Mathematical Challenge July 2019

Extractive multi-document query-focused summarization

References

[1] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*, 2004.

[2] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In Satinder P. Singh and Shaul Markovitch, editors, *AAAI*, pages 3075–3081. AAAI Press, 2017.

[3] Ramesh Nallapati, Bowen Zhou, and Mingbo Ma. Classify or select: Neural architectures for extractive document summarization. *CoRR*, abs/1611.04244, 2016.

[4] Pengjie Ren, Zhumin Chen, Zhaochun Ren, Furu Wei, Jun Ma, and Maarten de Rijke. Leveraging contextual sentence relations for extractive summarization using a neural attention model. In Noriko Kando, Tetsuya Sakai, Hideo Joho, Hang Li, Arjen P. de Vries, and Ryen W. White, editors, *SIGIR*, pages 95–104. ACM, 2017.

[5] Xiaojun Wan. Towards a unified approach to simultaneous single-document and multi-document summarizations. In Chu-Ren Huang and Dan Jurafsky, editors, *COLING*, pages 1137–1145. Tsinghua University Press, 2010.

[6] Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. A sentence compression based framework to query-focused multi-document summarization. In *ACL (1)*, pages 1384–1394. The Association for Computer Linguistics, 2013.

Description

Automatic summarization is the process of shortening a text document with software, in order to create a summary with the major points of the original document, i.e. find a subset of data which contains the 'information' of the entire set. Summarization techniques can roughly be divided into two large sub-groups according to the input and output type. Input

can either consist of a single document or a cluster of related documents (multi-document summarization). Output of a summarization engine can be extractive or abstractive. Extractive summarizers identify the most important sentences in the input (conveying key information) and string them together to form a summary. Abstractive summarization involves paraphrasing sections of the source input, i.e. identifying key concepts and forming understandable sentences conveying the key information.

Motivation

Query-focused summarization has many applications in finance:

- information extraction (document database search engines)
- customer service and retention (robo-advisors and chatbots)
- assistant tools for diverse applications (i.e interactive surveys for credit scoring)

Abstractive framework allows for some desirable abilities, such as paraphrasing, generalization or the incorporation of real-world knowledge. However, extractive approaches are often more attractive since they are easily interpretable, generally trained faster and because baseline levels of grammaticality and accuracy are ensured (which is not a given when summaries are abstractive).

Technical Details

Architecture

Recent state-of-the-art algorithms for extractive summarization ([2], [3]) have turned to neural network-based architectures. [6] and [4] additionally consider existence of queries in a multi-document setup.
Here, we will shortly present the approach outlined in [2]. They consider extractive summarization to be a sequence classification problem, where sentences are visited sequentially and decisions are made whether they should be included in the summary or not (taking into account previous decisions). The architecture can be seen on Figure 1.

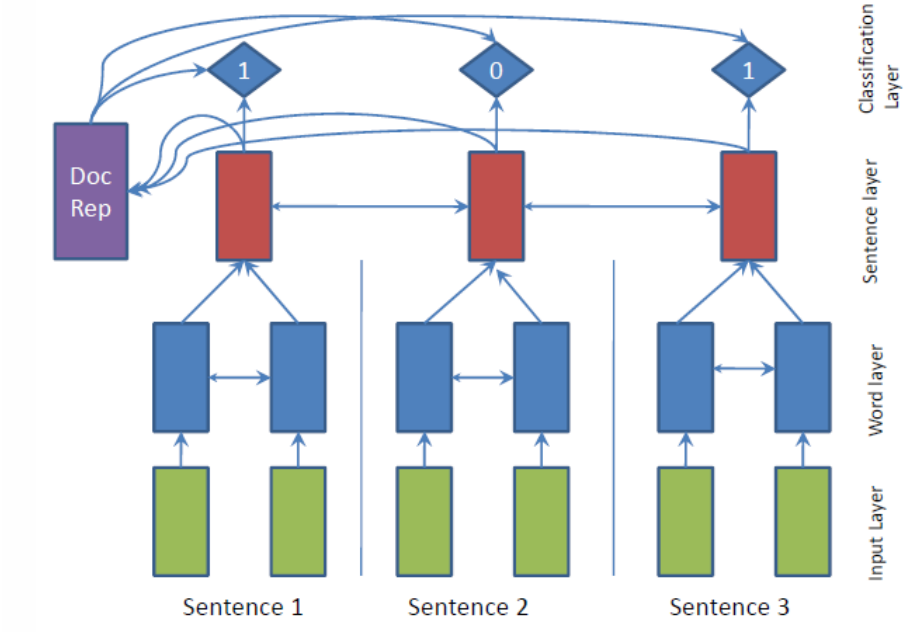

Figure 1: Two-layer RNN based sequence classifier

A two layer GRU-RNN is used as the base building block. It has two gates, namely update u and reset r, and can be described using following equations:

$$u_j = \sigma(W_{ux}x_j + W_{uh}h_{j-1} + b_u)$$
$$r_j = \sigma(W_{rx}x_j + W_{rh}h_{j-1} + b_r)$$
$$h_j^{'} = \tanh(W_{hx}x_j + W_{hh}(r_j \odot h_{j-1}) + b_u)$$
$$h_j = (1 - u_j) \odot h_j^{'} + u_j \odot h_{j-1}$$

where W and b are the parameters of the GRU-RNN and h_j is the real-valued hidden-state vector at timestep j, x_j is the corresponding input vector, and $\odot$ represents the Hadamard product.

The first layer runs at the word level to computes hidden state representation at each word position sequentially, based on the current word embeddings and the previous hidden state. The RNN is bi-directional, i.e one RNN runs from first to last word, while the other runs from last to first.

Second layer of the bi-directional RNN runs at the sentence-level and accepts the average-pooled, concatenated hidden states of the bi-directional word-level RNNs as input. The hidden states of the second layer RNN encode the representations of the sentences in the document. The representation of the entire document is then modeled as a non-linear transformation of the average pooling of the concatenated hidden states of the bi-directional sentence-level RNN:

$$d = \tanh\left(W_d \frac{1}{N_d} \sum_{j=1}^{N^d} \left[h_j^f, h_j^b\right] + b\right)$$

where h_j^f and h_j^b are the hidden states corresponding to the j-th sentence, N^d is the number of sentences and [] represents vector concatenation.
For classification, each sentence is revisited sequentially, where a logistic layer makes a decision as to whether that sentence belongs to the summary:

$$\begin{aligned} P\left(y_j = 1 | h_j, s_j, d\right) = & \sigma(W_c h_j && \text{(relevance)} \\ & + h_j^T W_s d && \text{(salience)} \\ & - h_j^T W_r \tanh(s_j) && \text{(novelty)} \\ & + W_{ap} p_j^a && \text{(absolute positional importance)} \\ & + W_{rp} p_j^r && \text{(relative positional importance)} \\ & + b) && \text{(bias term)} \end{aligned}$$

with $s_j = \sum_{i=1}^{j-1} h_i P(y_i = 1 | h_i, s_i, d)$ being the dynamic representation of the summary at position j and p^a, p^r being absolute and relative positional embeddings.
The model is trained by minimizing the negative log likelihood of the observed labels at training time.

Extractive supervision using abstractive summaries as ground truth

In case there are no explicit labels that can be used for extractive supervision, the ground truth in form of abstractive summary has to be transformed into sentence-level binary labels for each document, representing their membership in the summary. This is often the case, since many summarization corpora only contain human written abstractive summaries as ground truth.
One approach to solve this problem suggested in [2], is to use an unsupervised approach to convert the abstractive summaries to extractive labels. It is based on the idea that the selected sentences from the document should be the ones that maximize the Rouge score [1] with respect to abstractive summaries. Since it is computationally expensive to find a globally optimal subset of sentences that maximizes the Rouge score, a greedy approach is used where a sentence is added at a time incrementally to the summary, such that the Rouge score of the current set of selected sentences is maximized with respect to the reference summary .

Questions

- **Q1:** *Modify the approach in [2] to allow for inclusion of queries.*
- **Q2:** *Approach outlined in [2] suffers from domain adaptation (lower quality when summarizing documents from a corpus which is different to the one used for training). Investigate potential solutions to this problem. Use the DUC datasets (`https://duc.nist.gov/duc2005/tasks.html`) for benchmarking [5].*

- **Q3:** Investigate the approach outlined in [6] for summary post-processing and compression. Potential redundant information contained in extractive summaries is an inherent problem, since the summary is assembled directly from input sentences.

We look forward to your opinions and insights.

Best Quant Regards,

swissQuant Group Leadership Team