Hierarchical Transformer for multi document summarization

Jefferson Hsieh

Summarization

- Single Document vs Multi Document
- Abstractive vs Extractive

Wikisumm – abstractive MDS dataset+model

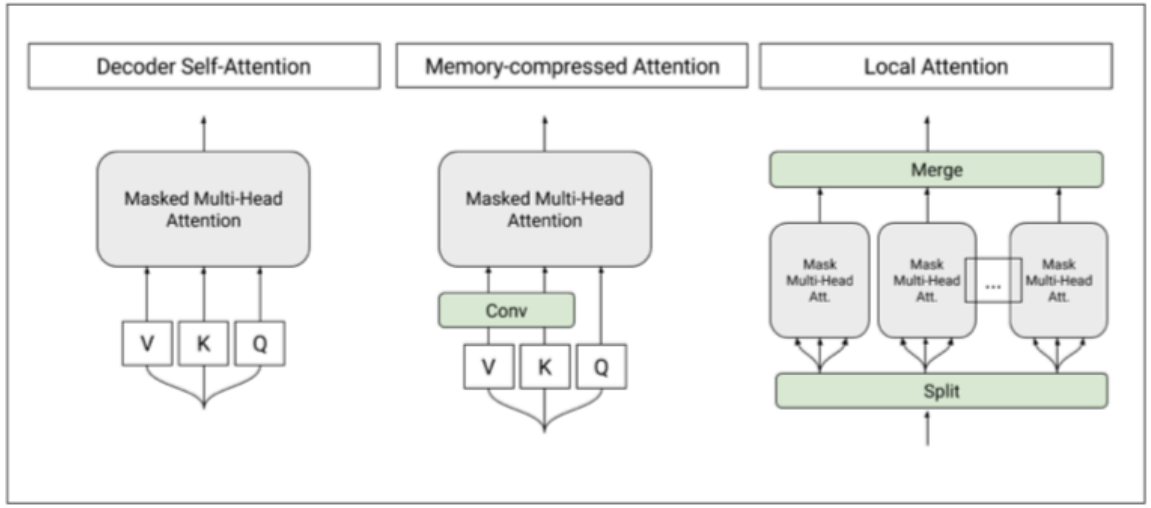

Figure 1: The architecture of the self-attention layers used in the T-DMCA model. Every attention layer takes a sequence of tokens as input and produces a sequence of similar length as the output. **Left:** Original self-attention as used in the transformer-decoder. **Middle:** Memory-compressed attention which reduce the number of keys/values. **Right:** Local attention which splits the sequence into individual smaller sub-sequences. The sub-sequences are then merged together to get the final output sequence.

Hiersumm(Yang Liu, Mirella Lapata 2019)

- Abstractive
- Trained on Wikisumm dataset

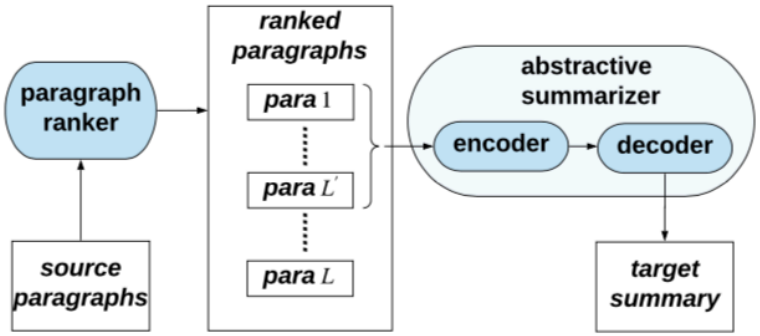

Figure 1: Pipeline of our multi-document summarization system. L source paragraphs are first ranked and the L'-best ones serve as input to an encoder-decoder model which generates the target summary.

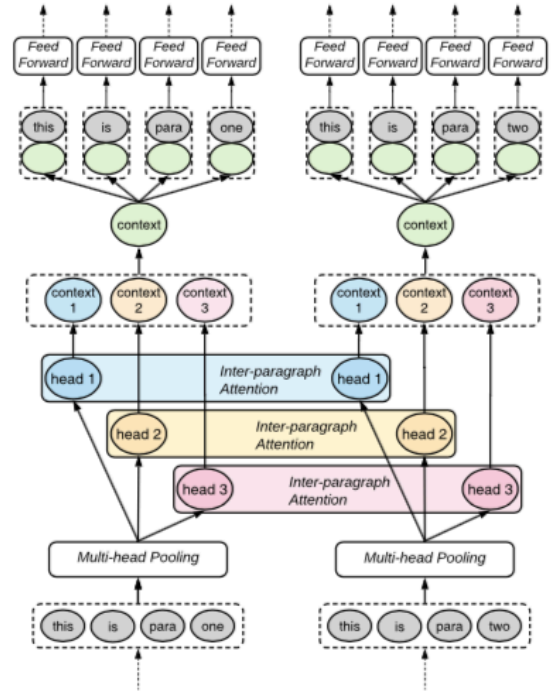

Figure 2: A global transformer layer. Different colors indicate different heads in multi-head pooling and inter-paragraph attention.

Reproducing results

Model	ROUGE-1	ROUGE-2	ROUGE-L
Lead	38.22	16.85	26.89
LexRank	36.12	11.67	22.52
FT (600 tokens, no ranking)	35.46	20.26	30.65
FT (600 tokens)	40.46	25.26	34.65
FT (800 tokens)	40.56	25.35	34.73
FT (1,200 tokens)	39.55	24.63	33.99
T-DMCA (3000 tokens)	40.77	25.60	34.90
HT (1,600 tokens)	**40.82**	**25.99**	35.08
HT (1,600 tokens) + Similarity Graph	40.80	25.95	35.08
HT (1,600 tokens) + Discourse Graph	40.81	25.95	**35.24**
HT (train on 1,600 tokens/test on 3000 tokens)	**41.53**	**26.52**	**35.76**

Table 2: Test set results on the WikiSum dataset using ROUGE F_1.

---

1 ROUGE-1 Average_R: 0.37926 (95%-conf.int. 0.37708 - 0.38151)
1 ROUGE-1 Average_P: 0.61810 (95%-conf.int. 0.61556 - 0.62057)
1 ROUGE-1 Average_F: 0.41340 (95%-conf.int. 0.41144 - 0.41534)

---

1 ROUGE-2 Average_R: 0.23940 (95%-conf.int. 0.23721 - 0.24160)
1 ROUGE-2 Average_P: 0.39258 (95%-conf.int. 0.38964 - 0.39558)
1 ROUGE-2 Average_F: 0.26367 (95%-conf.int. 0.26151 - 0.26569)

---

1 ROUGE-L Average_R: 0.32521 (95%-conf.int. 0.32304 - 0.32741)
1 ROUGE-L Average_P: 0.52971 (95%-conf.int. 0.52702 - 0.53237)
1 ROUGE-L Average_F: 0.35556 (95%-conf.int. 0.35353 - 0.35759)

Our modifications

- Product Key Memory layer (Guillaume Lample , Alexandre Sablayrolles, Marc’Aurelio Ranzato, Ludovic Denoyer , Hervé Jégou 2019)

- Extractive Hiersumm – take the pretrained encoder part of Hiersumm

Product Key Memory

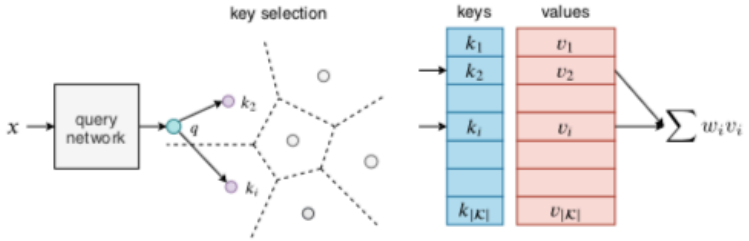

Figure 1: **Overview of a key-value memory layer:** The input x is processed thro network that produces a query vector q, which is compared to all the keys. The output weighted sum over the memories associated with the selected keys. For a large num $|\mathcal{K}|$, the key selection procedure becomes too expensive in practice. Our product ke exact and makes this search process very fast.

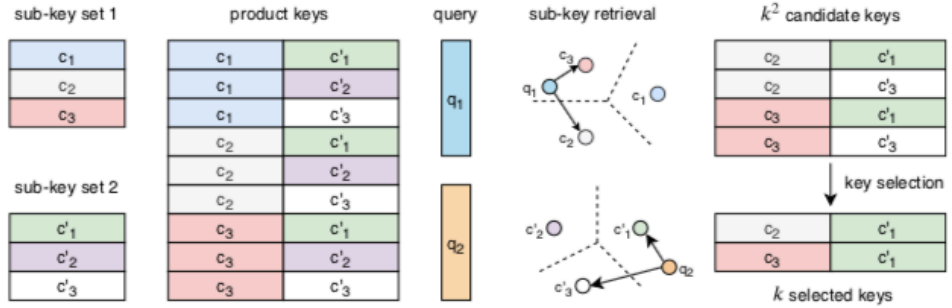

Figure 2: **Illustration of the product keys.** We define two discrete subsets of keys (sub-key set 1 and sub-key set 2). They induce a much larger set of keys, which are never made explicit (product keys). Given a query, we split it into two sub-queries (q_1 and q_2). Selecting the k closest keys ($k = 2$ in the figure) in each subset implicitly selects $k \times k$ keys. The k keys maximizing the inner product with the query are guaranteed to belong to this subset, on which the search can be done efficiently.

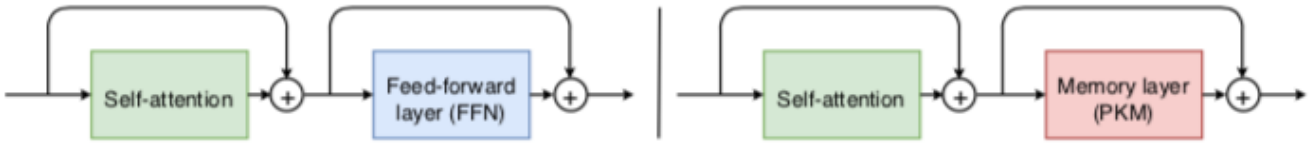

Figure 3: **Left:** A typical transformer block is composed by a self-attention layer followed by an FFN layer (a two layer network). **Right:** In our system, we replace the FFN layer with a product-key memory layer, which is analogous to a sparse FFN layer with a very large hidden state. In practice, we only replace the FFN layer in N layers, where typically $N \in \{0, 1, 2\}$.

Results

```
1 ROUGE-1 Average_R: 0.08686 (95%-conf.int. 0.08639 - 0.08732)
1 ROUGE-1 Average_P: 0.29698 (95%-conf.int. 0.29538 - 0.29857)
1 ROUGE-1 Average_F: 0.12360 (95%-conf.int. 0.12303 - 0.12416)
---------------------------------------------
1 ROUGE-2 Average_R: 0.01086 (95%-conf.int. 0.01073 - 0.01101)
1 ROUGE-2 Average_P: 0.04469 (95%-conf.int. 0.04419 - 0.04522)
1 ROUGE-2 Average_F: 0.01602 (95%-conf.int. 0.01583 - 0.01620)
---------------------------------------------
1 ROUGE-L Average_R: 0.08005 (95%-conf.int. 0.07965 - 0.08048)
1 ROUGE-L Average_P: 0.27174 (95%-conf.int. 0.27033 - 0.27307)
1 ROUGE-L Average_F: 0.11346 (95%-conf.int. 0.11296 - 0.11395)

[2019-11-08 19:08:48,814 INFO] Rouges at step 100000
>> ROUGE-F(1/2/3/l): 12.36/1.60/11.35
ROUGE-R(1/2/3/l): 8.69/1.09/8.00
```

Extractive Hiersumm

- Modification code Reference: Text Summarization with Pretrained Encoders(Yang Liu and Mirella Lapata, EMNLP2019)

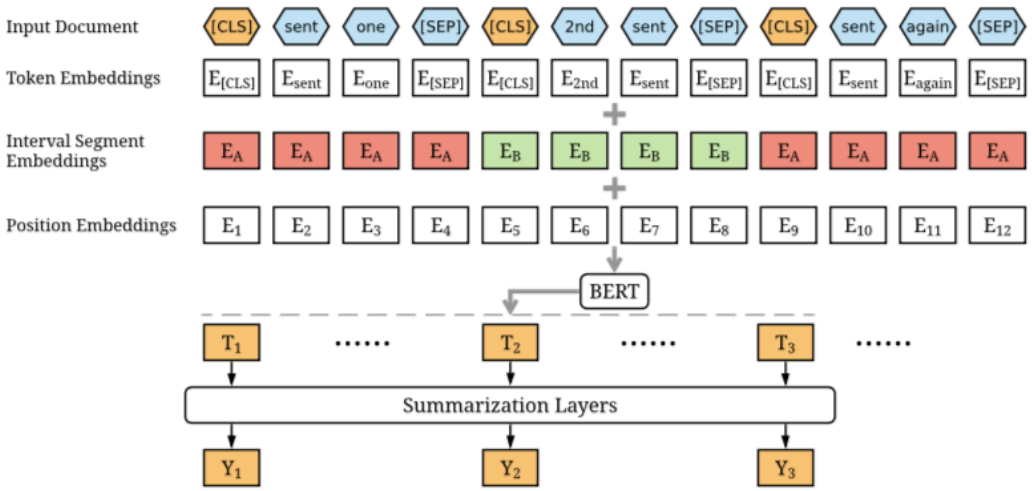

Figure 1: The overview architecture of the BERTSUM model.

Conclusion and Future work

- ~~PKM~~
- Reduce memory usage for extractive hiersumm