Report 3

> 2022.3.6-2022.3.13

Bottom-Up Abstractive Summarization

- Content selection stage: Use a neural sequence-tagging model to tag words as include or don't-include
- Bottom-up attention stage: The seq2seq+attention system can't attend to words tagged don't-include (apply a mask)

$$p(\tilde{a}_j^i|x, y_{1:j-1}) = \begin{cases} p(a_j^i|x, y_{1:j-1}) & q_i > \epsilon \\ 0 & \text{ow.} \end{cases}$$

> simple but effective (better selection and less copying)
>
> 两步走：计算词被选中的概率q_i 然后用此概率影响copy的概率。copy是see et al.的工作中pointer部分

Abstractive Text Summarization by Incorporating Reader Comments

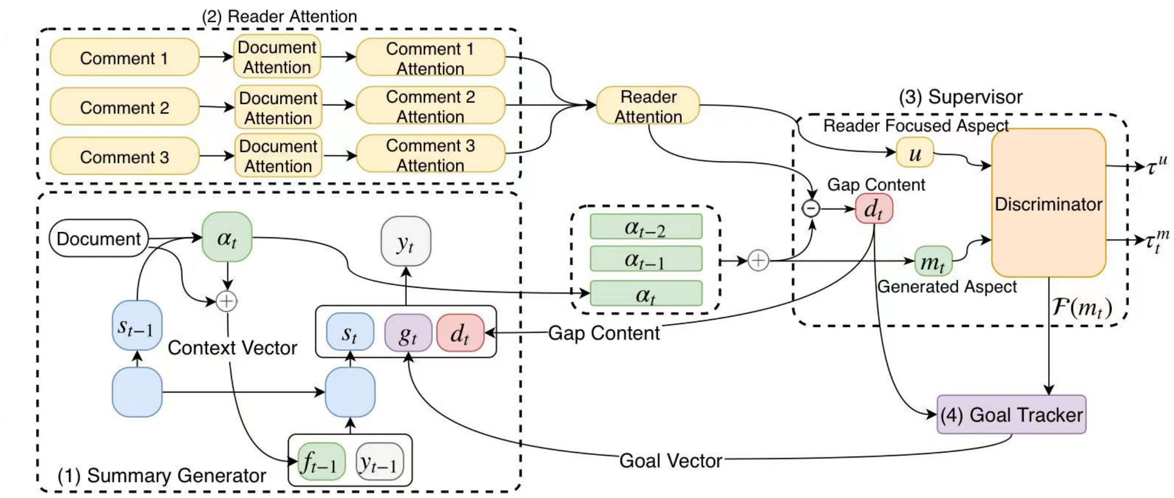

Figure 1: Overview of RASG. We divide our model into four parts: (1) *Summary generator* generates a summary to describe the main aspect of document. (2) *Reader attention* module models the readers attention of document. (3) *Supervisor* models the gap of focused document aspect between generated summary and reader comments. (4) *Goal tracker* sets a goal of summary generator according to gap given by supervisor.

本篇论文的创新之处在于将comments作为生成总结的部分依据。关注的问题有如何筛选comments，然后怎么把comments与生成过程结合。

模型有点复杂

- summary generator attention+seq2seq略有改变
- reader attention document第i个词的attention有所有comment词相似度加起来决定
- supervisor衡量decoder和reader comment的attention focus
- Goal tacker对其进行优化

- ## 总结一下读了的5篇abstractive summary

A Neural Attention Model for Abstractive Sentence Summarization是最基础的文本生成模型，是将attention和neural network用于文本生成的前期工作。

Get To The Point: Summarization with Pointer-Generator Network综合了Pointer-Generator和courage mechanism，保证了词汇库中没有原文有的词也有一定的几率生成，同时避免了重复。相关工作前人均有涉及，此篇论文有较强的工程性。后三篇都以这个为基础。

Closed-Book Training to Improve Summarization Encoder Memory多加了一个decoder

Bottom-Up Abstractive Summarization在最前面先对关键词进行筛选。

Abstractive Text Summarization by Incorporating Reader Comments讲情境放在了互联网文本上，由评论的辅助生成文本总结。基本模型还是第二篇。

代码部分

完成了cs224n assignment4的代码填空。通过了sanity check但是没GPU训练

- utils.py包含了read file, sort source and target sentence, 将sentence等长三个函数。这里填空的是pad_sents函数，即遍历句子列表给每个不够长的句子加上[pad_token]
- model_embeddings.py给source和target加上embedding layer
- nmt_model.py最重要的部分。这里完成的填空是初始化，encode，decode和step。初始化照着模型的结构，搞清楚输入输出tensor的维度即可。encode作用在source上返回hidden state和initial state of decoder。decode计算得到output vectors。step是decode的一步计算，同时包括了attention的计算。除此之外，模型还包括forward函数：输入一个mini-batch，调用encode和decode后计算log-likelihood。

基本上能看懂一个模型的大致框架，但具体实现细节还需要脚手架，经常会搞不清变量和步骤。

这周读的两篇论文都有相应代码，还没来得及阅读。