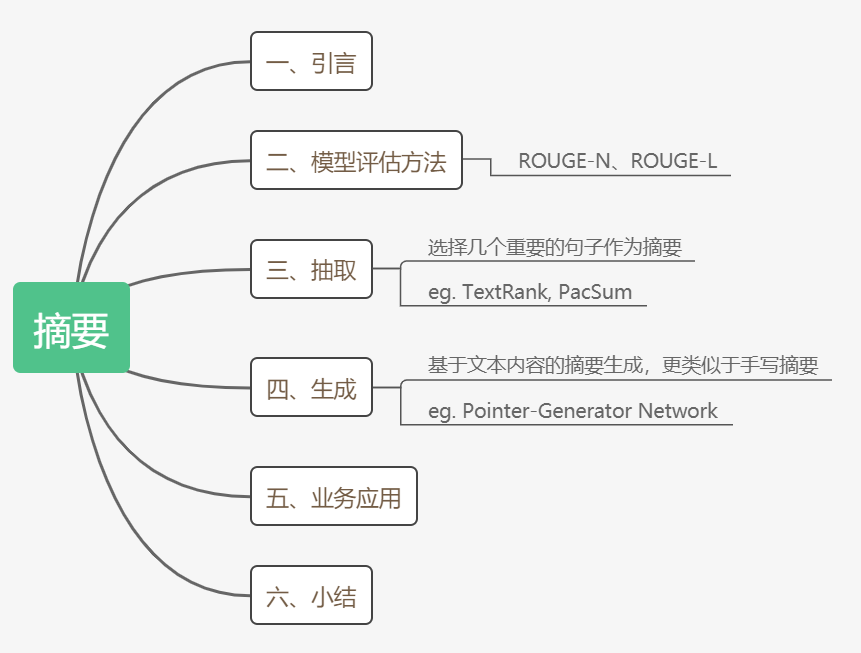


启航实验室(Statistical Artificial Intelligence and Learning Lab, SAIL Lab)是杨海钦博士于2019年9月入职后创立的实验室。寓意着实验室如大航海时代遨游在人工智能的知识海洋中，以统计学习为主要手段，积极探索前沿技术，推进技术创新，打造数据赋能的启航引擎平台SEED (SAIL Engine Empowered by Data)平台, 实现在自然语言处理、推荐系统、计算机视觉等领域的落地应用，在平安寿险人工智能团队中作为技术源动力，孵化新技术，引领技术革新。目前主要提供2大服务：在线服务(寒暄引擎)和离线服务(生成引擎)。

本次组内实习生施建华在尹曦老师的辅导下带来《摘要抽取的技术分享》。分享前，给大家带来两个广告：

1. 招聘优秀实习生
2. 欢迎大家关注并参与亚太神经网络学会支持的ICONIP’20会议，会议链接：<http://www.iconip2020.apnns.org/>

**分享概览**



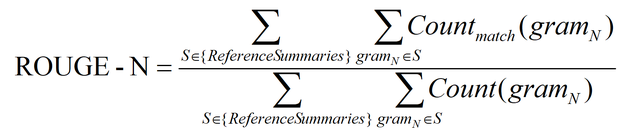
1. **引言**

文本摘要旨在将长文本进行压缩、归纳和总结，形成新的短文本对原文本进行精简地概括。根据文档的个数不同，文本摘要任务可以分成单文档摘要和多文档摘要。现在，大部分摘要模型都以单文档摘要为主。根据摘要的方法不同，亦可以分成抽取式方法[1][2]和生成式方法[3][4]。此两种方法的优缺点归纳如下：

|  |  |  |
| --- | --- | --- |
|  | 抽取 | 生成 |
| 优点 | 从原文抽取句子，所以句子流畅度较高 | 自动生成句子，抽象度较高，更具总结性质 |
| 缺点 | 会引入更多的冗余信息，无法体现摘要本身的特点 | 文本长度越长，效果越差。抽象程度越高，越难以评估 |

1. **摘要评估方法**

ROUGE(Recall-Oriented Understudy for Gisting Evaluation) [5]，是摘要质量评估的主流方法，它的核心想法来源于召回率，也就是说生成出来的摘要跟原本的人工摘要匹配度越高，分数就越高。而考虑到字和词的匹配情况，通常以ROUGE-*n*表示相应的召回率, 其中*n*表示对应的字数。通常实验会统计ROUGE-1 和 ROUGE-2作为实验结果。公式如下：

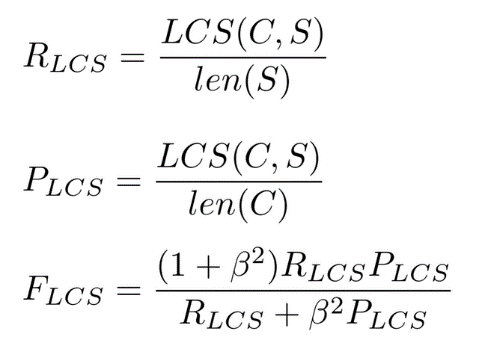


$$

\operatorname{ROUGE}-\mathrm{N}=\frac{\sum\_{S\in\{\text {Referencesummaries}\}} \sum\_{\text {gram}\_{N} \in S}\operatorname{Count}\_{\text {match}}\left(\operatorname{gram}\_{N}\right)}{\sum\_{S \in\{\text {ReferenceSummaries}\}} \sum\_{\text {gram}\_{N} \in S} \operatorname{Count}\left(\operatorname{gram}\_{N}\right)}

$$

另外，ROUGE-L (longest common subsequence, LCS) 亦常用做实验结果评估的指标。它通过输出摘要和原本摘要的最长公共子序列的匹配情况和准确度，计算出最后得分。公式如下：



$$

\begin{aligned}

R\_{L C S} &=\frac{L C S(C, S)}{\operatorname{len}(S)} \\

\\

P\_{L C S} &=\frac{L C S(C, S)}{\operatorname{len}(C)} \\

\\

F\_{L C S} &=\frac{\left(1+\beta^{2}\right) R\_{L C S} P\_{L C S}}{R\_{L C S}+\beta^{2} P\_{L C S}}

\end{aligned}

$$

其中*C*为输出摘要，S为人工摘要， *RLCS* , *PLCS*  和*FLCS*为对应的召回率(Recall), 精确度(Precision)和F值。

1. **抽取**

抽取式摘要的目的是选择文档中最能代表文档内容的几个重要句子作为摘要。可大致分为以下几个步骤：

1. 理解文档的内容
2. 对语句的重要性的计算和排名
3. 选择排名较高的几个语句
4. 对选择的语句排序
5. 得出最后摘要

以较早的TextRank [6]为例，它的灵感来源于大名鼎鼎的PageRank[7]算法，这是一个用作网页重要度排序的算法。TextRank 通过将文本分割成若干组成单元 (单词、句子) 并以此建立基于图的模型，图的每条边可以理解成每个单元 (单词、句子) 之间的关联，透过句子之间的关联性建立一个投票机制，找出在文档中重要的句子进行排序，也就是得票高者获胜的概念。最后，取几个较高分的句子作为摘要。公式如下：



$$

\text {Similarity}\left(S\_{i}, S\_{j}\right)=\frac{\mid\left\{w\_{k} \mid w\_{k} \in S\_{i} \& w\_{k} \in S\_{j}\right\} \mid}{\log \left(\left|S\_{i}\right|\right)+\log \left(\left|S\_{j}\right|\right)}

$$

其中，在图中节点的分数通过相邻节点的信息采用投票机制计算如下：



$$

W S\left(V\_{i}\right)=(1-d)+d \* \sum\_{V\_{j} \in I n\left(V\_{i}\right)} \frac{w\_{j i}}{\sum\_{V\_{k} \in Out\left(V\_{j}\right)} w\_{j k}} WS\left(V\_{j}\right)

$$

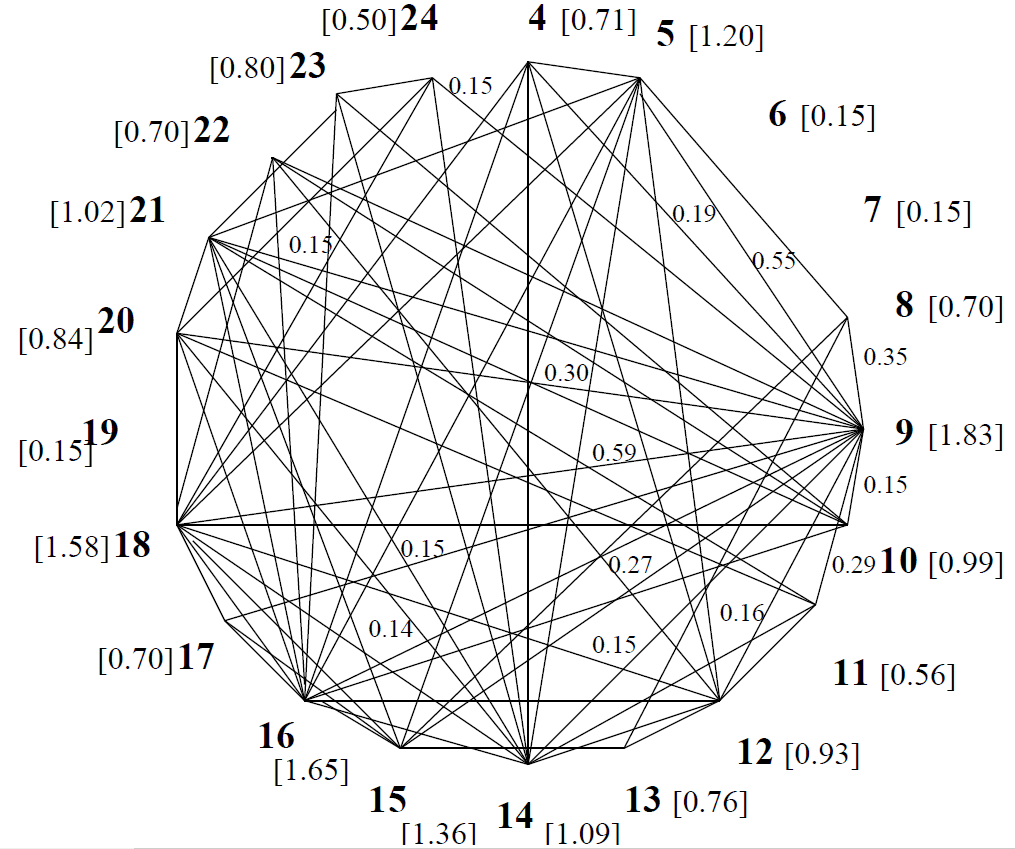


图5: TextRank图结构

随着预训练模型的兴起，PacSum [8]模型提出将BERT [9]预训练模型加入抽取式摘要的工作，并获得了更好的结果。其结构图如下：

一張含有 電子用品, 電腦, 螢幕, 鍵盤 的圖片

自動產生的描述

图6：PacSum 模型结构

如图6所示，PacSum 将文档分割成若干个句子，再将这些句子输入给 BERT，BERT 通过它的 Self-Attention 机制可以很好地学习各个句子之间的语义关系，最后将句子的语义向量输出。

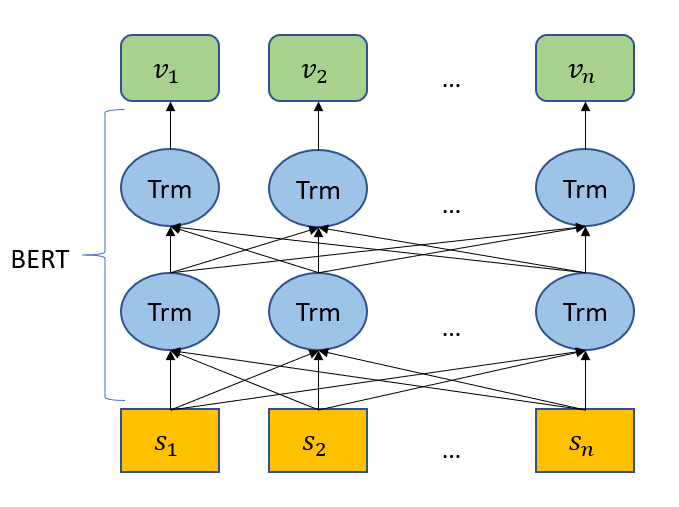


图7: *si*:第*i*句输入 *vi*:第*i*句的向量输出

得到多个具有丰富语义信息的向量后，PacSum 计算各个向量之间的相识度以决定各个句子对该文档的重要性。公式如下：

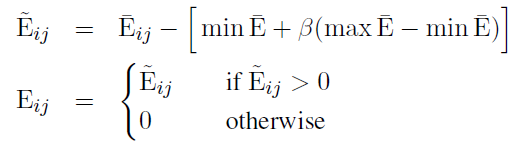


$$

\overline{\mathrm{E}}\_{i j}=v\_{i}^{\top} v\_{j}

$$

因为使用点积计算矩阵，所以再对矩阵进行修改操作，防止数值过大。计算如下:



$$

\begin{array}{l}

\tilde{\mathrm{E}}\_{i j}=\overline{\mathrm{E}}\_{i j}-[\min \overline{\mathrm{E}}+\beta(\max \overline{\mathrm{E}}-\min \overline{\mathrm{E}})] \\

\mathrm{E}\_{i j}=\left\{\begin{array}{ll}

\tilde{\mathrm{E}}\_{i j} & \text { if } \tilde{\mathrm{E}}\_{i j}>0 \\

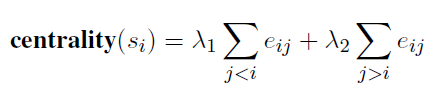
0 & \text { otherwise }

\end{array}\right.

\end{array}

$$

如果一个句子里包含了重要信息，那么该重要信息就有很高的概率被重复提起且出现在文档的靠前部分。这是根据正常的写作习惯推论出来的，大多数人在写作的时候，都会将关键性概念或者论点放在推论或扩展之后。最典型的例子便是新闻报道，关键句子往往会出现在文章的开头。所以，PacSum 透过正常的写作习惯结合相似度的方法对句子的重要度进行排序，最后选取将作为摘要的重要语句。方法如下：



$$

\operatorname{\textbf{centrality}}\left(s\_{i}\right)=\lambda\_{1} \sum\_{j<i} e\_{i j}+\lambda\_{2} \sum\_{j>i} e\_{i j}

$$

其中centrality(*s­­i*)为第*i*句的重要性，上式通过*λ*1, *λ*2加入句子的位置影响因素。

PacSum的实验结果证明相比于TextRank和其他抽取式摘要模型都有较优的结果。它也证明了根据写作习惯可以更好地抽取重要的摘要信息。实验结果如下：

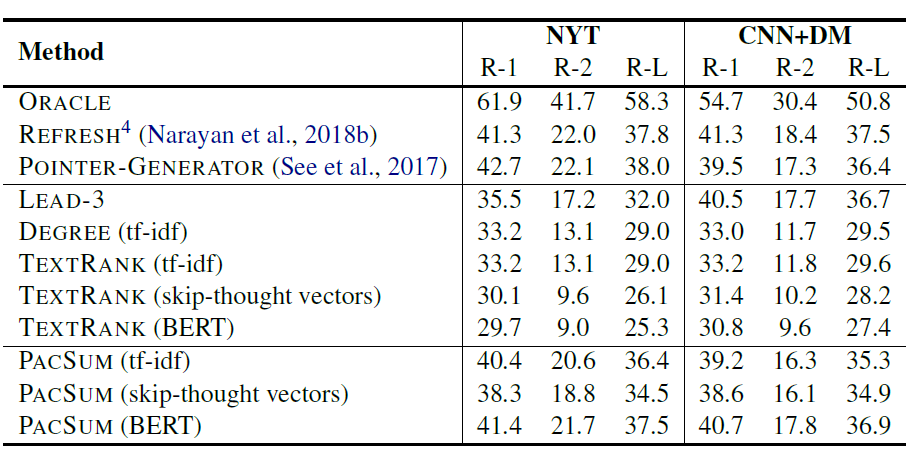


图11：实验结果[8]: R-1, R-2, R-L 为 ROUGE-1, ROUGE-2,ROUGE-L

1. **生成**

生成式摘要又称抽象式摘要生成，它的目的是通过模型生成出更具有人工摘要风格的摘要，也就是说生成出来的摘要要更抽象、更流畅、更具有自然语言的逻辑特性。所以，生成式摘要任务比抽取式更具难度且更难以评估其正确性，但更具有摘要的价值。

生成式摘要的结构基本上都是基于 Encoder-Decoder (编码-解码)，为了更好地将文本信息保留，现在有各式各样的编码器，有CNN-based、RNN-based、Graph-based，较流行的为 RNN-based。但随着预训练模型的兴起，Transform-based 摘要模型也占据了生成式摘要模型的一角。

本文将介绍比较经典的 Pointer-Generator Network [10]。这一个 Sequence-to-Sequence + Attention [11][12]的摘要模型，它引入了Copy Mechanism [13]和Coverage Mechanism 用来解决 Out-Of-Vocabulary(OOV)和重复生成的问题

1. 生成式摘要所面临的三个限制：
   * 1. Accuracy (准确度)，因为生成式摘要往往更具抽象度，所以其摘要准确度会比抽取式摘要弱上许多，但此模型在准确度上，可以表现得不弱于抽取式摘要。
     2. Out-Of-Vocabulary(OOV)，为了达到抽象生成摘要的目的，模型需要有预先准备好的词库，并从中选择。但这就可能出现词库中找不到好的词或字在摘要中进行搭配，这便是OOV的问题了。而此模型提出了Copy Mechanism很大程度上解决了这个问题。
     3. Repetition (重复生成)，在Decoder 解码的过程中，随着解码序列增长，模型忘记了已经生成的内容，进而对一个重要信息反复地解码，生成重复的摘要内容。为了解决整个问题，此模型提出了Coverage Mechanism，利用惩罚重复生成的损失函数来学习避免重复生成的能力。
2. Sequence to Sequence + Attention 模型

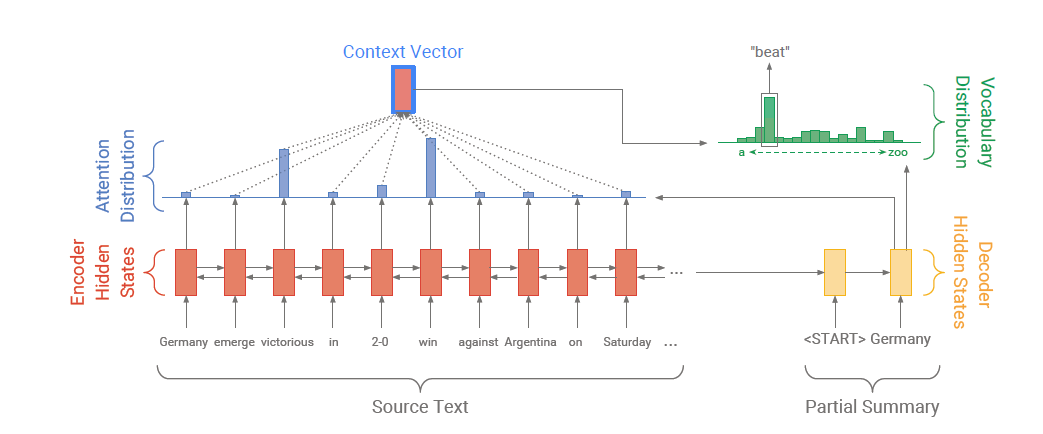
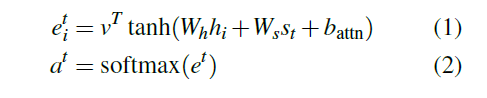


图12：Sequence to Sequence Attention 模型结构

* + 1. Sequence to Sequence: 这是一个典型的Encoder-Decoder的模型结构，此模型用单层双向的LSTM作为Encoder，由此可以学习文档中的上下文语义信息，用单层单向的LSTM作为Decoder能快速地解码。
    2. Attention：对Encoder层的每一步输出和当前输出做加权计算，最后通过softmax 得到 Attention Distribution。如下所示:



$$

\begin{array}{l}

e\_{i}^{t}=v^{T} \tanh \left(W\_{h} h\_{i}+W\_{s} s\_{t}+b\_{\text {attn }}\right) \\

a^{t}=\operatorname{softmax}\left(e^{t}\right)

\end{array}

$$

图13：Attention Distribution 公式

* + 1. Context Vector: 得到Attention Distribution后，便可以得到一个固定大小的文本表达，Context Vector，通过它就可以得到最后的字或词的分布了。



$$

h\_{t}^{\*}=\sum\_{i} a\_{i}^{t} h\_{i}

$$



$$

P\_{\text {vocab }}=\operatorname{softmax}\left(V^{\prime}\left(V\left[s\_{t}, h\_{t}^{\*}\right]+b\right)+b^{\prime}\right)

$$

图14：最终的词分布

1. Copy mechanism

概念定义：

* + 1. Pointer，指对文档包含的字或词的选择器，其vocabulary就是文档中所有的字或词，每步解码时，计算在文档字或词上的分布概率，选择概率最高的字或词作为结果。
    2. Generator，指原有的解码器，具有自己的vocabulary，通常包含常见的字或词和领域的特征词，与Pointer一样，Generator在每步解码的时候，选择概率最高的字或词作为结果。
    3. Switch，指解码结果的选择器，每步解码时决定选择Pointer的结果或者Generator的结果作为最终的结果。

计算方法：

1. Pointer: 模型将作为Pointer的attention distribution，而是文档位置上的概率分布，因此需要将不同位置上相同字或词的attention score相加，最后得到 作为最终的概率分布。
2. Generator: 模型按照以下的公式，以解码器的方式计算概率分布。
3. Switch：选择器如以下所示是一个二分类器，其结果将作为选择的概率，从而得出最终概率分布结果。



$$

p\_{\text {gen }}=\sigma\left(w\_{h^{\*}}^{T} h\_{t}^{\*}+w\_{s}^{T} s\_{t}+w\_{x}^{T} x\_{t}+b\_{\mathrm{ptr}}\right)

$$



$$

P(w)=p\_{\mathrm{gen}} P\_{\mathrm{vocab}}(w)+\left(1-p\_{\mathrm{gen}}\right) \sum\_{i: w\_{i}=w} a\_{i}^{t}

$$

图15：加入Copy Mechanism后的最终词分布

1. Coverage Mechanism

此模型通过惩罚重复生成的损失函数来学习避免重复生成的能力。具体步骤如下：

* + 1. Coverage Vector: 其作用是使解码器在第t步解码时知晓前t-1步对某位置的总注意力，从而在第t步时避免对该位置过多注意，导致重复生成。

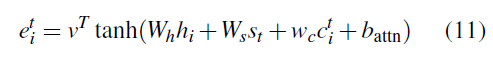


$$

c^{t}=\sum\_{t^{\prime}=0}^{t-1} a^{t^{\prime}}

$$

图16：Coverage Vector公式



$$

e\_{i}^{t}=v^{T} \tanh \left(W\_{h} h\_{i}+W\_{s} s\_{t}+w\_{c} c\_{i}^{t}+b\_{\text {attn }}\right)

$$

图17：Coverage Vector加入到 Attention机制

* + 1. Coverage Loss: 如果只添加Coverage Vector并不能快速学习到避免重复生成的能力，于是便在损失函数中添加了Coverage Loss，以强迫模型避免重复生成。公式如下：



$$

\operatorname{covloss}\_{t}=\sum\_{i} \min \left(a\_{i}^{t}, c\_{i}^{t}\right)

$$



$$

\operatorname{loss}\_{t}=-\log P\left(w\_{t}^{\*}\right)+\lambda \sum\_{i} \min \left(a\_{i}^{t}, c\_{i}^{t}\right)

$$

图18：加入Coverage Loss

1. 模型结构图

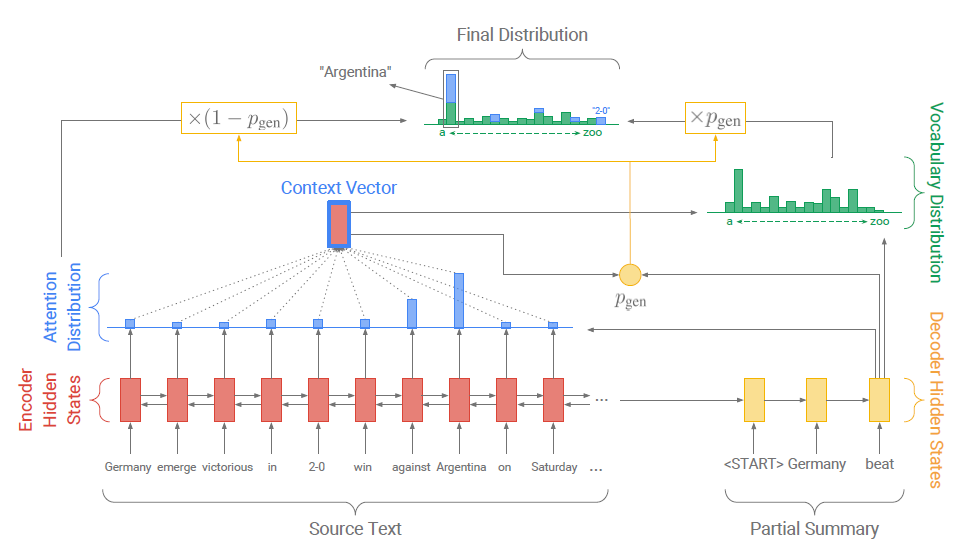


图19：Pointer-Generator Network模型结构

1. 实验结果

在训练模型方面，Pointer-Generator Network只需要比Sequence to Sequence + Attention 模型多一点点的参数，便可以达到更快更好的结果。详情如下：

|  |  |  |
| --- | --- | --- |
|  | Pointer-Generator Network | Sequence to Sequence + Attention |
| 参数 | 21,501,265 | 21,499,600 |
| 迭代次数 | 233,000 | 600,000 |
| 词表 | 50k | 150k |
| 训练时间 | 3 days 4 hours | 8 days 21 hours |

表1：模型与Baseline的基本参数对比

在实验结果方面，Pointer-Generator Network达到了当时state-of-the-art水平，可见其提出的Copy Mechanism和Coverage Mechanism的有效性。

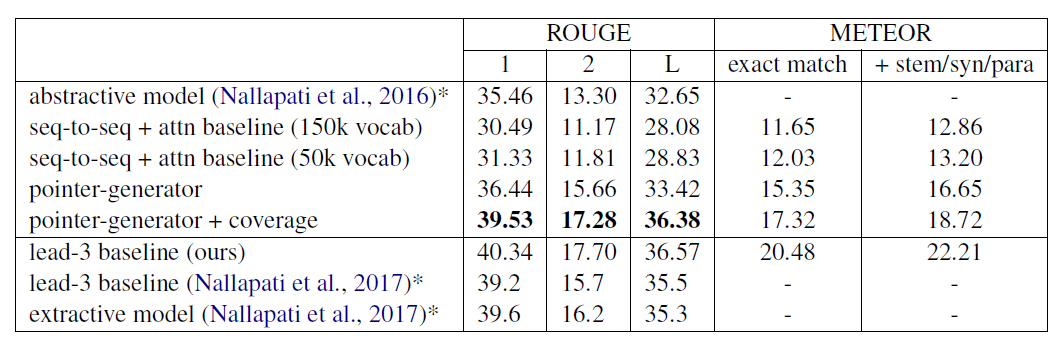


图20：实验结果[10]: Pointer-Generator + Coverage获得最好的ROUGE值。

1. **业务应用**
2. **小结**

本文说明了摘要模型的两种主流方向，并介绍了在抽取式摘要和生成式摘要领域中的不同模型。其中TextRank是基于图结构，PacSum是基于BERT，而Pointer-Generator Network则是基于Sequence to Sequence，它们特点各有不同，也具有不同的表现。但可以发现，随着预训练模型的兴起，越来越多的摘要模型都会利用Transformer提升模型的表现。而随着生成式摘要的效果越来越好，甚至超越了抽取式摘要，模型重心也逐渐向生成式摘要偏移，因为生成式摘要更能生成出具有抽象表达的摘要，从而达到最后可以代替人工摘要的目的。

参考文献

[1] Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In International ACM SIGIR conference on Research and development

in information retrieval.

[2] Horacio Saggion and Thierry Poibeau. 2013. Automatic text summarization: Past, present and future. In Multi-source, Multilingual Information Extraction and Summarization, Springer, pages 3–21.

[3] Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In Empirical Methods in Natural

Language Processing.

[4] Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. 2016. Efficient summarization with read-again and copy mechanism. arXiv preprint arXiv:1611.03382.

[5] Chin Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram cooccurrence statistics. In Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for

Computational Linguistics, pages 71–78, Edmonton, Canada.

[6] Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In Proceedings of EMNLP 2004, pages 404–411, Barcelona, Spain.

[7] Sergey Brin and Michael Page. 1998. Anatomy of a large-scale hypertextual Web search engine. In Proceedings of the 7th Conference on World Wide Web, pages 107–117, Brisbane, Australia.

[8] Hao Zhang and Mirella Lapata. Sentence centrality revisited for unsupervised summarization, 2019.

[9] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

[10] Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Volume 1: Long

Papers, pages 1073–1083.

[11] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In Computational Natural Language Learning.

[12] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In International Conference on Learning Representations.

[13] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In Neural Information Processing Systems.