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

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

Nianlong Gu
Institute of Neuroinformatics,
University of Zurich and
ETH Zurich
nianlong@ini.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 each step with information on the current extraction history. When MemSum iteratively selects sentences into the summary, it considers a broad information set 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 obtains state-of-the-art test-set performance (ROUGE) in summarizing long documents taken from PubMed, arXiv, and GovReport. Ablation studies demonstrate the importance of local, global, and history information. A human evaluation confirms the high quality and low redundancy of the generated summaries, stemming from MemSum's awareness of extraction history.

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 (Tas and Kiyani, 2007). Text summarization methods can be categorized into abstractive and extractive summarization (Gambhir and Gupta, 2017; Nenkova and McKeown, 2012). Given a document d consisting of an ordered list of N sentences, *extractive summarization* aims to pick up M ($M \ll N$) sentences as the summary of the document. The extracted summaries tend to be both grammatically and semantically more reliable than abstractive summaries (Liu* et al., 2018; Liu and Lapata, 2019a; Luo et al., 2019; Liao et al., 2020), as they are directly selected from the source text.

Extractive summarization is usually modeled as two sequential phases (Zhou et al., 2018): 1) *sentence scoring* and 2) *sentence selection*. In the sentence scoring phase, an affinity score is computed for each sentence by neural networks such as bidirectional RNNs (Dong et al., 2018; Narayan et al., 2018; Luo et al., 2019; Xiao and Carenini, 2019) or BERT (Zhang et al., 2019; Liu and Lapata, 2019b). In the sentence selection phase, sentences are selected by either i) predicting a label (1 or 0) for each sentence based on its score, and selecting sentences with label 1 (Zhang et al., 2019; Liu and Lapata, 2019b; Xiao and Carenini, 2019), or ii) ranking sentences based on their scores and selecting the top K sentences as the summary (Narayan et al., 2018), or iii) sequentially sampling sentences without replacement, where the normalized scores of the remaining sentences are used as sampling likelihoods (Dong et al., 2018; Luo et al., 2019).

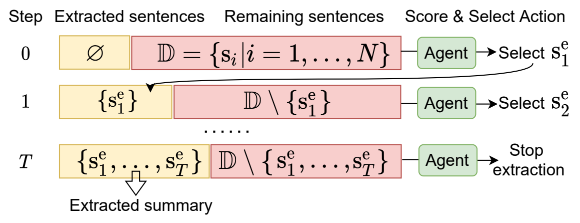


Figure 1: We modeled extractive summarization as a multi-step iterative process of scoring and selecting sentences. s_i represents the i_{th} sentence in the document $\mathbb{D}$.

In these approaches, sentence scores are generally not updated based on the current partial summary of previously selected sentences, indicating a lack of knowledge of *extraction history*. We deem extractive summarizers that are not aware of the extraction history to be susceptible to redundancy in a document, because they will repeatedly add sentences with high scores to a summary, regardless of whether similar sentences have been selected before. And, redundancy leads to performance decreases evaluated by ROUGE F1.

In this paper, we propose to model extractive summarization as a multi-step episodic Markov Decision Process (MDP). As shown in Figure 1, at each time step in an episode, we define a *sentence state* composed of three sub-states: 1) the local content of the sentence, 2) the global context of the sentence within the document, and 3) information on the extraction history, including the previously selected set of unordered sentences and the remaining sentences. At each time step, the policy network (agent) takes the current sentence state as input and produces 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 (Narayan et al., 2018; Dong et al., 2018; Luo et al., 2019) 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 (Hochreiter and Schmidhuber, 1997). To encode the extraction history and to select actions, we use a reduced number of attention layers (Vaswani et al., 2017) of relatively low dimensionality. These choices enable our model to be easily trainable and to summarize long documents such as scientific papers (Cohan et al., 2018; Huang et al., 2021) or reports (Huang et al., 2021).

The **contributions** of our work are as follows: 1) We propose to treat extractive summarization as a multi-step episodic MDP that is aware of the extraction history. 2) We show that extraction-history awareness allows our model to extract more compact summaries than models without history awareness and behave more robustly to redundancies in documents. 3) Our model outperforms both extractive and abstractive summarization models on PubMed, arXiv (Cohan et al., 2018), and GovReport (Huang et al., 2021) datasets. 4) Finally, human evaluators rate the MemSum summaries to be of higher quality than those from a competitive approach, especially by virtue of lower redundancy[1].

2 Related Work

Extraction history awareness was previously considered in NeuSum (Zhou et al., 2018), where a GRU encoded previously selected sentences into a hidden vector that then was used to update the scores of the remaining sentences to bias the next selection. NeuSum contains no stopping mechanism and therefore it can only extract a fixed number of sentences, which likely is sub-optimal. Also, the potential benefits of extraction history have not been quantified and so the idea remains unexplored to a large extent.

Recently, BERT-based extractors such as MatchSum (Zhong et al., 2020) achieved SOTA performance in extractive summarization of relatively short documents from the CNN/DM (Hermann et al., 2015) dataset. However, the quadratic computational and memory complexities (Huang et al., 2021) of such models limit their scalability for summarizing long documents with thousands of tokens, which is common for scientific papers and government reports. Although large pre-trained transformers with efficient attention (Huang et al., 2021) have been adapted for abstractive summarization of long documents, we believe that extractive summarization is more faithful in general, which is why we chose an extractive approach.

3 Model

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

3.1 Policy Gradient Methods

In an episodic task with a terminal state (i.e. *end of summary*), policy gradient methods aim to maximize the objective function $J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[R_0]$, 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 when the summary is complete. 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 according to Equation (1), so $R_t \equiv R_0 = r$. The reward r is usually expressed as (Dong et al., 2018):

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

According to the REINFORCE algorithm (Williams, 1992), 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

[1]Our code and data are available at `https://github.com/nianlonggu/MemSum`

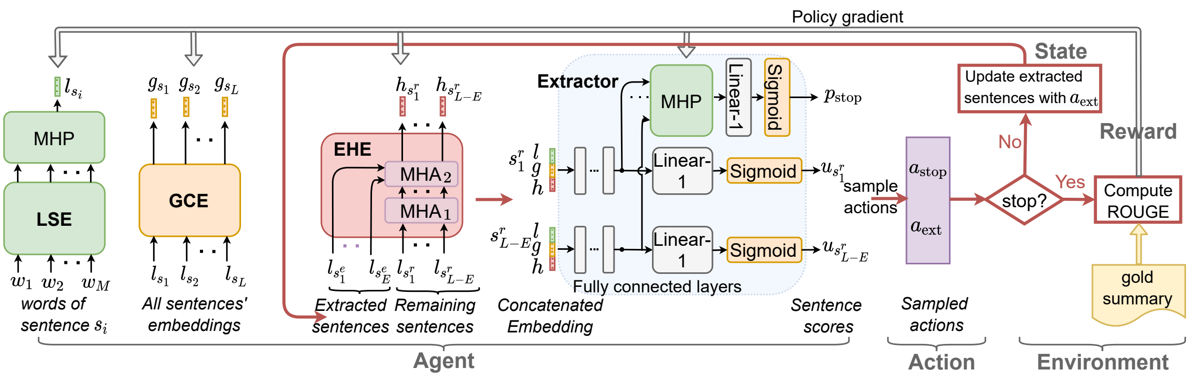


Figure 2: The architecture of our MemSum extractive summarizer with a multi-step episodic MDP policy. With the updating of the extraction-history embeddings h at each time step t, the scores u of remaining sentences and the stopping probability p_{stop} are updated as well.

state S_t. With α as the learning rate, the parameter update rule is (Sutton and Barto, 2018):

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

3.2 Multi-step Episodic MDP Policy

Different from one-step episodic MDP policies (Narayan et al., 2018; Dong et al., 2018; Luo et al., 2019) that extract the entire summary via a single action, we define an episode, i.e., the generation of a summary, consisting of multiple time steps. At each time step t, corresponding to extracting sentence number t, the action A_t is either to stop extraction or to select a sentence s_{a_t} from the remaining sentences. The agent's policy is:

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

where I_t denotes the index set of remaining sentences at time step t. If the agent does not stop, it first computes a score u_j for each remaining sentence and samples a sentence s_{a_t} according to the probability 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 calculating the reward according to Equation (1), the policy parameters are updated according to Equation (3) (for all time steps).

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 2.

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 (Hochreiter and Schmidhuber, 1997) transforms the word embeddings and maps them onto sentence embeddings l_{s_i} via a multi-head pooling layer (MHP) (Liu and Lapata, 2019a).

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. The EHE is composed of a stack of N_h identical layers. Within one layer, there are two multi-head attention sublayers, as contained in the transformer decoder in Vaswani et al. (2017). One sublayer is used to perform multi-head self-attention (MHA) among the local embeddings of the remaining sentences, so that each remaining sentence can capture the context provided by other remaining sentences. The other attention sublayer is used to perform multi-head attention over the embeddings of extracted sentences to enable each remaining sentence to attend to all the extracted sentences. The output of the two attention sublayers, one for each remaining sentence, captures the contextual information of both extracted and remaining sentences. The final output of the N_h^{th} layer of the EHE constitutes the extraction history embedding, one for each remaining sentence.

There is no positional encoding and the EHE produces the extraction history embeddings non-autoregressively by attending to both precedent and subsequent positions. Consequently, the extraction history embeddings $h_{s_i^r}$ for the remaining sentences are invariant to the order of the previously selected sentences. We believe that the sequential information of previously selected sentences is not crucial for reducing redundancy and for deciding whether to stop extraction or not.

The **Extractor** computes the score of 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 2, to produce the score $u_{s_i^r}$, the concatenated embedding of remaining sentence s_i^r is passed to 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. We deem that the extractor can learn to stop extraction based on the remaining sentences' states. Therefore, we apply an MHP to the last hidden vectors of all remaining sentences to output a single vector. 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 $1/(T+1)$ (Luo et al., 2019). Consequently, the return $R_t \equiv \frac{r}{T+1}$.

Algorithm 1 The training algorithm.

```
Parameters: learning rate $\alpha$
1:  for each document-summary pair ($D_i$, $G_i$) do
2:      LSE outputs local sent. embed $l_{s_1}, \ldots, l_{s_L}$
3:      GCE outputs global context embed $g_{s_1}, \ldots, g_{s_L}$
4:      Sample an episode $S_0, s_{a_0}, \ldots, S_{T-1}, s_{a_{T-1}}$, $S_T, A_{\text{stop}}, r$ from the high-ROUGE episodes set $\mathbb{E}_p$ of document $D_i$
5:      for each time step: $t$ = 0,1,...,T: do
6:          if $t > 0$ then
7:              EHE outputs extraction history embed $h_{s_1^r}, \ldots, h_{s_{L-E_t}^r}$ for remaining sentences
8:          else
9:              Initialize $h_{s_1^r}, \ldots, h_{s_{L-E_0}^r}$ to $\mathbf{0}$
10:         Extractor outputs scores $u_{s_1^r}, \ldots, u_{s_{L-E_t}^r}$ for remaining sentences and outputs $p_{\text{stop}}$
11:         Compute the action probability $\pi(A_t|S_t, \boldsymbol{\theta})$ according to Equation (4)
12:         $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \frac{r}{T+1} \nabla \log \pi(A_t|S_t, \boldsymbol{\theta})$
```

Algorithm 1 summarizes the training procedure of MemSum. We initialize the extraction history embeddings to $\mathbf{0}$, because at $t = 0$ no sentences have been extracted. E_t represents the number of sentences that have been extracted into the summary up to time step t. Following the strategy in Narayan et al. (2018) and Mohsen et al. (2020), 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 in Appendix E.

4 Experiments

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

Datasets. The documents to be summarized in the PubMed and arXiv datasets (Cohan et al., 2018)

Datasets	avg. doc. length		avg. summ. length		# of doc.-summ. pairs		
	# of words	# of sent.	# of words	# of sent.	Train	Valid	Test
PubMed	2,730	88	181	7	116,937	6,633	6,658
arXiv	5,206	206	238	10	202,880	6,436	6,440
PubMed$_{\text{trunc}}$	408	13	185	7	83,233	4,676	5,025
GovReport	7,932	307	501	18	17,517	974	973
CNN/DM	692	35	49	4	-	-	-

Table 1: An overview of datasets used in this paper. We count only strings composed of letters and numbers for # of words.

are the full bodies of scientific papers and the gold summaries are the corresponding abstracts. Zhong et al. (2020) proposed a truncated version of the PubMed dataset (PubMed$_{\text{trunc}}$ for simplicity) by defining a doument as the introduction section of a paper. The GovReport dataset (Huang et al., 2021) contains U.S. government reports with gold summaries written by experts. Except PubMed$_{\text{trunc}}$, all the other datasets contain significantly longer documents than the popular dataset CNN/DM (Table 1).

Baselines. Extractive baselines include Lead (directly using the first several sentences as the summary) (Gidiotis and Tsoumakas, 2020), SummaRuNNer (Nallapati et al., 2017), Atten-Cont (Xiao and Carenini, 2019), Sent-CLF and Sent-PTR (Pilault et al., 2020), MatchSum (Zhong et al., 2020), and the NeuSum model (Zhou et al., 2018) that we trained on our datasets.

Abstractive summarization models include PEGASUS (Zhang et al., 2020), BigBird (Zaheer et al., 2020), Dancer (Gidiotis and Tsoumakas, 2020), and Hepos (Huang et al., 2021) that achieved the state-of-the-art in long document summarization using a large-scale pretrained BART model (Lewis et al., 2020) with memory-efficient attention encoding schemes including Locality Sensitive Hashing (Kitaev et al., 2020) (Hepos-LSH) and Sinkhorn attention (Hepos-Sinkhorn). We also present the performance of the oracle extraction model based on the greedy approach (Nallapati et al., 2017) which sequentially selects from the document the sentence that maximally improves the average of R-1 and R-2 of selected sentences.

Implementation Details. We computed local sentence embeddings using pretrained Glove word embeddings (Pennington et al., 2014) 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.

We trained our model using the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$ (Kingma and Ba, 2015), fixed learning rate $\alpha = 1e^{-4}$, and weight decay $1e^{-6}$. The training was stopped when the validation performance started to degrade. 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 eight RTX 2080 Ti GPUs.

On the validating datasets we selected the best checkpoint of each model and determined the optimal N_{max} and stopping criterion p^*_{thres}. For Pubmed, arXiv, Pubmed$_{\text{trunc}}$, and GovReport, N_{max} was set to 7, 5, 7, and 22, and p^*_{thres} was set to 0.6, 0.5, 0.8, and 0.6, respectively. For the detailed selection procedure of the optimal stopping threshold, see Appendix D. Information on reproducibility is available in Appendix I.

Evaluation. We evaluated the performance of our model using F_1 ROUGE (Lin, 2004), including ROUGE-1,2, and L for measuring unigram, bigram, and longest common subsequence. We also conducted human evaluation in Section 5.4.

5 Results and Discussion

Here we present the results on various extractive summarization tasks and analyze the contribution of different modules via ablation studies.

5.1 Results Comparison

By comparing with extractive baselines on the PubMed and arXiv datasets, we observed that models utilizing extraction history, such as NeuSum and our MemSum, perform significantly better than other models, revealing the effectiveness of the extraction history. MemSum also significantly outperformed NeuSum, suggesting a *better utilization of extraction history*, which we ascribed to the following factors: 1) In MemSum, we treat stopping extraction also as an action and train the policy network to output a stopping probability. There-

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 baselines						
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
NeuSum	47.46	21.92	42.87	47.49	21.56	41.58
Abstractive summarization baselines						
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
MemSum (ours)	**49.25***	**22.94***	**44.42***	**48.42**	**20.30**	**42.54***

Table 2: Results on the PubMed and arXiv test sets. "*" indicates that they are statistically significant in comparison to the closest baseline with a 95% bootstrap confidence interval estimated by the ROUGE script[2].

Model	PubMed$_{\text{trunc}}$			GovReport		
	R-1	R-2	R-L	R-1	R-2	R-L
ORACLE	45.12	20.33	40.19	75.56	45.91	72.51
Extractive summarization baselines						
Lead	37.58	12.22	33.44	50.94	19.53	48.45
MatchSum	41.21	14.91	36.75	-	-	-
NeuSum	-	-	-	58.94	25.38	55.80
Abstractive summarization baselines						
Hepos-LSH	-	-	-	55.00	21.13	51.67
Hepos-Sinkhorn	-	-	-	56.86	22.62	53.82
MemSum (ours)	**43.08***	**16.71***	**38.30***	**59.43***	**28.60***	**56.69***

Table 3: Results on PubMed$_{\text{trunc}}$ and GovReport.

fore, MemSum is able to automatically stop extracting at an optimal time step based on extraction history, while NeuSum can only extract a predefined number of sentences; 2) With the policy gradient method REINFORCE we can train MemSum to maximize the ROUGE score directly, while in NeuSum the loss was set to the KL-divergence between the model-computed sentence scores and the ROUGE score gains at each step, which is less intuitive. We further compare MemSum with NeuSum via human evaluation in Section 5.4.

We observed that the ROUGE performance on the PubMed$_{\text{trunc}}$ dataset is significantly lower than that on the PubMed dataset, with a 16.87 drop in R-1 for the extractive oracle and a 6.23 drop in R-1 for MemSum, *indicating that the introduction section is not sufficient to generate summaries close to the ground truth (abstracts)*. Even so, our model still significantly outperformed MatchSum on PubMed$_{\text{trunc}}$, and we attribute this improvement to the fact that MatchSum truncates the introduction section further to 512 tokens because it needs to compute document embeddings using Bert. Consequently, MatchSum extracts sentences mainly from the first 15 sentences of the document, while our MemSum produces a similar distribution of extracted sentence positions as the extractive oracle, Figure 3. Thus, *summarizing long documents is a non-trivial task*, and models that work well on summarizing short documents (e.g., CNN/DM) may fail to generalize to long documents.

MemSum also significantly outperformed the state-of-the-art abstractive summarization model Hepos as measured by ROUGE scores, especially on the GovReport dataset. A comparison of an exemplary MemSum-extracted summary and the corresponding Hepos-Sinkhorn-generated summary from the GovReport dataset (Table 4) is consistent with the ROUGE comparison, showing that the MemSum-extracted summary is more accurate

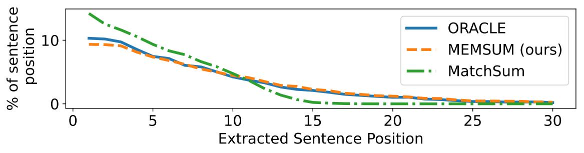


Figure 3: The position distribution of extracted sentences in the PubMed$_{\text{trunc}}$ dataset.

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, (...) 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): (...) 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: Comparison of the summary extracted by MemSum and the summary abstractively generated by Hepos-Sinkhorn (Huang et al., 2021). Compared with the abstractive summary, the MemSum-extracted summary has higher overlap with the human-written summary.

[2]https://pypi.org/project/rouge-score/

Model	R-1	R-2	R-L
MemSum	**49.25**	**22.94**	**44.42**
MemSum w/o LSE	48.12	22.04	43.36
MemSum w/o GCE	46.85	20.31	41.95
MemSum w/o EHE	48.08	22.77	43.55
MemSum with GRU-EHE	49.11	22.86	44.28
MemSum w/o auto-stop	48.25	22.63	43.70
MemSum with "STOP"	47.18	21.81	42.20

Table 5: Ablation study on the PubMed dataset.

Model	R-1	R-2	R-L	duplicate percentage
MemSum	**49.16**	**22.78**	**44.39**	0%
MemSum w/o auto-stop	48.21	22.59	43.76	0%
MemSum w/o EHE	42.82	18.18	36.68	41%
MemSum w/o EHE +3gram blocking	46.85	19.93	42.40	0%

Table 6: Performance on the redundant PubMed dataset.

than the Hepos-Sinkhorn-generated summary and has higher overlap with the gold summary. We deem that this particularly good extraction performance on the GovReport dataset results from the higher "extractiveness" of the gold summaries in the GovReport dataset compared to other datasets, which may be due in part to technical language being difficult to abstractively summarize without a change in meaning. This is evidenced by the fact that the ROUGE scores of the extractive oracle on the GovReport dataset (Table 3) are higher than those of the PubMed and arXiv datasets (Table 2). Therefore, *extractive summarization may be more proper than abstractive summarization due to the requirement of stringent faithfulness of government report summaries*.

5.2 Ablation Test

We conduct ablation studies by comparing the full MemSum model with the following *variations in structures*: 1) MemSum w/o LSE, where we obtain local sentence embeddings by replacing the bi-LSTM based LSE by simple averages of word embeddings; 2) MemSum w/o GCE where we remove the GCE; 3) MemSum w/o EHE where we remove EHE, compute the scores for all sentences in one step, and samples sentences following the BanditSum policy (Dong et al., 2018); 4) MemSum with GRU-EHE where we use a GRU to encode previously extracted sentences at each time step, and uses the last hidden state as the extraction history embedding for all remaining sentences, following Zhou et al. (2018).

Meanwhile, we also tested two variations that adopted *different stopping mechanisms*: 1) MemSum w/o auto-stop that does not stop extraction automatically based on p_{stop}, but that extracts a fixed number of sentences; 2) MemSum with "STOP" that inserts a special stop sentence (e.g. "STOP") into the document, and stops extraction once the agent selects this sentence.

Contribution of Modules. Removing GCE has a greater impact on performance than removing LSE (Table 5), suggesting that modeling global contextual information is more critical than modeling local sentence information in our MemSum framework, which contrasts with the result that modeling local sentence information is more important in the Atten-Cont (Xiao and Carenini, 2019) framework. Furthermore, we observed a significant performance degradation when removing EHE, but no significant difference between MemSum and MemSum with GRU-EHE, indicating that EHE is necessary, but our *MemSum policy is not strongly dependent on the specific structure of this module* (e.g., attention-based or RNN-based).

Influence of Stopping Mechanisms. MemSum w/o auto-stop achieves lower ROUGE scores than MemSum, revealing the necessity of auto stopping in our MemSum architecture. Meanwhile, MemSum with "STOP" produced summaries with fewer extracted sentences (3.9 vs. 6.0 sentences on average) and significantly lower ROUGE scores. We attribute this reduction to the predictable positive reward obtained from selecting the special stop sentence that ends an episode, which leads to a preference for this final action and increases the likelihood of taking this action prematurely.

5.3 History Awareness Avoids Redundancy

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 dataset, we trained and tested MemSum and MemSum w/o EHE (no history awareness), and we compared different models in terms of ROUGE scores and average *duplicate percentage* that is defined as the average percentage of the duplicated sentences among all extracted sentences in a summary.

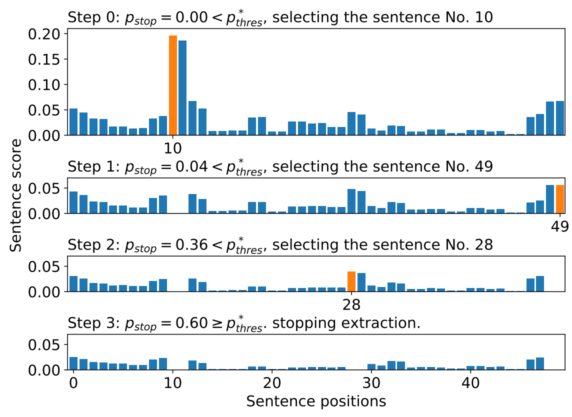


Figure 4: The sentence scores of 50 sentences computed by MemSum at extraction steps 0 to 3. In the document, there is artificial redundancy in that the $(2n)_{\text{th}}$ and the $(2n+1)_{\text{th}}$ sentences are identical ($n = 0, 1, ..., 24$).

As reported in Table 6, for MemSum w/o EHE, on average 41% of sentences in the extracted summaries were duplicated. Along with the high duplicate ratio came a significant decrease in ROUGE score. By contrast, the performance of the full MemSum model with history awareness was only slighted affected when comparing the results of the MemSum on the PubMed dataset (Table 2) and on the redundant PubMed dataset (Table 6).

Meanwhile, using the Trigram Blocking method that skips a sentence if it has a trigram that overlaps with the current summary (Liu and Lapata, 2019b) is also successful in avoiding repetitive sentences. However, the ROUGE scores associated with Trigram Blocking were significantly lower than those of the 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 or crude rules, and thus shows better performance.

Case Study: How does MemSum Avoid Redundancy?

We let MemSum 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 4. 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 and their following sentences, revealing competent

Criteria	Experiment I		Experiment II	
	NeuSum	MemSum	NeuSum	MemSum w/o auto-stop
overall	1.58	**1.37**	1.57	**1.38**
coverage	**1.46**	1.49	**1.44**	1.51
non-redundancy	1.67	**1.28***	1.65	**1.30***
avg. summ. length				
# of sentences	7.0	**5.6***	7.0	7.0
# of words	248.8	**189.3***	263.6	**239.5***

Table 7: The average ranking of NeuSum and MemSum is reported. The smaller the ranking, the better the model. Four volunteers participated in these experiments, and evaluated 67 and 63 pairs of summaries in Experiment 1 and 2, respectively. "*" indicates statistical significance (p<0.005) in a Wilcoxon signed-rank test (Woolson, 2008).

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" selected sentences' positions.

5.4 Human Evaluation

We conducted human evaluation following Wu and Hu (2018); Dong et al. (2018); Luo et al. (2019). For each document sampled from the test set of the PubMed dataset, we provide a reference summary, and volunteers are asked to rank a pair of randomly ordered summaries produced by two models according to three criteria: non-redundancy, coverage, and overall quality. The better model will be ranked #1 while the other is ranked #2, and if both models extract the same summary, then they will both get the #1 rank. In experiment 1, we compared NeuSum, which always extracts 7 sentences, and MemSum, which extracts a flexible number of sentences thanks to automatic stopping. In experiment 2, we discounted for differences in the number of extracted sentences by making MemSum w/o autostop to also extract 7 sentences. A user-friendly interactive web interface was implemented to assist the evaluation process, with details in Appendix G.

Table 7 reports the human evaluation results for both experiments. Both MemSum and MemSum w/o auto-stop ranked significantly higher (p<0.005) than NeuSum in terms of non-redundancy and achieved a better average overall quality. In terms of word count, MemSum produces shorter summaries than NeuSum in both experiments, even though both models extract the same number of sentences in experiment 2. These results show that redundancy avoidance of MemSum is particularly good, even without the auto-stop mechanism. The slightly better performance of NeuSum in terms of coverage needs to be weighed against it extracting significantly longer summaries. Note that neither NeuSum nor our model is trained to optimize the order of the extracted sentences. Therefore, we did not use fluency, which depends on sentence order, as a metric for human evaluation. Improving the fluency of the extracted summaries will be the subject of our future research.

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, MemSum is given 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 be robust against redundancy in the document. As a lightweight model (Appendix C), MemSum outperforms both extractive and abstractive baselines on diverse long document summarization tasks. Because MemSum achieves SOTA performance on these tasks, MDP approaches will be promising design choices for further research.

Acknowledgements

We acknowledge support from the Swiss National Science Foundation (grant 31003A_182638) and the NCCR Evolving Language, Swiss National Science Foundation Agreement No. 51NF40_180888. We also thank the anonymous reviewers for their useful comments.

References

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. 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. Association for Computational Linguistics.

Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. 2018. 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. Association for Computational Linguistics.

Markus Freitag and Yaser Al-Onaizan. 2017. Beam search strategies for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 56–60, Vancouver. Association for Computational Linguistics.

Mahak Gambhir and Vishal Gupta. 2017. Recent automatic text summarization techniques: A survey. *Artif. Intell. Rev.*, 47(1):1–66.

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

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. 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. MIT Press.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1419–1436, Online. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. 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. Association for Computational Linguistics.

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

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Peter J. Liu*, Mohammad Saleh*, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*.

Yang Liu and Mirella Lapata. 2019a. Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081, Florence, Italy. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019b. 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. Association for Computational Linguistics.

Ling Luo, Xiang Ao, Yan Song, Feiyang Pan, Min Yang, and Qing He. 2019. 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.

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

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. 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.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. 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. Association for Computational Linguistics.

Ani Nenkova and Kathleen McKeown. 2012. A Survey of Text Summarization Techniques. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 43–76. Springer US, Boston, MA.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. In *NAACL 2018 - Conference of the North American Chapter of the Association for Computational Linguistics*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. 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. Association for Computational Linguistics.

Jonathan Pilault, Raymond Li, Sandeep Subramanian, and Chris Pal. 2020. 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. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27, pages 3104–3112. Curran Associates, Inc.

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

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256.

R. F. Woolson. 2008. *Wilcoxon Signed-Rank Test*, pages 1–3. John Wiley & Sons, Ltd.

Yuxiang Wu and Baotian Hu. 2018. Learning to extract coherent summary via deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

Wen Xiao and Giuseppe Carenini. 2019. 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. Association for Computational Linguistics.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural*

Information Processing Systems, volume 33, pages 17283–17297. Curran Associates, Inc.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR.

Xingxing Zhang, Furu Wei, and Ming Zhou. 2019. 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. Association for Computational Linguistics.

Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. Extractive summarization as text matching. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6197–6208, Online. Association for Computational Linguistics.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–663, Melbourne, Australia. Association for Computational Linguistics.

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: PubMed (Cohan et al., 2018), arXiv (Cohan et al., 2018), and GovReport (Huang et al., 2021). The results are reported 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 must be extracted, the more actions are needed of selecting sentences within an episode.

Datasets	Validating			Test		
	R-1	R-2	R-L	R-1	R-2	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
PubMed$_{\text{trunc}}$	43.46	16.77	38.65	43.08	16.71	38.30
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.

Datasets	avg. doc. length (words)	Avg. extractive summ. length (# sentences)	Avg. extractive summ. time (ms)
PubMed	2,730	6.0 ± 1.2	91.7 ± 8.6
arXiv	5,206	4.8± 0.5	114.0 ± 5.0
PubMed$_{\text{trunc}}$	408	5.3± 1.4	27.7 ± 4.6
GovReport	7,932	21.7 ± 1.8	197.0 ± 14.8

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

D Selection of optimal stopping threshold

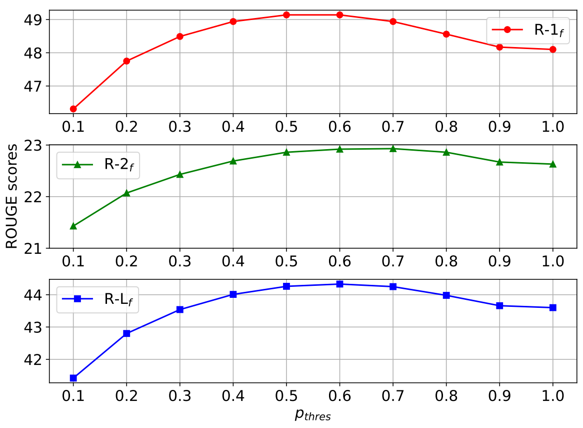


Figure 5: The ROUGE scores for different stopping thresholds p_{thres} on the PubMed validating set.

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

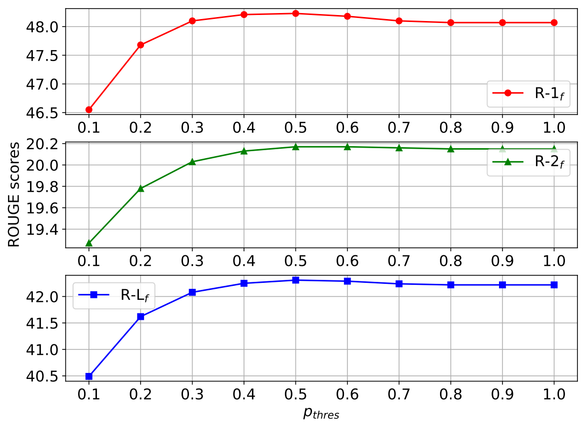


Figure 6: The ROUGE scores for different stopping thresholds p_{thres} on the arXiv validating set.

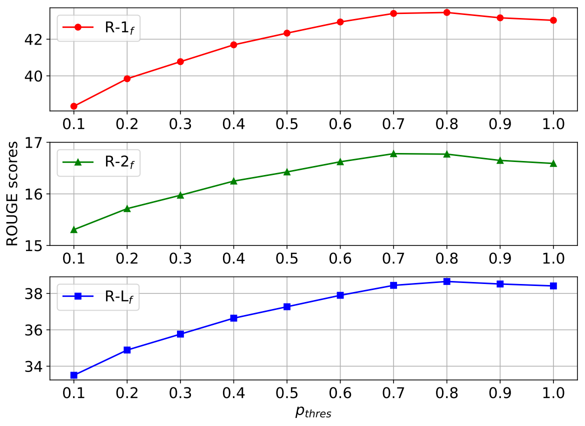


Figure 7: The ROUGE scores for different stopping thresholds p_{thres} on the PubMed$_{\text{trunc}}$ validating set.

threshold are shown in Figure 5, 6 and 8 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 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 $p^*_{\text{thres}} = 0.6,\ 0.5,\ 0.8$ and 0.6 for the PubMed, the arXiv, the PubMed$_{\text{trunc}}$, 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 $(S_0,s_{a_0},\dots,S_{T-1},s_{a_{T-1}},\ S_T,A_{\text{stop}},r)$. Here, $\{s_{a_0}\dots s_{a_{T-1}}\}$ is the extracted summary consist-

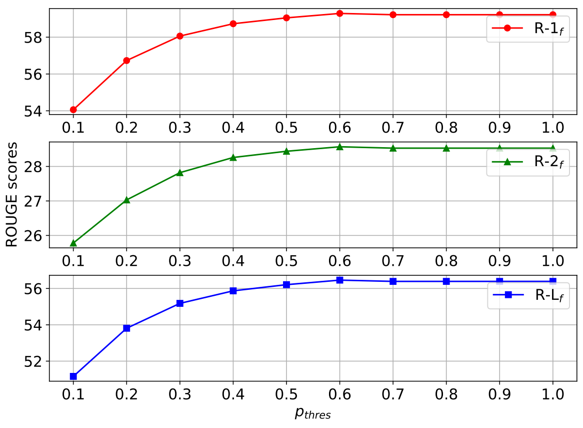


Figure 8: The ROUGE scores for different stopping thresholds p_{thres} on the GovReport validating set.

ing of a set of T sentences, and r is the average of the associated ROUGE-1, ROUGE-2, and ROUGE-L F1 scores.

In (Nallapati et al., 2017), 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.

In this paper, we define a high-ROUGE episodes set $\mathbb{E}_p$ as the set of multiple episodes where each episode has a high average ROUGE-1/2/L F1 score. To obtain not a single episode in $\mathbb{E}_p$ but multiple episodes with high average ROUGE-1/2 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 the branching size, analogous to beam search strategies in neural machine translation (Sutskever et al., 2014; Freitag and Al-Onaizan, 2017). We set $B = 2$ by default.

In practice, we notice that ROUGE-L F1 score is computationally intensive. Because when creating $\mathbb{E}_p$ we need to iteratively re-compute ROUGE scores once a new sentence is added to the current summary, including the ROUGE-L F1 score into computation would heavily slow down the process of creating the high-ROUGE episodes set for training. As a compromise, we do not incorporate the

ROUGE-L F1 score into the intermediate steps of our modified greedy approach. Instead, we calculate the ROUGE-L F1 score only once after a complete high-ROUGE episode is selected, and use this ROUGE-L F1 score together with ROUGE-1/2 F1 scores to compute the reward r for each episode. A similar strategy was adopted in Zhou et al. (2018) to create the training dataset by maximizing ROUGE-2 F1 scores only.

We refer to an episode (S_0,s_A,S_1,s_B,S_2,s_C,S_3,A_{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 nearly the same average ROUGE-1/2 scores (although ROUGE-L score may differ), we decided to equally sample them with the hope 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 Padding and Truncation of Sentences and Documents

In the training process, we used mini-batch gradient descent. To enable efficient batch-wise parallel GPU computation, each document in a mini batch needs to have the same number of sentences, and each sentence needs to have the same number of tokens. Therefore, in order to unify the sentence length to a common value L_{sen}, we appended "PAD" tokens at the end of sentences shorter than L_{sen}, and we truncated sentences longer than L_{sen}. To unify the document length in terms of number of sentences to a common value L_{doc}, we appended empty-string sentences at the end of documents shorter than L_{doc}, and truncated documents longer than L_{doc}. To ensure consistency between training and testing we also performed the same padding and truncation setting during testing. We set L_{doc} to 500 for the PubMed, the arXiv, and the GovReport datasets and 50 for the PubMed$_{\text{trunc}}$ dataset based on the document length statistics shown in Table 1 in the main paper. We set L_{sen} to 100 for the PubMed, the PubMed$_{\text{trunc}}$, and the GovReport datasets and 150 for the arXiv dataset, because we noticed a larger variance in the length of sentences in the arXiv dataset.

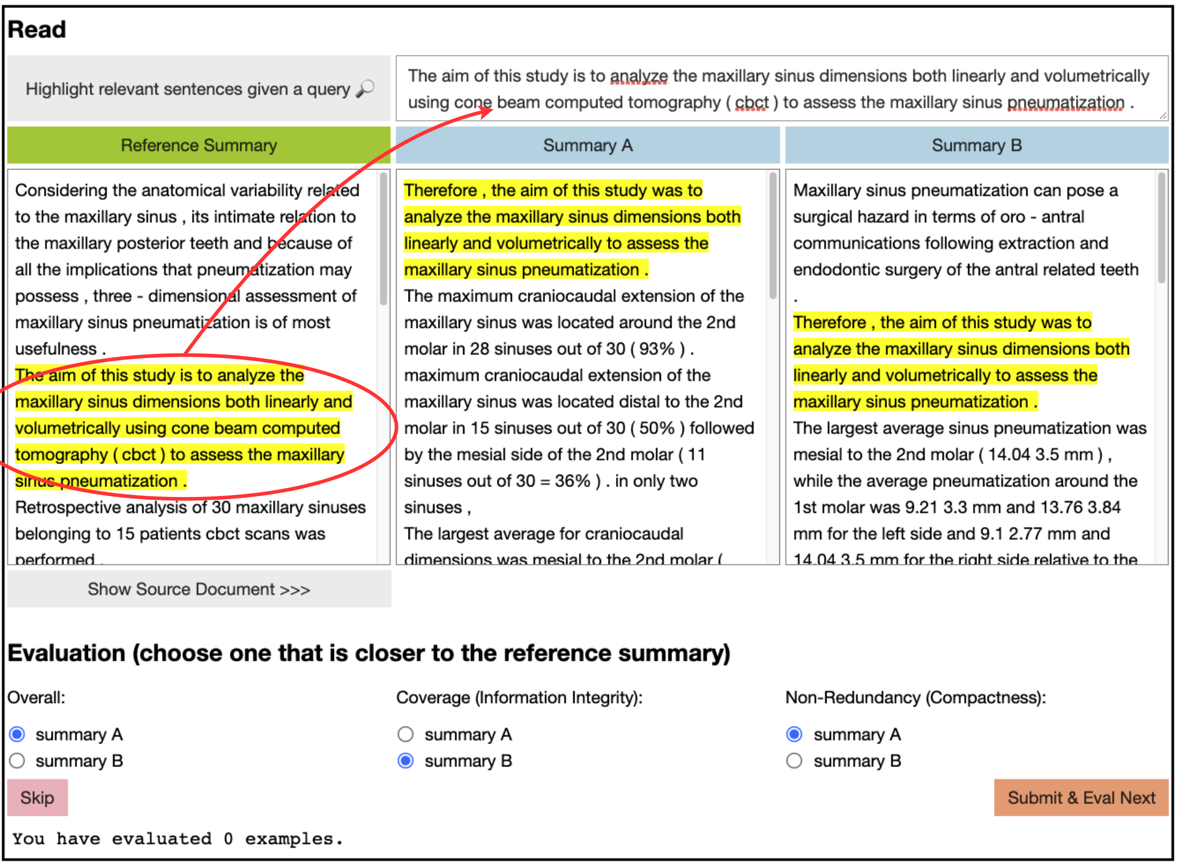
G Interactive Web Interface for Human Evaluation

To provide for a convenient evaluation procedure for volunteers, we designed an interactive web interface based on Jupyter Widgets[3]. As shown in Figure 9, for each document, we display the reference summary, summary A, and summary B from left to right. The reference summary contains the ground-truth abstract. Summaries A and B are the summaries extracted by the two models assigned in a random order, so that the volunteers do not know which model either summary came from. Meanwhile, the volunteers were allowed to read the source document by clicking the button "Show Source Document >>>". We also provided a sentence highlighting function to help the volunteers rapidly retrieve relevant content. We allowed evaluators to copy a sentence from the reference summary and paste it to the text box above. After clicking the button "Highlight relevant sentences given a query", relevant sentences in both summaries were highlighted, to help the volunteers rapidly find information of interest. The relevance score of a pair of sentences was given by the cosine similarity of the two sentences' embeddings computed with Sent2vec (Pagliardini et al., 2018). In the evaluation panel the volunteers selected the better summary (A or B) by comparing the model-produced summary with the reference summary on three criteria: overall quality, coverage (in terms of information content), and non-redundancy. After making a choice they clicked the button "Submit & Eval Next" to submit the current evaluation result and evaluate the next summaries, or click "Skip" if they were not sure which summary was indeed better.

H Examples of Extracted Summaries

We provide summarization examples in Table 10 and 11. In Table 10, we compared MemSum trained on the arXiv dataset with Dancer Pegasus (Gidiotis and Tsoumakas, 2020) on a typical paper on which MemSum achieved higher ROUGE-1 F score than Dancer Pegasus. In Table 11 we provide the extractive summary of this paper itself using our MemSum model. Sentences with similar meanings in different summaries are highlighted in the same color.

[3] `https://ipywidgets.readthedocs.io/`


Read

Highlight relevant sentences given a query

The aim of this study is to analyze the maxillary sinus dimensions both linearly and volumetrically using cone beam computed tomography (cbct) to assess the maxillary sinus pneumatization .

Reference Summary

Considering the anatomical variability related to the maxillary sinus , its intimate relation to the maxillary posterior teeth and because of all the implications that pneumatization may possess , three - dimensional assessment of maxillary sinus pneumatization is of most usefulness . The aim of this study is to analyze the maxillary sinus dimensions both linearly and volumetrically using cone beam computed tomography (cbct) to assess the maxillary sinus pneumatization . Retrospective analysis of 30 maxillary sinuses belonging to 15 patients cbct scans was performed .

Summary A

Therefore , the aim of this study was to analyze the maxillary sinus dimensions both linearly and volumetrically to assess the maxillary sinus pneumatization . The maximum craniocaudal extension of the maxillary sinus was located around the 2nd molar in 28 sinuses out of 30 (93%) . maximum craniocaudal extension of the maxillary sinus was located distal to the 2nd molar in 15 sinuses out of 30 (50%) followed by the mesial side of the 2nd molar (11 sinuses out of 30 = 36%) . in only two sinuses , The largest average for craniocaudal dimensions was mesial to the 2nd molar (

Summary B

Maxillary sinus pneumatization can pose a surgical hazard in terms of oro - antral communications following extraction and endodontic surgery of the antral related teeth . Therefore , the aim of this study was to analyze the maxillary sinus dimensions both linearly and volumetrically to assess the maxillary sinus pneumatization . The largest average sinus pneumatization was mesial to the 2nd molar (14.04 3.5 mm) , while the average pneumatization around the 1st molar was 9.21 3.3 mm and 13.76 3.84 mm for the left side and 9.1 2.77 mm and 14.04 3.5 mm for the right side relative to the

Show Source Document >>>

Evaluation (choose one that is closer to the reference summary)

Overall: [✓ summary A] summary B

Coverage (Information Integrity): summary A [✓ summary B]

Non-Redundancy (Compactness): [✓ summary A] summary B

Skip

Submit & Eval Next

```
You have evaluated 0 examples.
```

Figure 9: We designed an interactive web interface for the human evaluation experiments introduced in Section 5.4.

I Reproducibility

The MemSum code and variants of MemSum that we used in our ablation study, as well as the MemSum parameters trained on the PubMed dataset, can be found in the submitted code.zip file. Also, we provide a sample of the datasets used in this paper in the data.zip file, as well as the raw data for the human evaluation. This will ensure that the results in this work are well reproducible.

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	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	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 (Gidiotis and Tsoumakas, 2020) 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 each step with information on the current extraction history. When MemSum iteratively selects sentences into the summary, it considers a broad information set 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 obtains state-of-the-art test-set performance (ROUGE) in summarizing long documents taken from PubMed, arXiv, and GovReport. Ablation studies demonstrate the importance of local, global, and history information. A human evaluation confirms the high quality and low redundancy of the generated summaries, stemming from MemSum's awareness of extraction history.
MemSum summary	In this paper, we propose to model extractive summarization as a multi-step episodic Markov Decision Process (MDP). As shown in Figure 1, at each time step in an episode, we define a sentence state composed of three sub-states: 1) the local content of the sentence, 2) the global context of the sentence within the document, and 3) information on the extraction history, including the previously selected set of unordered sentences and the remaining sentences. To efficiently encode local and global sentence states, we design an extraction agent based on LSTM networks. We show that extraction-history awareness allows our model to extract more compact summaries than models without history awareness and behave more robustly to redundancies in documents. 3) Our model outperforms both extractive and abstractive summarization models on PubMed, arXiv, and GovReport datasets.
ROUGE1-F1	48.57

Table 11: MemSum summary of this paper.