45
MemSum (ours)	**59.43**	**28.60**	**56.69**

Table 3: Results on the GovReport test set.

Human-written Summary:

(...) While CMS is generally required to disallow, or ***recoup, federal funds*** from states for ***eligibility-related improper payments*** if the state's ***eligibility error rate exceeds 3 percent***, it has not done so for decades, because the method it used for calculating eligibility error rates was found to be insufficient for that purpose. To address this, in July 2017, CMS ***issued revised procedures through which it can recoup funds for eligibility errors, beginning in fiscal year 2022***. (...)

Hepos-Sinkhorn (abstractive):

(...) The selected states also reported that they did not have adequate processes to address these issues. CMS has taken steps to improve its oversight of the Medicaid program, including issuing guidance to states on the use of MAGI-exempt bases for determining eligibility, but these efforts have not been fully implemented. (...)

MemSum (ours, extractive):

(...) In 1983, CMS implemented its statutory requirement to ***recoup funds*** associated with Medicaid ***eligibility-related improper payments*** for states with an ***eligibility error rate above 3 percent*** through its MEQC program. (...) However, the agency has ***introduced new procedures through which it can, under certain circumstances, begin to recoup funds based on eligibility errors in fiscal year 2022***. (...)

Table 4: Sample summary extracted by MemSum and comparison with the abstractive summary generated by Hepos-Sinkhorn [20].

5.1 Results Comparison

We compared our MemSum with both extractive and abstractive models on the PubMed, arXiv, and GovReport datasets. Extractive baselines include SumBasic [30], Lead-10 (which directly uses the first 10 sentence in the document as the summary) [31], SummaRuNNer [13, 21], Atten-Cont [21], and Sent-CLF and Sent-PTR [32]. Abstractive summarization models include PEGASUS [33], BigBird [34], Dancer [31] and Hepos [20]. Among these baselines, Hepos achieved the state-of-the-art in long document summarization using a large-scale pretrained BART model [35] with memory-efficient attention encoding schemes including Locality Sensitive Hashing [36] (Hepos-LSH) and Sinkhorn (Hepos-Sinkhorn).

The oracle model is based on the greedy approach [13] that sequentially selects from the document the optimal sentence that maximally improves the average of R-1 and R-2 of selected sentences. We also evaluated the performance of our own Lead-10 baseline on PubMed and arXiv datasets.

5.1.1 Performance on PubMed Dataset

The performance comparison on the PubMed dataset is shown in Table 2. The Lead-10 baseline yielded ROUGE scores close to the ones provided in Gidiotis and Tsoumakas [31], indicating that our test dataset and ROUGE computation process is consistent with previous works. MemSum achieved the new state-of-the-art performance compared with both extractive and abstractive summarization models. Compared with the extractive baselines, MemSum outperformed Sent-CLF by 4.24/3.03/3.26 of R-1/R-2/R-L scores respectively. With many fewer trainable parameters, MemSum also outperformed Hepos-LSH, the state-of-the-art abstractive model, by 1.13/1.88/1.70 of R-1/R-2/R-L scores. The high R-2 and R-L scores of our MemSum model indicate that MemSum-extracted summaries have high overlap with gold summaries on bigrams and longer n-grams.

5.1.2 Performance on arXiv Dataset

On the arXiv dataset, MemSum largely outperformed the previous extractive summarization model Sent-PTR, improving R-1/R-2/R-L by 6.10/4.67/4.48. MemSum also exhibited better performance than the state-of-the-art abstractive summarization model Hepos-LSH, especially in terms of ROUGE-L.

5.1.3 Performance on GovReport Dataset

The GovReport dataset is a newly proposed dataset and has been only tested with abstractive summarization methods. We tested three extractive methods on this dataset: 1) an extractive Oracle model, 2) a Lead-20 baseline that directly uses the first 20 sentences in the document as the summary, and 3) our MemSum model. As shown in Table 3, we observed that the Lead-20 is a strong baseline. Furthermore, our MemSum model outperforms the state-of-the-art abstractive summarization model, Hepos with Sinkhorn attentions, by 2.57/5.98/2.87 of R-1/R-2/R-L scores.

Table 4 shows a comparison between the summary extracted by MemSum and the one generated by Hepos-Sinkhorn. Compared with the abstractively generated summary, the summary extracted by our MemSum has more precise and relevant content with respect to the gold summary. These results indicate that the human-written summaries in the GovReport dataset tend to be highly extractive.

Model	**R-1**	**R-2**	**R-L**	**Avg. summ. length (# sentences)**
MemSum$_{w/o_hist}$	48.08	22.77	43.55	7.0
MemSum$_{full}$ (w/o early stopping)	48.25	22.63	43.70	7.0
MemSum$_{full}$ (w/ early stopping)	**49.25**	**22.94**	**44.42**	6.0

Table 5: Ablation study on the PubMed dataset. The full MemSum model achieved higher ROUGE scores than the MemSum$_{w/o_hist}$ model.

Model	**R-1**	**R-2**	**R-L**	**Avg. duplicate ratio**
MemSum$_{w/o_hist}$	42.82	18.18	36.68	0.41
MemSum$_{w/o_hist}$+Trigram blocking	46.85	19.93	42.40	0.0
MemSum$_{full}$ (w/o early stopping)	48.21	22.59	43.76	0.0
MemSum$_{full}$ (w/ early stopping)	**49.16**	**22.78**	**44.39**	0.0

Table 6: Performance on the redundant PubMed dataset.

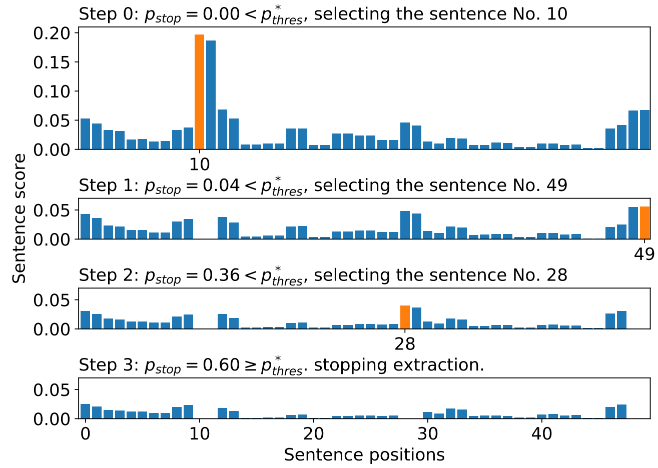

Figure 2: The sentence scores of 50 sentences computed by the full MemSum model at extraction steps 0 to 3 (histograms from top to bottom). In the document there is artificial redundancy in that the $(2n)_{\text{th}}$ and the $(2n+1)_{\text{th}}$ sentences are identical ($n = 0, 1, ..., 24$).

5.2 *Analysis*

5.2.1 *Influence of History Awareness*

In order to analyze the influence of extraction history awareness, we performed an ablation study by comparing the full MemSum model with a MemSum model without history awareness (*MemSum$_{w/o_hist}$* for short). We removed the Extraction History Encoder in the original MemSum model, so that the input to the Extractor is only the concatenation of the local sentence embedding and the global context embedding. We used the Extractor to compute the scores for all sentences in one step, and then sampled sentences following the BanditSum policy [14] introduced in Section 2.

First, we assessed whether early-stopping from extraction history awareness helps MemSum to extract more compact summaries. To verify this, we compared the full MemSum and the MemSum without history MemSum$_{w/o_hist}$ on the PubMed dataset. In the MemSum$_{w/o_hist}$, extraction stops until the maximum number of extracted sentences $N_{\max}$ is reached. We tested the full MemSum under two different settings: 1) always extracting $N_{\max}$ sentences, and 2) stopping extraction when either the stop likelihood $p_{\text{stop}} \geq p^*_{\text{thres}}$ or when $N_{\max}$ is reached. We used the same $N_{\max}$ for both models and chose p^*_{thres} as introduced in the implementation details in Section 4. The results are shown in Table 5.

When both models always extracted $N_{\max}$ sentences, the full MemSum performed marginally better than the MemSum$_{w/o_hist}$ on R-1 and R-L. When early stopping was allowed, the full MemSum model achieved higher ROUGE scores than the MemSum$_{w/o_hist}$ model, while producing summaries with fewer sentences, indicating a higher degree of compactness.

Second, we hypothesized that the extraction history allows MemSum to avoid sentences that are similar to existing sentences in the current partial summary, intuitively mimicking what humans do when extractively summarizing documents. To verify this, we created a redundant PubMed dataset in which we repeated each sentence in the document, with the replicated sentences immediately following the originals. On this redundant PubMed dataset we trained and tested the full MemSum and the MemSum$_{w/o_hist}$, and we compared different models in terms of ROUGE scores and the average *duplicate ratio* that is defined by the average ratio of the number of duplicated sentences to the total number of sentences in the extracted summaries.

As reported in Table 6, for the MemSum$_{w/o_hist}$ model, the average duplicate ratio was 0.41, which means that on average 41% of the sentences in the extracted summaries were duplicated. Along with the high duplicate ratio came a significant decrease in ROUGE scores of MemSum$_{w/o_hist}$. By contrast, the performance of the full MemSum models with history awareness was only slighted affected when comparing the results of the MemSum$_{full}$ (w/ early stopping) model in Table 5 and Table 6.

To better reveal how the full MemSum model avoids redundancy, we let it summarize a document sampled from the test set of the redundant PubMed dataset and monitored the sentence scores produced by the Extractor during each extraction step. The results are shown in Figure 2. At time step 0, the 10_{th} sentence obtained the maximum score and was thus selected into the summary. At time step 1, we noticed that the 11_{th} sentence, which is a replica of the 10_{th} sentence, had a score close to zero. The same was also true for the other selected sentences, revealing competent repetition avoidance of the Extractor.

Because the EHE is insensitive to the extraction order and to sentence position information, as described in Section 3.3, we can conclude that the full MemSum avoids redundancy by evaluating the similarity between selected and remaining sentences, rather than by “remembering" the position of the previously selected sentences.

Meanwhile, results in Table 6 show that using the Trigram Blocking method that skips a sentence if it has a trigram that overlaps with the current summary [23] is also successful in avoiding repetitive sentences. However, the associated ROUGE scores were significantly lower than those of the full MemSum with awareness of extraction history. In summary, the history-aware MemSum model spontaneously learns an optimized strategy to avoid redundant sentences without explicit human guidance

Early stopping mechanism	R-1	R-2	R-L	Avg. summ. length (# sentences)
stop when selecting STOP sentence	47.18	21.81	42.20	3.9
stop when $p_{\text{stop}} \geq p^*_{\text{thres}}$	**49.25**	**22.94**	**44.42**	6.0

Table 7: Performance on the PubMed dataset of MemSum models using different early stopping mechanisms.

or crude rules, and thus shows better robustness to potential redundancy in documents.

5.2.2 *Choice of Early-Stopping Mechanism*

Apart from stopping the extraction early when the stop likelihood p_{stop} exceeds the threshold p^*_{thres}, an alternative way of stopping is to insert a special stop sentence (e.g. "STOP") into the document. Once the agent selects the stop sentence, the extraction stops early. To train such an agent we followed the setting introduced in Section 3.4, except that we added the special stop sentence as the last extracted sentence, so that an example episode with T extracted sentences looks like $(S_0, s_{a_0}, \ldots, S_{T-1}, s_{a_{T-1}}, S_T, s_{\text{STOP}}, r)$ where s_{STOP} represents the stop sentence.

The performance of this agent on the PubMed dataset is shown in Table 7: it produces summaries with fewer extracted sentences on average and lower ROUGE scores than the model that stops based on the stop likelihood p_{stop}. We believe this result is due to the predictable positive reward r provided by the stop sentence, which leads to preference of this last action and to increasing the likelihood of taking it early. Therefore, we chose the stop likelihood as the default early-stopping mechanism.

6 Conclusion

Extractive summarization can be achieved effectively with a multi-step episodic Markov decision process with history awareness. Using encoders of local sentence, global context, and extraction history, the MemSum agent has access to information that is intuitively also used by humans when they summarize a document. Awareness of the extraction history helps MemSum to produce compact summaries and to exhibit robustness against redundancy in the document. As a lightweight model, MemSum outperforms both extractive and abstractive baselines on diverse long document summarization tasks.

References

[1] Oguzhan Tas and Farzad Kiyani. A survey automatic text summarization. *PressAcademia Procedia*, 5(1):205–213, 2007.

[2] Mahak Gambhir and Vishal Gupta. Recent automatic text summarization techniques: A survey. *Artif. Intell. Rev.*, 47(1):1–66, January 2017. ISSN 0269-2821. doi: 10.1007/s10462-016-9475-9. URL https://doi.org/10.1007/s10462-016-9475-9.

[3] Ani Nenkova and Kathleen McKeown. A Survey of Text Summarization Techniques. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 43–76. Springer US, Boston, MA, 2012. ISBN 978-1-4614-3223-4. doi: 10.1007/978-1-4614-3223-4_3. URL https://doi.org/10.1007/978-1-4614-3223-4_3.

[4] N Moratanch and S Chitrakala. A survey on extractive text summarization. In *2017 international conference on computer, communication and signal processing (ICCCSP)*, pages 1–6. IEEE, 2017.

[5] Ling Luo, Xiang Ao, Yan Song, Feiyang Pan, Min Yang, and Qing He. Reading like her: Human reading inspired extractive summarization. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3024–3034, 2019.

[6] Pengcheng Liao, Chuang Zhang, Xiaojun Chen, and Xiaofei Zhou. Improving abstractive text summarization with history aggregation. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2020.

[7] Peter J. Liu*, Mohammad Saleh*, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Hyg0vbWC-.

[8] Yang Liu and Mirella Lapata. Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1500. URL https://www.aclweb.org/anthology/P19-1500.

[9] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W04-1013.

[10] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1747–1759, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1158. URL https://www.aclweb.org/anthology/N18-1158.

[11] Kamal Al-Sabahi, Zhang Zuping, and Mohammed Nadher. A hierarchical structured self-attentive model for extractive document summarization (hssas). *IEEE Access*, 6:24205–24212, 2018.

[12] Farida Mohsen, Jiayang Wang, and Kamal Al-Sabahi. A hierarchical self-attentive neural extractive summarizer via reinforcement learning (hsasrl). *Applied Intelligence*, pages 1–14, 2020.

[13] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. Summarunner: A recurrent neural network based sequence

model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 3075–3081. AAAI Press, 2017.

[14] Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. BanditSum: Extractive summarization as a contextual bandit. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3739–3748, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1409. URL https://www.aclweb.org/anthology/D18-1409.

[15] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256, May 1992. ISSN 0885-6125. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

[16] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[19] 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. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2097. URL https://www.aclweb.org/anthology/N18-2097.

[20] Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. Efficient attentions for long document summarization. *arXiv preprint arXiv:2104.02112*, 2021.

[21] Wen Xiao and Giuseppe Carenini. Extractive summarization of long documents by combining global and local context. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3011–3021, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1298. URL https://www.aclweb.org/anthology/D19-1298.

[22] Xingxing Zhang, Furu Wei, and Ming Zhou. HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5059–5069, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1499. URL https://www.aclweb.org/anthology/P19-1499.

[23] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1387. URL https://www.aclweb.org/anthology/D19-1387.

[24] Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. Extractive summarization as text matching. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6197–6208, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.552. URL https://www.aclweb.org/anthology/2020.acl-main.552.

[25] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

[26] Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1693–1701, Cambridge, MA, USA, 2015. MIT Press.

[27] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://www.aclweb.org/anthology/D14-1162.

[28] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *ICLR (Poster)*, 2015. URL http://arxiv.org/abs/1412.6980.

[29] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Hk99zCeAb.

[30] Lucy Vanderwende, Hisami Suzuki, Chris Brockett, and Ani Nenkova. Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion.

Inf. Process. Manage., 43(6):1606–1618, November 2007. ISSN 0306-4573. doi: 10.1016/j.ipm.2007.01.023. URL https://doi.org/10.1016/j.ipm.2007.01.023.

[31] Alex Gidiotis and Grigorios Tsoumakas. A divide-and-conquer approach to the summarization of long documents. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:3029 – 3040, 11 2020. doi: 10.1109/TASLP.2020.3037401.

[32] Jonathan Pilault, Raymond Li, Sandeep Subramanian, and Chris Pal. On extractive and abstractive neural document summarization with transformer language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9308–9319, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.748. URL https://www.aclweb.org/anthology/2020.emnlp-main.748.

[33] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/zhang20ae.html.

[34] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf.

[35] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 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*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://www.aclweb.org/anthology/2020.acl-main.703.

[36] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkgNKkHtvB.

[37] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27, pages 3104–3112. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf.

[38] Markus Freitag and Yaser Al-Onaizan. Beam search strategies for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 56–60, Vancouver, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-3207. URL https://www.aclweb.org/anthology/W17-3207.

Dataset	Validating R-1	Validating R-2	Validating R-L	Test R-1	Test R-2	Test R-L
PubMed	49.14	22.92	44.33	49.25	22.94	44.42
arXiv	48.23	20.17	42.31	48.42	20.30	42.54
GovReport	59.29	28.57	56.46	59.43	28.60	56.69

Table 8: Validating and testing scores of the MemSum model tested on the PubMed, the arXiv and the GovReport datasets.

Dataset	avg. doc. length (words)	Avg. extractive summ. length (# sentences)	Avg. extractive summ. time (ms)
PubMed	3016	6.0 ± 1.2	91.7 ± 8.6
arXiv	4938	4.8± 0.5	114.0 ± 5.0
GovReport	9409	21.7 ± 1.8	197.0 ± 14.8

Table 9: Average extractive summarization time of our MemSum on different datasets.

A Computing Hardware

We trained our MemSum model and its variations on 8 NVIDIA GeForce RTX 2080 Ti 11GB GPUs. During testing, we used a single NVIDIA TITAN X Pascal 12GB GPU.

B Comparison of Validating and Testing Performance

We compare the validating and testing performance of the MemSum model on the following datasets: the PubMed dataset [19], the arXiv dataset [19] and the GovReport dataset [20]. The results are shown in Table 8.

C Summarization Time

We analyzed the average time taken by MemSum to extractively summarize a source document from the test set. The average summarizaion time is positively correlated with the document length and the number of extracted sentences, Table 9. On the one hand, on longer documents, it takes longer to compute the scores of remaining sentences, which delays the action of either stopping extraction or selecting a sentence. On the other hand, the more sentences are needed to be extracted, the more actions are needed of selecting sentences within an episode.

D Selection of optimal stopping threshold

The stopping threshold p_{thres} is an important hyperparameter that sets the stopping probability in an episode, as described in the Implementation Details. We determined the optimal stopping threshold p^*_{thres} as follows: For each data set and each stopping threshold $p_{\text{thres}} \in \{0.1, 0.2, \ldots, 1.0\}$ we chose as optimal stopping threshold p^*_{thres} the one with maximal ROUGE score on the corresponding validating set.

The ROUGE scores as a function of stopping threshold are shown in Figure 3, 4 and 5 on the validating set of the PubMed, the arXiv and the GovReport data set, respectively. The functions exhibit a local maximum between 0.1 and 1.0, which

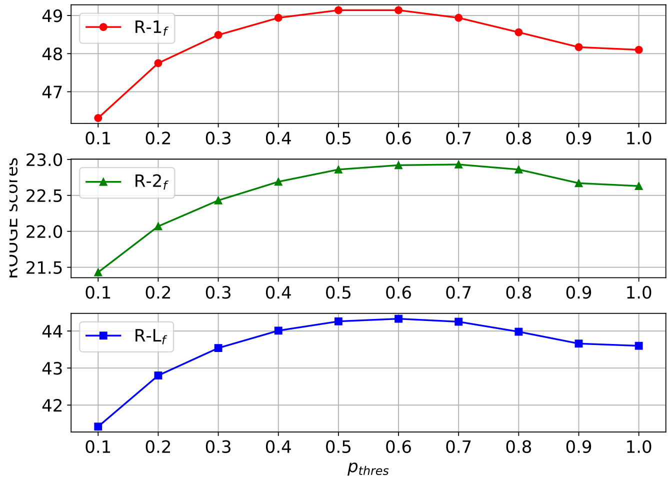

Figure 3: The ROUGE scores for different stopping thresholds p_{thres} on the PubMed validating set.

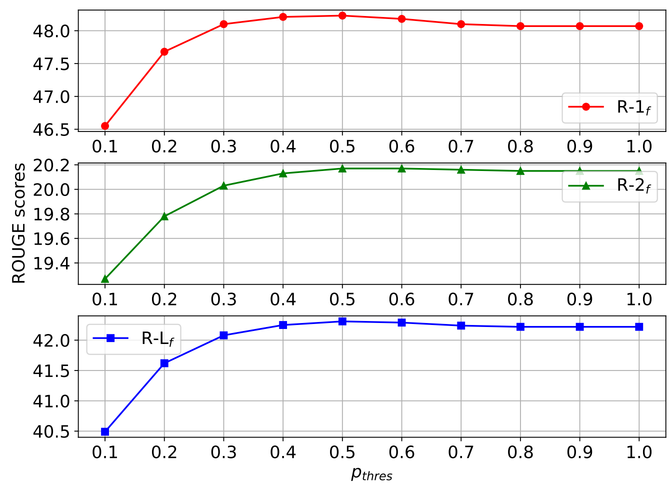

Figure 4: The ROUGE scores for different stopping thresholds p_{thres} on the arXiv validating set.

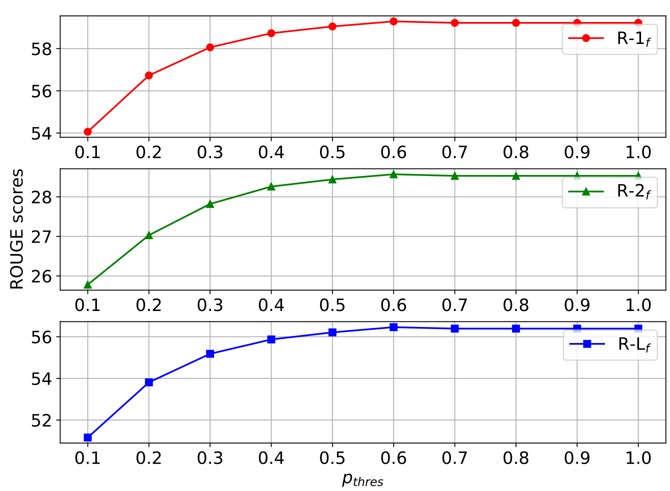

Figure 5: The ROUGE scores for different stopping thresholds p_{thres} on the GovReport validating set.

implies that when p_{thres} is too low, summaries tend to be too short, while when p_{thres} is too high, summaries will be unduly lengthy. We chose $p^*_{\text{thres}} = 0.6,\ 0.5$ and 0.6 for the PubMed, the arXiv, and the GovReport dataset, respectively.

E Creating High-ROUGE Episodes for Training

As introduced in Section 3.4 and Algorithm 1 in the main paper, at each training iteration, we sampled a high-ROUGE episode from the set $\mathbb{E}_p$. An episode can be viewed as a sequence of state-action pairs as well as the final reward, such as $(S_0, s_{a_0}, \ldots, S_{T-1}, s_{a_{T-1}}, S_T, A_{\text{stop}}, r)$. Here, $\{s_{a_0} \ldots s_{a_{T-1}}\}$ is the extracted summary consisting of a set of T sentences, and r is the average of the associated ROUGE-1, ROUGE-2, and ROUGE-L F1 scores.

The high-ROUGE episodes set $\mathbb{E}_p$ contains multiple episodes with high average ROUGE-1/2/L scores. In [13], a greedy approach was proposed to select candidate summaries by sequentially selecting from the source document the optimal sentence that maximally improves the average ROUGE-1/2/L score once added to the current subset of selected sentences. To obtain not a single episode in $\mathbb{E}_p$ but multiple episodes with high average ROUGE-1/2/L scores, we modified the greedy approach by considering not only the optimal sentence at each sentence selection step but also $B-1$ sub-optimal sentences. This sentence-sampling step is repeated for each of these B new subsets to result in a potentially exponentially growing number of high ROUGE-score episodes. This process stops until no sentence can further improve the average ROUGE-1/2/L score or a maximum number $N_{\max}$ of selected sentences per episode is reached. B can be considered as the beam size, analogous to beam search strategies in neural machine translation [37, 38]. We set $B = 2$ by default.

We refer an episode $(S_0, s_A, S_1, s_B, S_2, s_C, S_3, A_{\text{stop}}, r)$ as "(s_A, s_B, s_C)" for simplicity. Because permuted episodes (s_A, s_B, s_C), (s_A, s_C, s_B), and (s_C, s_B, s_A) have the same average ROUGE-1/2/L score, we decided to equally sample them (presumably to avoid overfitting). This decision does not interfere with our usage of extraction history, because under (s_A, s_B, s_C) the agent learns to extract s_C from $\{s_A, s_B\}$, while under (s_C, s_B, s_A) it learns to extract s_A from $\{s_B, s_C\}$. Thus, history plays a role in both cases.

F Examples of Extracted Summaries

We showed the summarization examples in Table 10 and 11. In Table 10 we compared our MemSum trained on the arXiv dataset and the Dancer Pegasus [31] summaries of the same paper. MemSum achieved higher ROUGE-1 F score than Dancer Pegasus. In Table 11 we showed the extractive summary of this paper itself using our MemSum model.

Title	BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
Original Abstract	We introduce a new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be finetuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial taskspecific architecture modifications. BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement)
DANCER PEGASUS summary	Language model pre-training has been shown to be effective for improving many natural language processing tasks such as sentence-level paraphrasing and entity recognition tasks. However, current approaches to pre-trained language models are restricted to unidirectional language models. In this paper, we propose a new approach to pre-trained language models based on bidirectional encoder transformers (BERT). BERT is inspired by the pre-training objective of cloze task (Taylor et al., 1953), where the goal is to predict some masked language representations from the input. We introduce BERT and its detailed implementation in this paper. The BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. Rich unsupervised pre-training is an integral part of many language understanding systems. In particular, these results enable even low-resource tasks to benefit from deep unidirectional architectures. Our major contribution is further generalizing these findings to deep bidirectional architectures, allowing the same pre-trained model to successfully tackle a broad set of NLP tasks.
ROUGE1-F1	36.52
MemSum summary	Language model pre-training has been shown to be effective for improving many natural language processing tasks. In this paper, we improve the fine-tuning based approaches by proposing BERT: Bidirectional Encoder Representations from Transformers. The masked language model randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked word based only on its context. Unlike Radford et al. (2018), which uses unidirectional language models for pre-training, BERT uses masked language models to enable pretrained deep bidirectional representations. BERT is the first finetuning based representation model that achieves state-of-the-art performance on a large suite of sentence-level and token-level tasks, outperforming many task-specific architectures.
ROUGE1-F1	44.29

Table 10: Example summaries for Dancer Pegasus [31] and MemSum.

Title	(This paper) MemSum: Extractive Summarization of Long Documents using Multi-step Episodic Markov Decision Processes
Original Abstract	We introduce MemSum (Multi-step Episodic Markov decision process extractive SUMmarizer), a reinforcement-learning-based extractive summarizer enriched at any given time step with information on the current extraction history. Similar to previous models in this vein, MemSum iteratively selects sentences into the summary. Our innovation is in considering a broader information set when summarizing that would intuitively also be used by humans in this task: 1) the text content of the sentence, 2) the global text context of the rest of the document, and 3) the extraction history consisting of the set of sentences that have already been extracted. With a lightweight architecture, MemSum nonetheless obtains state-of-the-art test-set performance (ROUGE score) on long document datasets (PubMed, arXiv, and GovReport). Supporting analysis demonstrates that the added awareness of extraction history gives MemSum robustness against redundancy in the source document.
MemSum summary	For each sentence in the document and at each time step in the process, we define a sentence state composed of three sub-states: 1) the content of the sentence, 2) the context of the sentence within the document, and 3) information on the extraction history, including the previously selected set of unordered sentences versus the remaining sentences. To efficiently encode local and global sentence states, we design an extraction agent based on LSTM networks. The contributions of our work are as follows: • We propose to treat extractive summarization as a multistep episodic MDP that is aware of the extraction history. • We show that the awareness of the extraction history allows our model to extract more compact summaries and behave more robustly to redundancies in documents than models without history awareness. • Our model outperforms both extractive and abstractive summarization models on PubMed, arXiv and GovReport.
ROUGE1-F1	51.39

Table 11: MemSum summary of this paper.