Fast Abstractive Summarization with Reinforce-Selected Sentence Rewriting-hoho

论文试图解决什么问题？

文档归纳总结当前使用两种范式：信息抽取或抽象化归纳。但是这两种方法对于篇幅长的文档来说，处理速度慢并且结果不准确。即使有attention机制，可以考虑所有的单词，但也是很慢。另外生成句子时也比较容易出发重复单词的现象。

本文需要解决这种文档归纳不准确且生成速度慢的问题

这是否是一个新的问题？

不是新的问题。从最开始只使用信息抽取来归纳，到后来尝试用抽象化归纳，再后来有seq2seq神经网络的加持，一直在力图提升模型的准确度。

这篇文章要验证一个什么科学假设？

用强化学习来架构语义归纳生成模型，能否提高生成的准确度和生成速度。

有哪些相关研究？如何归类？谁是这一课题在领域内值得关注的研究员？

当时的研究工作主要有：

1. 基础模型的研究，包括早期的信息抽取与压缩的基本方法，如基于HMM、基于规则的信息提取，后来出现用神经网络的抽象概括方法，还有在神经网络上加上分层注意力机制的
2. 基于强化学习来优化不可微语言生成度量的方法
3. 在机器翻译上如何提高解码质量和实时性
4. QA领域用于生成answer的方法

比较值得关注的研究员如Abigail See和Oriol Vinyals，跟pointer network相关的研究。而且本文的实验大部分是跟Abigail See的研究作对比。

论文中提到的解决方案之关键是什么？

作者构建如下MDP进行文档总结归纳：

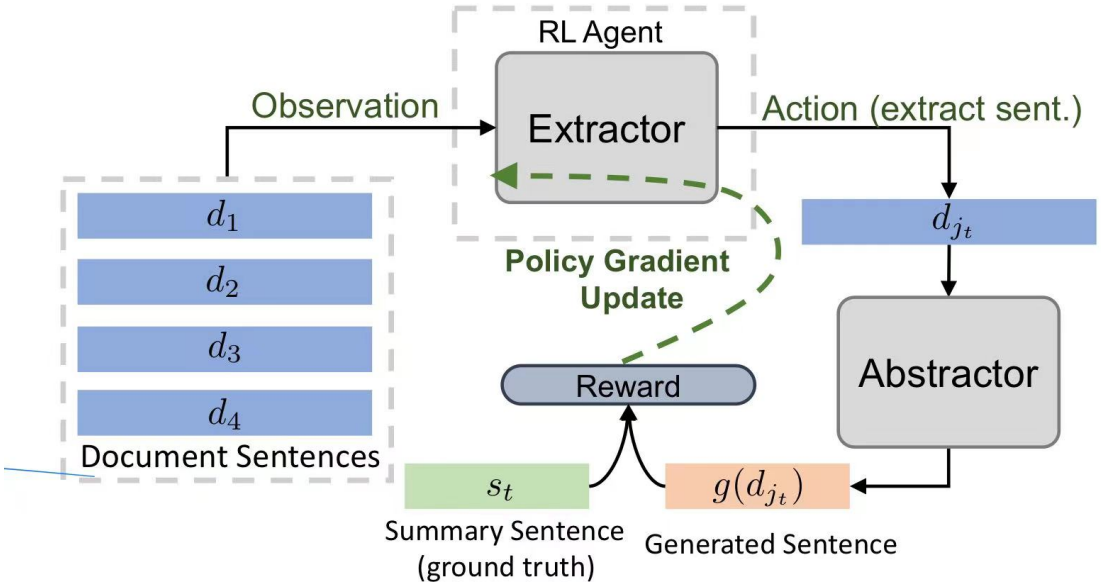
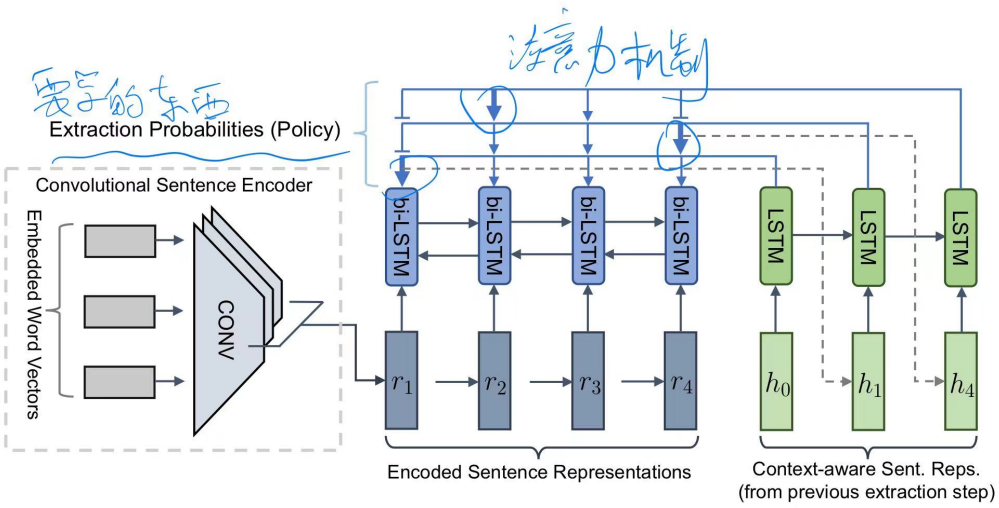


Figure 2: Reinforced training of the extractor (for

- Extractor作为Agent，接收环境状态（输入）$c = (D, d_{j_{t-1}})$，D是文档的句子集合，$d_{j_{t-1}}$是上一个时间步的归纳出来的句子

Extractor的架构如图：



1. conv卷积网络encode每一个句子

2. 句子向量输入到Bi-LSTM（上图蓝色）提取特征h_j
3. 使用另一个LSTM（上图绿色）训练一个pointer network，目的是学习一个提取句子的概率分布（策略）：

$$u_j^t = \begin{cases} v_p^T tanh(W_{p1} h_j + W_{p2} e_t) & \text{if } j_t \neq j_k \\ -\infty & \text{if } otherwise \end{cases}$$

$$P(j_t|j_1, ..., j_{t-1}) = softmax(u^t)$$

其中，

$$\begin{aligned} a_j^t &= v_g^T tanh(W_{g1} h_j + W_{g2} z_t) \\ \alpha^t &= softmax(a^t) \\ e_t &= \sum_j \alpha_j^t W_{g1} h_j \end{aligned}$$

W和v都是训练参数。

这里其实使用了两个注意力机制：z_t是绿色部分的LSTM的输出，它attent到h_j生成e_t，然后又attent到h_j生成u_j^t

- Abstractor使用标准的encoder-decoder架构，对extractor agent的输出进行抽象改写

学习过程如下：

1. 为了避免随机初始化时Extractor的网络总是选择不相关的句子传给Abstractor进行抽象改写，另一方面，Abstractor由于没经过训练，使得最终产生带noisy的回报。于是首先分别对Extractor和Abstractor进行最大似然估计的优化：

（1）Extractor最大似然：由于训练集是“文档-归纳文本“的数据对，这里需要对训练标签进行转换，对每个归纳文本寻找最相近的文档中的句子d_i：$j_t = \arg\max_i(ROUGE - L_{recall}(d_i, s_t))$，然后最小化交叉熵进行优化；

（2）Abstractor最大似然：使用归纳文本与Extractor提取的句子进行训练数据，进行交叉熵：$L(\theta_{abs}) = -\frac{1}{M}\sum_{m=1}^{M} logP_{\theta_{abs}}(w_m|w_{1:m-1})$

2. RL训练过程：环境输入状态$c_t = (D, d_{j_{t-1}})$，agent采样动作$j_t \sim P(j)$以提前文档中的一个句子d_{j_t}，环境返回回报：$r(t+1) = ROUGE - L_{F_1}(g(d_{j_t}), s_t)$，其中g是Abstractor，整个过程使用A2C算法

论文中的实验是如何设计的？

作者另外建立了一个简单的feed forward network extractor作为baseline。对extractor、abstractor与RL的搭配组合进行实验对照。

使用ROUGE和METEOR作为模型度量。

本文也是使用了人类直觉进行验证，使用Amazon Mturk实验平台，将Abigail See研究员的研究模型跟本文模型的输出做对比，进行好坏排序。

用于定量评估的数据集是什么？代码有没有开源？

使用CNN/Daily Mail数据集（https://www.github.com/deepmind/rc-data/），使用DUC-2002数据集用于测试

代码已经开源：https://github.com/ChenRocks/fast_abs_rl

论文中的实验及结果有没有很好地支持需要验证的科学假设？

实验基本上达到当时的SOTA

Models	ROUGE-1	ROUGE-2	ROUGE-L	METEOR
Extractive Results				
lead-3 (See et al., 2017)	40.34	17.70	36.57	22.21
Narayan et al. (2018) (sentence ranking RL)	40.0	18.2	36.6	-
ff-ext	40.63	18.35	36.82	**22.91**
rnn-ext	40.17	18.11	36.41	22.81
rnn-ext + RL	**41.47**	**18.72**	**37.76**	22.35
Abstractive Results				
See et al. (2017) (w/o coverage)	36.44	15.66	33.42	16.65
See et al. (2017)	39.53	17.28	36.38	18.72
Fan et al. (2017) (controlled)	39.75	17.29	36.54	-
ff-ext + abs	39.30	17.02	36.93	20.05
rnn-ext + abs	38.38	16.12	36.04	19.39
rnn-ext + abs + RL	40.04	17.61	37.59	**21.00**
rnn-ext + abs + RL + rerank	**40.88**	**17.80**	**38.54**	20.38

在人类验证上，从CNN/DM数据集随机采样100个样本，对每个样本使用本文的模型与Abigail See研究的模型输出作对比，从可读性和相关性方面，每个样本询问3个测

试员，对结果进行排序。结果显示本文模型较优：

	Relevance	Readability	Total
See et al. (2017)	120	128	248
rnn-ext + abs + RL + rerank	**137**	**133**	**270**
Equally good/bad	43	39	82

在生成速度上也较优：

	Speed	
Models	total time (hr)	words / sec
(See et al., 2017)	12.9	14.8
rnn-ext + abs + RL	0.68	361.3
rnn-ext + abs + RL + rerank	2.00 (1.46 +0.54)	109.8

Table 5: Speed comparison with See et al. (2017)

这篇论文到底有什么贡献？

1. 是第一个提出句子级别的使用RL技术的文档归纳生成方法，有效利用word-then-sentence分层结构，使用不带标记的机器学习方法（文档-归纳文本对）
2. 达到当时SOTA，无论从人工验证还是机器验证
3. 比之前最好的生成模型生成速度快10~20倍

下一步呢？有什么工作可以继续深入？

结合强化学习的一些NLP任务。