Paper Reading

Sliding Selector Network with Dynamic Memory for Extractive Summarization of Long Documents

M2 Shogo Fujita

2021/6/17

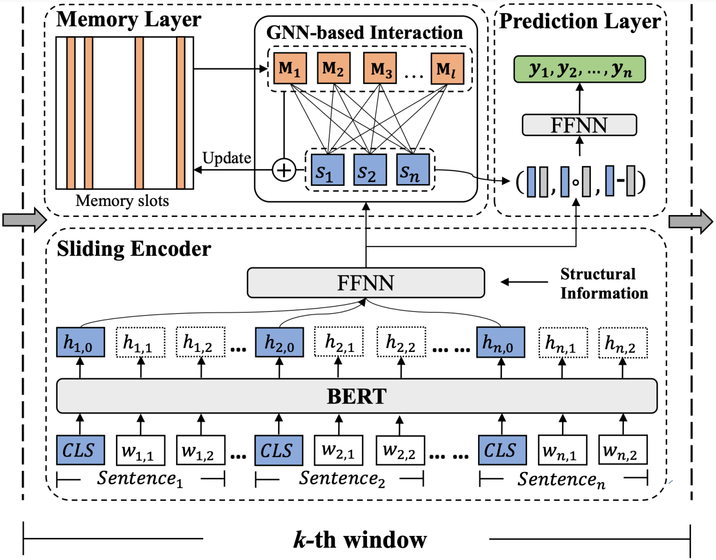
Introduction

Task Extract summary from documents

Contribution

- They propose the sliding selector network with dynamic memory for extractive summarization of long-form documents
- They perform qualitative and quantitative investigations on how our model works and where the performance gain comes from

Overview


Memory Layer
GNN-based Interaction
Prediction Layer
M_1 M_2 M_3 … M_l
Update
Memory slots
s_1 s_2 s_n
$y_1, y_2, \dots, y_n$
FFNN
Sliding Encoder
FFNN
Structural Information
$h_{1,0}$ $h_{1,1}$ $h_{1,2}$ … $h_{2,0}$ $h_{2,1}$ $h_{2,2}$ … … $h_{n,0}$ $h_{n,1}$ $h_{n,2}$
BERT
CLS $w_{1,1}$ $w_{1,2}$ … CLS $w_{2,1}$ $w_{2,2}$ … … CLS $w_{n,1}$ $w_{n,2}$
$Sentence_1$ $Sentence_2$ $Sentence_n$
k-th window

Sliding Encoder

Let $seg^k = s_1^k, s_2^k, ..., s_n^k$ be the k-th window consisting of n sentences.

$$\begin{aligned} O_B &= BERT(w_{1,CLS}^k, ..., w_{1,SEP}^k, ..., w_{n,SEP}^k) \\ &= \{h_{1,CLS}^k, ..., h_{1,SEP}^k, ..., h_{n,SEP}^k\} \qquad (1) \\ H^k &= \{h_{1,CLS}^k, h_{2,CLS}^k, h_{n,CLS}^k\} \qquad (2) \\ \{h^k\} &= tanh(W_1 H^k + W_2 e_w^k + W_3 e_s) \qquad (3) \end{aligned}$$

where e_w^k indicates the k-th window-level position embedding, and e_s is the section embedding. $\{h_k\}$ is the series of representation vectors for sentences in k-th window.

Graph-based Memory Interaction

$M_K = m_1^k, m_2^k, ..., m_l^k$ are memory nodes.

$$z_{i,j}^k = LeaklyRelu(W_a[h_i^k; SG(m_j^k)]) \tag{4}$$

$$\alpha_{i,j} = \frac{exp({z^k}_{i,j})}{\sum_{j'=1}^{l} exp_(^k z_{i,j'})} \tag{5}$$

$$\tilde{h}_i^k = \|_{t=1}^{T} \sum_{j=1}^{l} tanh(\alpha_i^t, W_c^t SG(m_j^k)) \tag{6}$$

Function SG stands for stop-gradient operation.

Prediction Layer

$$\tilde{y}_i = f_o(\tilde{h}_i^k, h_i^k, \|\tilde{h}_i^k - h_i^k\|, \tilde{h}_i^k \odot h_i^k) \quad (7)$$

$$L = -\sum\{y_i log(\tilde{y}_i) + (1 - y_i) log(1 - \tilde{y}_i)\} \quad (8)$$

$\tilde{y}_i$ represents the predicted probability of i-th sentence. they rank all the sentences and select top-k as the final summary.

Dynamic Memory Updating

$$\sigma_i^k = tanh(W_m * \tilde{m}_i^k) \quad (9)$$

$$u_i^k = \sigma_i^k m_i^k + (1 - \sigma_i^k) \odot \tilde{m}_i^k \quad (10)$$

$$r_{sum}^k = \sum_{i=1}^{n} \tilde{y}_i * h_i^k \quad (11)$$

$$m_i^{k+1} = tanh(W_4 m_i^k + W_5 r_{sum}^k) \quad (12)$$

$$(maybe \quad m_i^{k+1} = tanh(W_4 u_i^k + W_5 r_{sum}^k)) \quad (13)$$

$\sigma_i^k \in R^{dm}$ is an gate vector.

Experiment

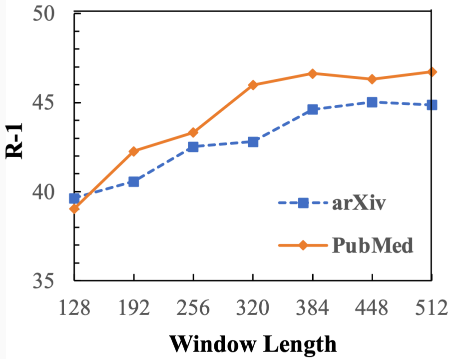
Models	arXiv			PubMed		
	R-1	R-2	R-L	R-1	R-2	R-L
Lead	33.66	8.94	22.19	35.63	12.28	25.17
LexRank+	33.85	10.73	28.99	39.19	13.89	34.59
LSA+	29.91	7.42	25.67	33.89	9.93	29.70
Oracle*	53.88	23.05	34.90	55.05	27.48	38.66
Seq2seq-attention+	29.30	6.00	25.56	31.55	8.52	27.38
PGN+	32.06	9.04	25.16	35.86	10.22	29.69
Disourse-aware+	35.80	11.05	31.80	38.93	15.37	35.21
Cheng & Lapta (2016)*	42.24	15.97	27.88	43.89	18.53	30.17
SummaRuNNer*	42.81	16.52	28.23	43.89	18.78	30.36
Seq2seq-local&global*	43.62	17.36	29.14	44.85	19.70	31.43
Match-Sum	40.59	12.98	32.64	41.21	14.91	**36.75**
Topic-GraphSum	44.03	18.52	32.41	45.95	20.81	33.97
SSN-DM	**45.03**	19.03	32.58	**46.73**	**21.00**	34.10
SSN-DM + discourse	44.90	**19.06**	**32.77**	46.52	20.94	35.20

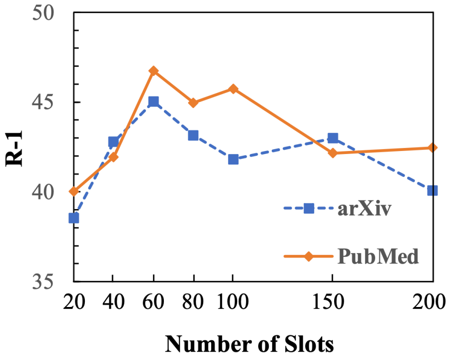
Varying iteration numbers t of GAT

Iteration Numbers	Rouge-1	
	arXiv	PubMed
$t = 0$	44.79	46.42
$t = 1$	44.95	46.69
$t = 2$	**45.03**	46.73
$t = 3$	44.97	**46.74**
$t = 4$	45.01	46.71

To select the best iteration number (hop number) t, we compare the performance of different t on the validation sets of two datasets.

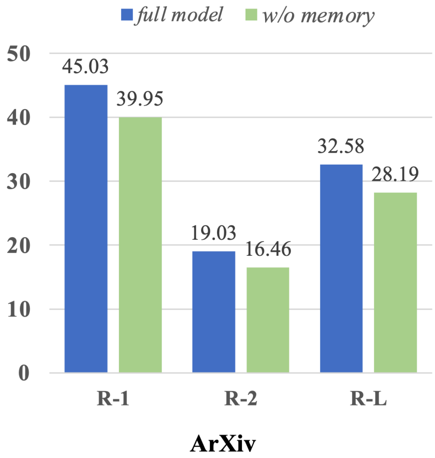
Results on Varying Hyperparameters

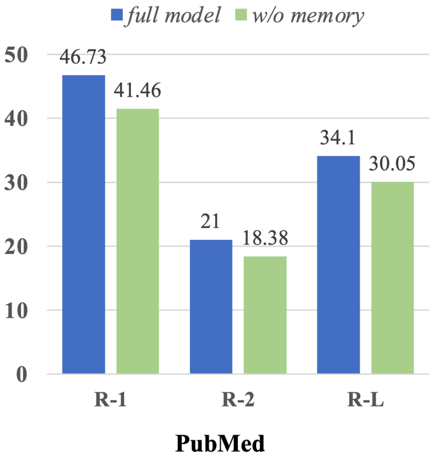




Left figure presents the Rouge-1 results on varying window length. Right figure presents the Rouge-1 results on varying slot numbers / in memory.

Effect of Dynamic Memory





Effect of Dynamic Memory

Models	S_{Rep}	S_{Noise}	S_{Len}
	arXiv		
w/o memory	0.105	0.118	1.247
Full model	0.033	0.011	0.295
	PubMed		
w/o memory	0.107	0.097	1.106
Full model	0.031	0.008	0.343

$S_{rep} = 1 - \frac{CountUniq(ngram)}{Count(ngram)}$, $S_{Noise} = \frac{Count(NoisySent)}{Count(ExtractSent)}$,
$S_{Len} = \frac{|sum| - |ref|}{|ref|}$

Conclusion

- They propose a novel extractive summarization that can summarize long-form documents without content loss
- They have validated the effectiveness of the proposed mechanism