计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2023,32(1):224−232 [doi: 10.15888/j.cnki.csa.008858]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

# 基于指针生成网络的中文对话文本摘要模型①

胡清丰, 魏　赟, 邬春学

(上海理工大学 光电信息与计算机工程学院, 上海 200093)
通信作者: 魏　赟, E-mail: weiyun@usst.edu.cn

**摘　要**: 针对传统 Seq2Seq 序列模型在文本摘要任务中无法准确地提取到文本中的关键信息、无法处理单词表之外的单词等问题, 本文提出一种基于 Fastformer 的指针生成网络 (pointer generator network, PGN) 模型, 且该模型结合了抽取式和生成式两种文本摘要方法. 模型首先利用 Fastformer 模型高效的获取具有上下文信息的单词嵌入向量, 然后利用指针生成网络模型选择从源文本中复制单词或利用词汇表来生成新的摘要信息, 以解决文本摘要任务中常出现的 OOV (out of vocabulary) 问题, 同时模型使用覆盖机制来追踪过去时间步的注意力分布, 动态的调整单词的重要性, 解决了重复词问题, 最后, 在解码阶段引入了 Beam Search 优化算法, 使得解码器能够获得更加准确的摘要结果. 实验在百度 AI Studio 中汽车大师所提供的汽车诊断对话数据集中进行, 结果表明本文提出的 Fastformer-PGN 模型在中文文本摘要任务中达到的效果要优于基准模型, 具有更好的效果.

**关键词**: 深度学习; 文本摘要; 指针生成网络 (PGN); 覆盖机制; Fastformer 模型

引用格式: 胡清丰,魏赟,邬春学.基于指针生成网络的中文对话文本摘要模型.计算机系统应用,2023,32(1):224–232. http://www.c-s-a.org.cn/1003-3254/8858.html

## Chinese Dialogue Text Summarization Model Based on Pointer Generator Network

HU Qing-Feng, WEI Yun, WU Chun-Xue

(School of Optoelectronic Information and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China)

**Abstract**: Considering the problems that the traditional Seq2Seq model cannot accurately extract key information from texts and process words outside the word list in text summarization tasks, this study proposes a pointer generator network (PGN) model based on Fastformer. The model combines the text summarization methods of extraction and generation. Specifically, the Fastformer model is used to efficiently obtain the word embedding vector with context information, and then PGN helps choose to copy words from the source text or use vocabulary to generate new summary information, so as to solve the out-of-vocabulary (OOV) problem that often occurs in text summarization tasks. At the same time, the model uses the coverage mechanism to track the attention distribution of the past time step and dynamically adjust the importance of words to solve the problem of repeated words. Finally, the Beam Search algorithm is introduced in the decoding stage to make the decoder obtain more accurate summary results. The experiments on the dataset of auto-diagnosis dialogues provided by Auto Master in AI Studio of Baidu show that the Fastformer-PGN model proposed in this study achieves better performance in text summarization tasks of Chinese dialogues than the benchmark model.

**Key words**: deep learning; text summarization; pointer generator network (PGN); coverage mechanism; Fastformer model

① 基金项目: 国家重点研发计划 (2018YFC0810204)
收稿时间: 2022-04-12; 修改时间: 2022-05-22; 采用时间: 2022-05-28; csa 在线出版时间: 2022-11-14
CNKI 网络首发时间: 2022-11-16

随着互联网的高速发展, 网络世界中的数据量也在逐步加大, 如何从海量的数据中准确地提取出我们所需要的信息是目前自然语言处理中非常重要的一项任务. 文本摘要任务是从文本信息中提取出最重要的信息的过程[1], 即通过一段简短的话术准确地表达出原文的核心观点.

文本摘要有两种策略: 抽取策略和生成策略. 抽取策略通过从源文本中选择能显著表达文本含义的单词来创建摘要, 而生成策略则是通过对句子进行重叙或重构来形成摘要. 相比于生成式文本摘要方法, 抽取式文本摘要方法更加简单, 因为从原文中抽取一部分单词或句子保证了最基本的语法正确和准确率. 而生成式摘要则需要考验模型对原文档充分的理解, 包括词语与词语还有句子与句子之间的关系的理解, 因此, 抽取式摘要是文摘摘要生成任务中的一项非常大的挑战.

近些年来由于深度学习技术的发展, 神经网络被应用到了越来越多的领域, 在自然语言处理任务中, 由神经网络搭建的 Sequence-to-Sequence 模型实现了较好的效果, 被广泛应用到了诸如机器翻译和语音识别等任务, 且在文本摘要任务中的应用也取得了显著的效果.

本文基于 Fastformer 和指针生成网络 (pointer generator network, PGN) 提出了一种面向新闻文本的自动生成式文本摘要模型——Fastformer-PGN 模型. 本论文的创新之处在于: 1) 使用 Fastformer 线性复杂度下的注意力计算来实现高效的上下文语义建模, 使 PGN 模型获取更好的单词嵌入向量. 2) 使用指针生成网络将抽取式方法和生成式方法进行融合, 使模型具有从源文档中复制文本的能力. 3) 引入 coverage 机制来降低摘要中重复词的出现次数. 实验在百度 AI Studio 比赛中汽车大师提供的 8 万多条技师与用户的汽车诊断对话文本上进行训练与验证, 通过与其他几个模型进行对比实验, 实验表明本文提出的模型在 *ROUGE*-1、*ROUGE*-2 和 *ROUGE-L* 指标上都获得了提升.

## 1 相关工作

过去一段时间内, 由于生成式摘要任务的困难性较大, 因此大部分的工作都是针对于解决抽取式文本任务, 在 20 世纪 90 年代, 随着机器学习技术的发展, 其在自然语言处理任务中也得到了广泛的应用, 其中一些研究通过使用统计技术来从目标文本中获取所需摘要. 1958 年 Luhn[2] 提出了根据句子中实词的个数来计算句子的权值, 提出了第一个自动文本摘要系统. Kupiec 等[3] 提出将朴素贝叶斯技术应用在文本摘要生成任务之中, Paice[4] 提出根据各种指示性短语 (例如 in this paper, the purpose of this paper) 来选择摘要句子的方法. 然而, 这些传统的机器学习方法在文本摘要的准确性上表现得不尽人意.

Sutskever 等[5] 在 2014 年提出了极具影响的序列模型 Sequence-to-Sequence, 由于递归神经网络 (RNNs) 具有能够自由生成文本的特征, 这使得解决抽象式摘要任务的性能得到了很大提升, 随后 Chopra 等[6] 提出了一种基于注意力机制的以 RNN 作为编码器和解码器的 Seq2Seq 模型, 更够更好地提取到上下文中的信息. 同时 Nallapati 等[7] 提出了关键字建模方法, 以捕捉句子或单词结构的层次结构. 尽管这些研究都在当时获得了比较好的效果, 但是他们都有着不可避免的缺点, 比如对于原文的理解不够精准, 或者无法解决摘要任务中的未登录词 (out-of-vocabulary, OVV) 问题.

See 等[8] 在 2017 年基于指针网络 (pointer network)[9] 提出了指针生成网络模型 (pointer generator network, PGN), 构造了一个新的架构来解决以上提到的问题, PGN 模型具有从源文本中复制词语的能力, 因此提高了文本提取的准确度并且还能克服之前提到的 OOV 问题, 同时, 该模型还保留了生成单词的能力. PGN 模型可以看作是一种融合了提取式摘要和抽象式摘要的模型, 这方面和 2016 年 Gu 等[10] 提出的 CopyNet 相似. 模型还引入了覆盖 (coverage) 机制, 通过覆盖机制, 系统能追踪到过去时间步中对哪些词语赋予了较多的注意力, 转而能对被赋予较少注意力的词语较多的关注, 因此能有效地解决文本摘要生成中的重复词问题.

2017 年, 来自谷歌实验室的 Vaswani 等[11] 提出的 Transformer 模型, 完全摒弃了 RNN 和 CNN 等网络结构, 仅采用 Attention 机制来解决机器翻译任务, 并取得了显著的效果, 自此以来, 注意力机制变成了相关领域研究者们的研究热点. Transformer 模型和之后研究者们提出的各类变体在相当多的领域都获得了极大的成功, 之后提出的 BERT[12] 和 GPT[13] 等预训练模型, 在各种任务中都达到了显著的效果. Transformer 的核心是自注意力机制, 它允许模型对整个输入序列进行上下文环境建模, 但是 Transformer 需要对每个位置的单词都进行点乘操作, 导致其运算具有较大的复杂度[11],

当输入一段较长的文本时, 模型性能就会大幅降低[14].

近些年来, 为了提升 Transformer 自注意力机制运算的效率, 研究者们提出了许多新的方法. 一些方法采用稀疏注意机制来降低自我注意力计算过程的复杂度, 例如 Longformer[15] 使用一个滑动窗口注意力机制来获取上下文信息. Big bird[16] 将局部注意力和特定位置的全局注意力进行融合. 然而这些方法往往需要生成较多的标识符, 因而只能达到有限的性能提升. 2021 年 Wu 等[17] 提出了一种 Transformer 的变体 Fastformer, 该模型基于加性注意力机制 (addictive attention), 实现了线性复杂度下的高效上下文建模, 在该模型中, 作者针对 Transformer 模型在计算注意力时不可避免地产生二次复杂度的问题, 提出了采用元素级乘积的方法, 在保证实现良好的上下文建模的情况下, 将模型计算的复杂度有效降低.

传统指针生成网络由于使用了具有时序性特点的 RNN 结构作为编码器, 无法有效地对文本上下文进行建模, 本文提出的模型通过使用 Fastformer 来对文本进行预编码, 在实现线性复杂度下的注意力计算的同时高效地对上下文文本进行建模, 使文本摘要的结果更加准确.

## 2 模型算法设计

本文提出的 Fastformer-PGN 模型通过以下两个阶段构建: 第 1 阶段为 Fastformer 预编码阶段, 输入的文本经过 Fastformer 模型进行高效的特征提取, 然后将结果与输入进行残差连接后作为第 2 阶段的输入. 第 2 阶段为文本摘要生成阶段, 将上一阶段的结果输入到指针生成网络中, 结合生成式和抽象式摘要方法, 使模型具有从源文本中复制单词的能力, 同时也具有生成新单词的能力, 有效解决了未登录词 (OOV) 的问题, 并引入 coverage 机制解决重复单词问题. 模型的总体架构如图 1 所示.

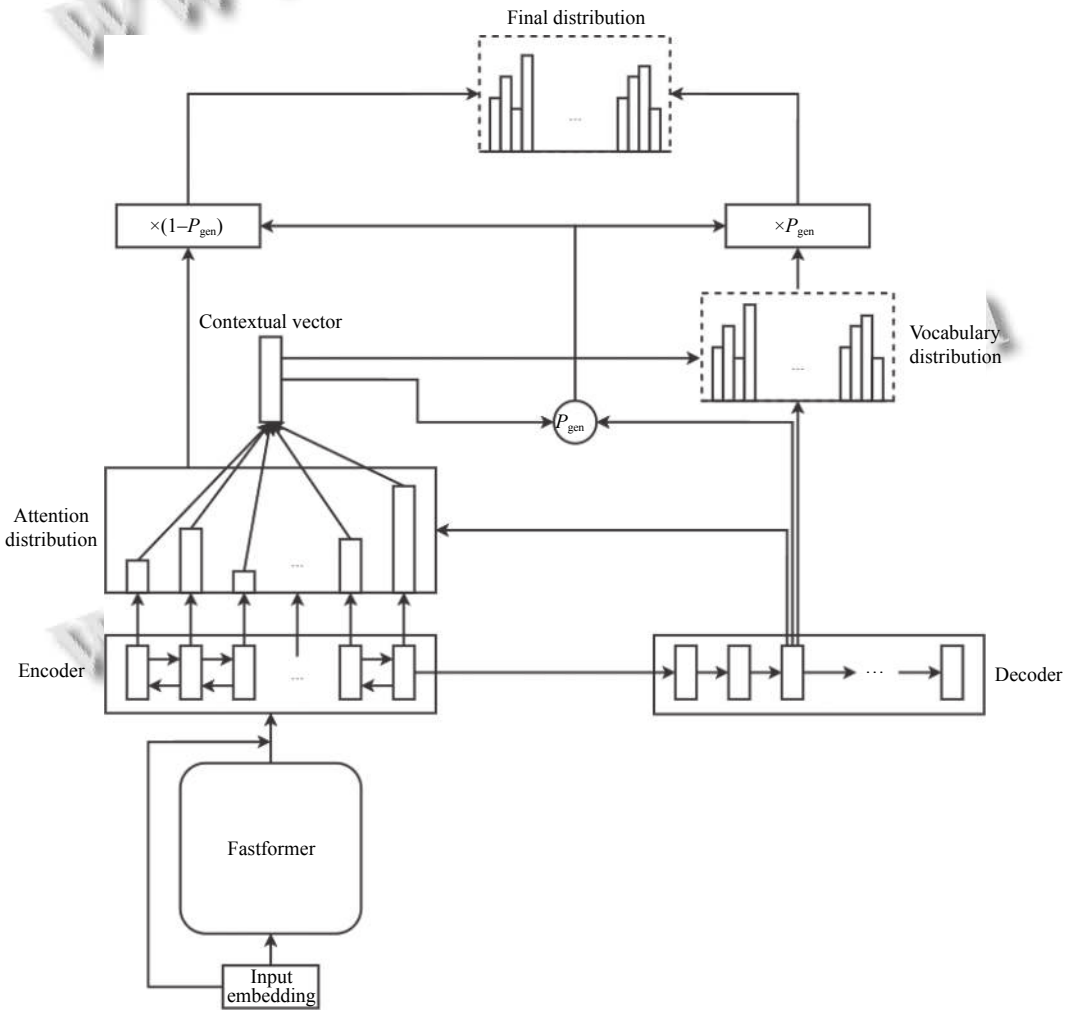


图 1 模型整体架构

### 2.1 基于 Fastformer 的预编码阶段

在本文提出的模型中, 为了对源文本进行充分的特征提取, 利用 Fastformer 作为指针生成网络模型的预编码器, 使模型能够更加充分的对上下文信息进行理解. 在传统的 Transformer 模型中, 注意力计算的复杂度为序列长度 $N$ 的二次方, 当序列很长时, 计算机会消耗大量的计算资源. Fastformer 是 Transformer 的一种变体, 采用元素级乘法使注意力计算的复杂度从序列长度 $N$ 的二次方降为 $N$, 从而使计算资源消耗大幅下降, 并能够有效地捕捉到文本的上下文信息.

Fastformer 的模型架构如图 2 所示. 首先通过加性注意力机制将 query 序列转换成一条全局 query 向量, 将所得的 query 向量与 key 序列进行元素乘 (element-wise product) 操作, 然后再次利用加性注意力机制将序列转换成一条全局 key 向量, 将所得到的 key 向量与 value 序列进行元素乘操作, 再利用线性转化得到一个能代表全局信息的注意力序列, 最后将注意力序列与 query 序列相加得到输出结果.

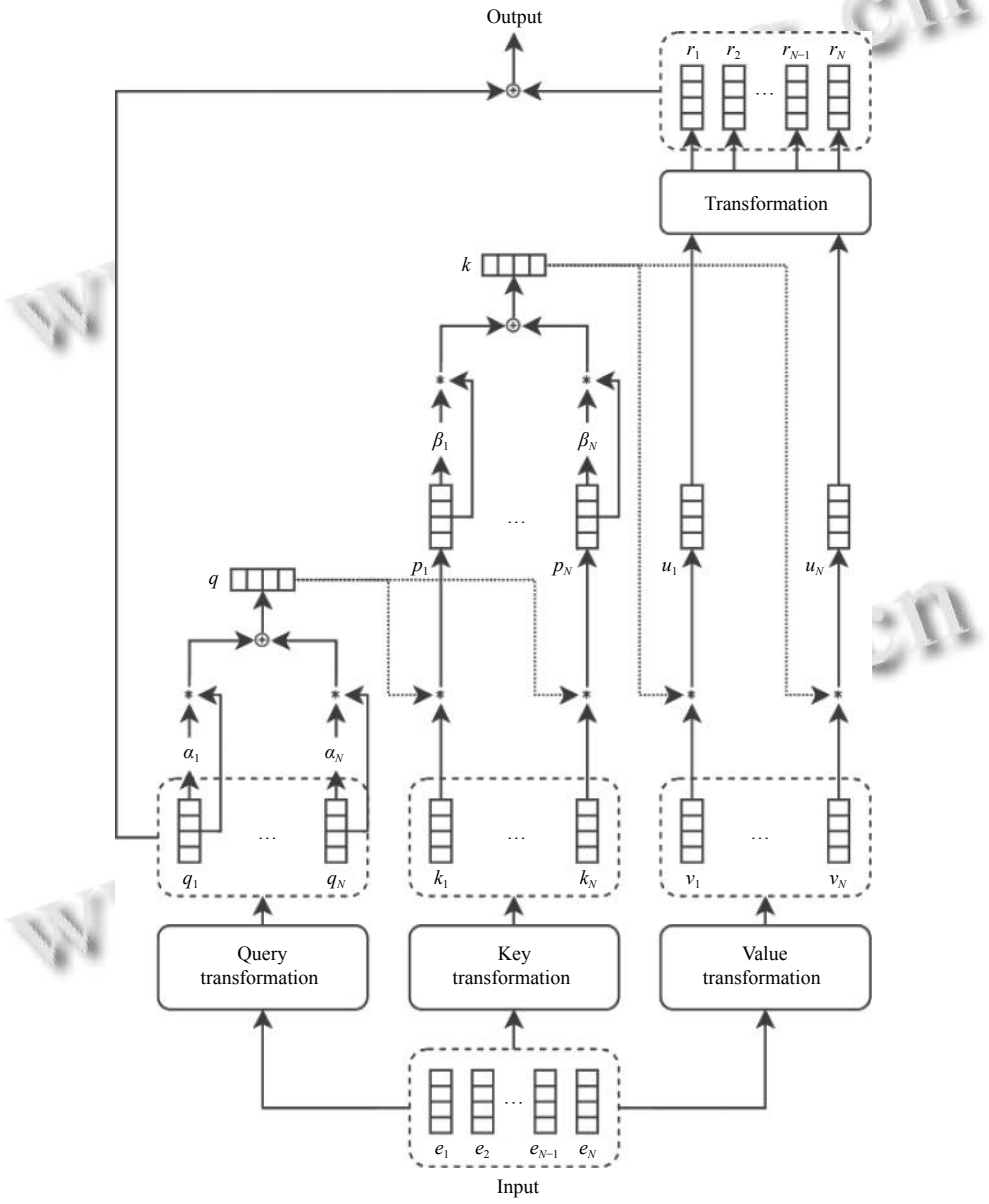


图 2 Fastformer 模型架构

将输入矩阵记为$E \in \mathbb{R}^{N\times d}$, 其中 $N$ 为输入序列长度, $d$ 为隐藏层维度, 输入矩阵的向量表示为$[e_1, e_2, \cdots, e_N]$. Fastformer 模型中采用多头注意力机制, 在每一个注意力头中利用线性层将输入矩阵转换成 3 个

注意力矩阵, 记为$Q=[q_1,q_2,\cdots,q_N]$, $K=[k_1,k_2,\cdots,k_N]$, $V=[v_1,v_2,\cdots,v_N]$.

Transformer[11] 中, 计算 $Q$, $K$, $V$ 之间的全局注意力使用的是点积操作, 点积操作所带来的二次复杂度将会导致长序列建模的性能大幅降低. 在 Fastformer 中, 为解决这个问题, 模型在计算 $Q$, $K$, $V$ 之间注意力前先对矩阵内部进行总结, 形成一条全局 $q$ 向量$q\in\mathbb{R}^d$, 将注意力 query 中的全局上下文信息融入到了一条向量之中. 第 $i$ 条 query 向量的注意力权重$\alpha_i$的计算公式为:

$$\alpha_i=\frac{\exp\left(W_q^{\mathrm{T}}q_i/\sqrt{d}\right)}{\sum_{j=1}^{N}\exp\left(W_q^{\mathrm{T}}q_j/\sqrt{d}\right)} \tag{1}$$

其中, $W_q\in\mathbb{R}^d$是一个可学习的参数向量. 全局 $q$ 向量的计算公式如下:

$$q=\sum_{i=1}^{N}\alpha_i q_i \tag{2}$$

接下来, 需要将得到的全局 $q$ 向量与 $K$ 矩阵之间进行交互建模, Fastformer 模型利用元素级乘积[18] 来进行计算, 元素级乘积是对两个向量之间建模的一种有效的方式. 模型通过以上方式得到一个包含全局上下文信息的 key 矩阵, 并将矩阵中的每一条向量记为 $p_i$, $p_i$的计算方式为$p_i=q*k_i$. 然后利用加性注意力机制将 key 矩阵总结为一条包含全局上下文信息的全局 $k$ 向量. 第 $i$ 条查询向量的注意力权重$\alpha_i$的计算公式如下:

$$\beta_i=\frac{\exp\left(W_k^{\mathrm{T}}p_i/\sqrt{d}\right)}{\sum_{j=1}^{N}\exp\left(W_k^{\mathrm{T}}p_j/\sqrt{d}\right)} \tag{3}$$

其中, $W_q\in\mathbb{R}^d$是一个可学习的参数向量. 全局 $k$ 向量的计算公式如下:

$$k=\sum_{i=1}^{N}\beta_i p_i \tag{4}$$

最后, 将得到的全局 $k$ 向量与 $V$ 矩阵之间进行交互建模. 模型使用与之前相似的方式, 将得到的 $k$ 向量与 $V$ 矩阵之间进行元素级乘法运算, $k$ 向量与 $V$ 矩阵中的每一条向量$v_i$之间的计算方式为$u_i=k*v_i$, 然后利用一个线性层去得到 key-value 之间的隐藏表示, 将得到的全局注意力矩阵记为$R=[r_1,r_2,\cdots,r_N]\in\mathbb{R}^{N\times d}$. 最后将 $R$ 与 $Q$ 注意力矩阵相加得到最终的输出.

接下来将得到的输出和输入进行残差连接, 作为下一阶段指针生成网络模型的输入.

### 2.2 基于指针生成网络的摘要生成阶段

在这个阶段, 我们将上一部分 Fastformer 计算得到的预编码结果传入指针生成多网络中的编码器 (单层双向 LSTM), 生成一个编码器隐藏状态$h_i$序列. 在每一个时间步 $t$ 上, 解码器 (单层双向 LSTM) 接收之前单词的单词嵌入, 得到时间步 $t$ 的隐藏状态$s_t$, 通过式 (5) 和式 (6) 得到注意力分布:

$$e_i^t=v^t\tanh\left(W_h h_i+W_s s_t+b_{\mathrm{attn}}\right) \tag{5}$$

$$a_t=Softmax\left(e^t\right) \tag{6}$$

其中, $v^t$、$W_h$、$W_s$、$b_{\mathrm{attn}}$ 为可学习参数. 接下来将所得到的注意力分布与编码器的隐藏状态进行加权求和, 得到上下文向量$h_t^*$:

$$h_t^*=\sum\nolimits_i a_i^t h_i \tag{7}$$

将$h_t^*$与$s_t$拼接后输入到两个线性层得到词汇分布 $P_{\mathrm{vocab}}$:

$$P_{\mathrm{vocab}}=Softmax\left(V'\left(V\left[s_t,h_t^*\right]+b\right)+b'\right) \tag{8}$$

其中, $V'$、$V$、$b'$、$b$为可学习参数. $P_{\mathrm{vocab}}$为词汇表中所有单词的概率分布.

指针生成网络为了解决 OOV单词问题, 允许模型通过直接复制源文本中的单词, 同时, 模型还具备从词汇表中选择单词生成摘要的功能. 对于每一个时间步 $t$, 生成摘要的概率$P_{\mathrm{gen}}$ 通过上下文向量$h_t^*$、解码器状态$s_t$和解码器输入$x_t$三者得到:

$$P_{\mathrm{gen}}=\sigma\left(w_{h^*}^{\mathrm{T}}h_t^*+w_s^{\mathrm{T}}s_t+w_x^{\mathrm{T}}x_t+b_{ptr}\right) \tag{9}$$

其中, $w_h^{\mathrm{T}}$、$w_s^{\mathrm{T}}$、$w_x^{\mathrm{T}}$、$b_{ptr}$为可学习参数, $\sigma$ 为激活函数 *Sigmoid* 函数. 接下来, $P_{\mathrm{gen}}$将作为一个软开关根据词汇概率$P_{\mathrm{vocab}}$来决定是从词汇表中生成单词还是根据注意力分布$a_t$从源文本中复制单词. 对于最终的词汇概率分布, 我们用式 (10) 来表示:

$$P(w)=P_{\mathrm{gen}}P_{\mathrm{vocab}}(w)+\left(1-P_{\mathrm{gen}}\right)\sum\nolimits_{i:w_i=w}a_i^t \tag{10}$$

值得注意的是, 在这个公式当中, 如果$w$是一个词汇外 (OOV) 单词, $P_{\mathrm{vocab}}$将等于零, 同样, 如果 $w$ 没有出现在源文本中, 则$\sum_{i:w_i=w}a_i^t$将等于零. 在训练过程中, 时间步 $t$ 的损失值为目标单词$w_t^*$的负对数似然值:

$$loss_t=-\log P\left(w_t^*\right) \tag{11}$$

整个序列的整体损失值为:

$$loss = \frac{1}{T}\sum_{t=0}^{T} loss_t \tag{12}$$

### 2.3 覆盖机制

针对文本摘要任务中常出现的重复词问题, 本文在 PGN 模型中引入覆盖机制 (coverage mechanism) 去惩罚重复的位置, 覆盖模型使用解码器过去所有时间步的注意力总和来定义一个覆盖向量$c^t$:

$$c^t = \sum_{t'=0}^{t-1} a^{t'} \tag{13}$$

其中, $c^t$表示到 $t$ 时刻 (当 $t$=0 时, $c$为零向量) 为止这些单词从注意力机制中获取的覆盖程度, 将覆盖向量作为注意力机制的附加输入, 将式 (13) 替换为:

$$e_i^t = v^{\mathrm{t}} \tanh\left(W_h h_i + W_s s_t + w_c c_i^t + b_{\mathrm{attn}}\right) \tag{14}$$

$w_c$是一个与$v$具有相同长度的可学习参数向量, 这种方式确保了当前的注意力决定将基于之前的决定, 使得注意力机制避免重复关注相同的位置, 以此避免产生相同的内容.

本文定义一个覆盖损失, 当关注到相同位置时, 将对其给予惩罚:

$$cov_{loss_t} = \sum_i \min\left(a_i^t, c_i^t\right) \tag{15}$$

其中, $a_i^t$是$t$时刻的注意力分布. 在$cov_{loss}$中加入超参数 $\lambda$, 最终的损失函数为:

$$loss_t = -\log P(w_t^*) + \lambda \sum_i \min\left(a_i^t, c_i^t\right) \tag{16}$$

## 3 实验分析

### 3.1 实验环境

本文实验在 Windows 10 操作系统上进行, 所用编程语言为 Python 3.7, 使用 PyTorch 作为深度学习框架, GPU 为 NVIDIA Geforce RTX1080ti, 使用 CUDA 进行 GPU 加速.

### 3.2 数据集

实验所用到的数据集来自 Baidu AI Studio 中汽车大师所提供的 82 943 条记录, 文本基于对话、用户问题、车型与车系, 输出包含摘要与推断的报告文本, 实验将数据集以 8:1:1 的比例划分为训练集、验证集和测试集.

### 3.3 数据预处理

由于数据集中存在大量脏数据, 首先需要进行一系列的预处理操作将数据集中的无意义词、空值、停用词和特殊符号等进行清理, 预处理过程中使用 jieba 分词工具对句子进行中文分词, 预处理前后的数据样例如表 1 所示.

表 1 预处理前后的数据样例

| 原汽车诊断对话 | 预处理后的对话 |
|---|---|
| 奔驰, 奔驰GL级, 方向机重, 助力泵, 方向机都换了还是一样, 技师说:[语音]\|车主说: 新的都换了\|车主说: 助力泵, 方向机\|技师说:[语音]\|车主说: 换了方向机带的有\|车主说:[图片]\|技师说:[语音]\|车主说: 有助力就是重, 这车要匹配吧\|技师说: 不需要\|技师说: 你这是更换的部件有问题\|车主说: 跑快了还好点, 就倒车重的很.\|技师说: 是非常重吗\|车主说: 是的, 累人\|技师说:[语音]\|车主说: 我觉得也是, 可是车主是以前没这么重, 选吧助理泵换了不行, 又把放向机换了, 现在还这样就不知道咋和车主解释.\|技师说:[语音]\|技师说:[语音], 随时联系 | 奔驰, 奔驰级, 方向机重助力泵方向机都换了还是一样, 技师说车主说新的都换了车主说助力泵方向机技师说车主说换了方向机带的有车主说技师说车主说有助力就是重这车要匹配吧技师说不需要技师说你这是更换的部件有问题车主说跑快了还好点就倒车重的很技师说是非常重吗车主说是的累人车主说我觉得也是可是车主是以前没这么重选吧助理泵换了不行又把放向机换了现在还这样就不知道咋和车主解释技师说技师说, 随时联系 |

### 3.4 Beam Search 优化

在文本摘要任务的模型生成过程中, 每一个时间步的输出都基于之前时间步的结果, 即基于历史生成结果的一个条件概率, 为了得到一个完整的摘要结果, 需要通过解码将多个时间步的输出进行融合, 使得最终的序列的每一步的条件概率连乘起来获得最大值.

在使用较多的贪心解码中, 每一个时间步都是选取神经网络输出层的最大值作为模型的解码结果, 再将之前所有时间步的结果作为下一个时间步的输入来获取输出结果, 最终得到想要的解码序列. 贪心解码虽然简单易用, 但是解码过程中的每一个时间步所得到的仅是局部最优解, 抛弃了绝大多数的可能解, 所以在本文所提出的模型中, 使用 Beam Search 算法[19] 对解码过程进行优化.

Beam Search 算法在当前级别的状态下计算所有的可能性, 并按照递增的顺序对它们进行排序, 但仅保留一定数量的可能结果, 后按照这个结果对其进行扩展, 直到迭代完所有的结果后返回最高概率的结果.

算法中定义一个超参数 beam size, 将其设为 $k$, 在每个时间步中, 我们按照概率进行排序, 选取前 $k$ 个概

率最大的词, 在下一个时间步中, 我们将当前的 $k$ 个词与之前的 $k$ 个词进行排列组合, 选择组合中概率最大的组合, 最终得到想要的摘要序列. Beam Search 算法即放宽了解码范围, 使解码器能够考虑到全局最优解. Beam Search 算法的伪代码如算法 1 所示.

算法1. Beam Search

```
数据: Graph (G), start word (w), goal word (g), beam size (k); 结果: 概率最大的摘要结果; 算法: beamSearch(G, w, g, k)
openList ← w
closedList ← empty list
path ← emptylist
while openList is not empty do
  b ← best word from openList
  openList.remove(b)
  closedList.add(b)
  if b is g then
    path.add(b)
    return path
  end
  N ← neighbors(b)
  for w in N do
    if w is in neither closedList nor openList then
      openList.add(w)
    else if w is in openList then
      if current parent path ← old parent path then
        Replace parents of w
      end
    end
    if number of words in openList > k then
      openList ← best k words in openList
    end
  end
  return path
end
```

### 3.5 训练

在模型训练过程中, 首先通过 Fastformer 模型对输入文本进行上下文特征提取, 其后将得到的输出进行残差连接输入给 PGN 模型, PGN 模型的词向量嵌入维度为 512, 隐藏层维度为 256. PGN 编码器使用单层双向 LSTM 网络, 实验采用 Adam 算法进行优化, 初始学习率为 0.001.

将单词词汇表大小设置为 30 000, 并限制源文本中最大词汇数量为 300, 生成摘要的最大词汇数量被限制为 50. Fastformer-PGN 模型的主要参数如表 2 所示.

采用以上参数对本文模型进行训练, 得到的 *loss* 如图 3 所示, 本文模型采用 Fastformer 作为预编码器, 使用 Beam Search 解码算法强化模型解码能力, 由于 Beam Search 算法维护了一个大小为 $k$ 的窗口, 所以前期 *loss* 降低速率较慢, 但最终本文模型的 *loss* 相较于 PGN 要小.

表 2 模型主要参数

| 名称 | 数值 |
|---|---|
| Embedding size | 512 |
| Hidden state | 256 |
| Learning rate | 0.001 |
| Optimizer | Adam |
| Batch size | 32 |
| Epoch | 10 |

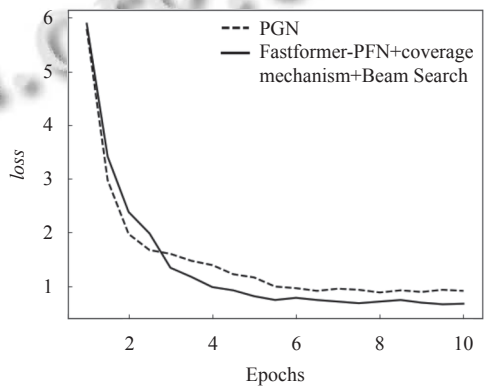


图 3 模型训练 *loss* 曲线

### 3.6 实验评价指标

实验采用 *ROUGE* (recall-oriented understudy for gisting evaluation)[20] 中的 *ROUGE-N* 和 *ROUGE-L* 作为评估标准, 其中 *ROUGE-N* 的计算公式为:

$$ROUGE\text{-}N=\frac{\sum_{S\in Ref}\sum_{gram_n\in S}Count_{match}(gram_n)}{\sum_{S\in Ref}\sum_{gram_n\in S}Count(gram_n)} \tag{17}$$

其中, 分母为统计在人工摘要中 *n-gram* 的数量, 分子则是统计模型摘要和人工摘要中共同的 *n-gram* 数量.

*ROUGE-L* 指标比较人工摘要和模型摘要的最长公共子序列, 能够衡量摘要的流畅程度.

### 3.7 实验结果

为了验证本文模型的有效性, 本文使用 3 种模型与本文提出的 Fastformer-PGN 模型进行对比.

ABS: Rush 等[21] 于 2015 年提出的基于 Seq2Seq 模型, 并将注意力机制应用于模型之中的一种模型, 应用于自动文本摘要任务的生成式文本摘要方法之中, 在本文中, 将 ABS 模型作为生成式摘要的基准模型与本文模型进行对比.

TextRank: 该模型通过计算文本中每个句子的重要性, 对不同的句子进行全新的排序组合形成最终的摘要, 将该模型作为抽取式摘要的基准模型与本文模型进行对比.

PGN: 指针生成网络通过使用指针网络来使模型具备从源文本中复制单词的能力, 并将覆盖机制融入到模型之中, 追踪过去时间步中对单词的注意力分配, 避免产生重复单词.

上述模型与本文提出的 Fastformer-PGN 模型在数据集中的对比实验结果如表 3 所示.

基于上述 3 种模型做对比实验, 针对 *ROUGE* 值做对比分析, 由表 3 中的实验结果可以得到:

1) 本文提出的 Fastformer-PGN 模型在 *ROUGE* 值评估的各个指标上都达到了最好效果, *ROUGE*-1 达到了 38.37, *ROUGE*-2 达到了 13.42, *ROUGE-L* 达到了 30.43.

2) 通过 ABS 和 TextRank 的对比实验可以看出, 抽取式摘要模型 TextRank 比生成式摘要模型 ABS 取得了更好的效果, 证明本文所用数据集上摘要结果更多的来自源文本中曾出现的词语.

3) PGN 模型由于同时结合了生成式策略和抽取式策略的特点, 使得摘要生成的效果有了大幅提升, 而本文提出的 Fastformer-PGN 模型由于在 PGN 模型的基础上对输入文本做了预编码, 使得模型能够获取上下文信息, 以解决 PGN 模型使用 LSTM 而无法结合上下文语境的缺点.

4) 最后在 Fastformer-PGN 模型基础上引入 coverage 机制, 有效地解决了文本摘要问题中经常出现的重复词问题, 使模型效果达到了所以模型中最好的效果.

表 3 模型 *ROUGE* 值对比

| 模型 | *ROUGE*-1 | *ROUGE*-2 | *ROUGE-L* |
|---|---|---|---|
| ABS | 29.54 | 10.34 | 25.76 |
| TextRank | 32.27 | 11.31 | 25.87 |
| PGN | 36.12 | 12.82 | 29.64 |
| Fastformer-PGN | 36.08 | 12.96 | 29.36 |
| Fastformer-PGN+coverage | 38.37 | 13.42 | 31.43 |

### 3.8 实例分析

从测试集中抽取若干条数据, TextRank、PGN 还有 Fastformer-PGN 这 3 种模型所生成的文本摘要如表 4 所示.

表 4 摘要结果示例

| 文本内容 | TextRank | PGN | Fastformer-PGN |
|---|---|---|---|
| 奥迪, 奥迪A8, 13年a8L4.0t急加速达到4 000转左右就会亮epc灯 用电脑检测故障码为P029900 增压压力控制没有达到控制极限 请问一下像这种车型一般是什么问题, 技师说: 你好, 没有极限, 一个是漏气, 一个就是涡轮增压的问题, 还有事旁通阀, 电池阀这里, 还有就是泄压阀, 增压压力传感器\|车主说: 直接更换两个涡轮增压总成怎么样\|技师说: 不一定是涡轮增压的问题, 检查一下旁通阀跟泄压阀 | 增压压力传感器直接更换两个涡轮增压总成怎么样 不一定是涡轮增压的问题, 检查一下旁通阀跟泄压阀 用电脑检测故障码为P029900 增压压力控制没有达到控制极限, 还有事旁通阀, 一个就是涡轮增压的问题 | 增压控制, 没有达到压力极限, 涡轮增压问题, 检查泄压阀 | 涡轮增压问题, 控制检查旁通阀, 泄压阀 |
| 大众, 速腾, 今天不经意发现, 开直线时方向盘摆正车子往右跑, 要把方向盘向左偏点才能开直线.师傅帮看看是什么问题, 技师说: 你好, 这个得话先检查一下轮胎气压, 在检查一下地盘悬架有没有松动的, 没有松动的话就做一下四轮定位, 就可以了 | 在检查一下地盘悬架有没有松动的, 要把方向盘向左偏点才能开直线.开直线时方向盘摆正车子往右跑, 这个得话先检查一下轮胎气压 | 开直线时方向盘左偏, 检查轮胎四轮定位 | 开直线时方向盘摆正车子往右跑, 检查轮胎气压, 做四轮定位 |

从表 4 中不同模型所生成的摘要内容可以看出抽取式模型 TextRank 模型由于倾向于将出现频率较高的句子作为摘要的内容, 导致生成的摘要篇幅过长, 并不能体现出摘要的特点, 而本文提出的模型跟其他模型相比, 由于采用了 Fastformer 对文本进行了预编码, 并利用 PGN 模型良好的摘要生成能力, 生成的摘要更加贴合上下文地总结出汽车诊断对话中的核心内容.

## 4 结论与展望

本文提出了一种基于 Fastformer 和指针生成网络模型的中文文本摘要生成方法. 首先使用 Fastformer 对文本进行高效的上下文编码, 使模型能够更加充分地理解输入的文本, 并采用 PGN 模型使得模型在能够自动生成摘要的情况下还具备从源文本中复制单词的能力, 同时引入覆盖机制防止生成重复内容. 实验在百度 AI Studio 汽车大师数据集上进行了训练和测试, 结果表明, 该模型可以生成更加贴合源文本含义的文本摘要和取得更高的 *ROUGE* 评分. 后续工作将尝试引入更多的特征因素, 简化模型, 减少模型训练所需的时间.

## 参考文献


1 Carreras X, Màrquez L. Introduction to the CoNLL-2004 shared task: Semantic role labeling. Proceedings of the 8th Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004. Boston: Association for Computational Linguistics, 2004. 89–97.

2 Luhn HP. The automatic creation of literature abstracts. IBM Journal of Research and Development, 1958, 2(2): 159–165. [doi: 10.1147/rd.22.0159]

3 Kupiec J, Pedersen J, Chen F. A trainable document summarizer. Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Seattle: ACM, 1995. 68–73.

4 Paice CD. Constructing literature abstracts by computer: Techniques and prospects. Information Processing & Management, 1990, 26(1): 171–186.

5 Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. Proceedings of the 27th International Conference on Neural Information Processing Systems. Montreal: MIT Press, 2014. 3104–3112.

6 Chopra S, Auli M, Rush AM. Abstractive sentence summarization with attentive recurrent neural networks. Proceedings of 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. San Diego: Association for Computational Linguistics, 2016. 93–98.

7 Nallapati R, Zhou BW, dos Santos C, *et al*. Abstractive text summarization using sequence-to-sequence RNNs and beyond. Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning. Berlin: Association for Computational Linguistics, 2016. 280–290.

8 See A, Liu PJ, Manning CD. Get to the point: Summarization with pointer-generator networks. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. Vancouver: Association for Computational Linguistics, 2017. 1073–1083.

9 Vinyals O, Fortunato M, Jaitly N. Pointer networks. Proceedings of the 28th International Conference on Neural Information Processing Systems. Montreal: MIT Press, 2015. 2692–2700.

10 Gu JT, Lu ZD, Li H, *et al*. Incorporating copying mechanism in sequence-to-sequence learning. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. Berlin: Association for Computational Linguistics, 2016. 1631–1640.

11 Vaswani A, Shazeer N, Parmar N, *et al*. Attention is all you need. Proceedings of the 31st International Conference on Neural Information Processing Systems. Long Beach: Curran Associates Inc., 2017. 6000–6010.

12 Devlin J, Chang MW, Lee K, *et al*. BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1. Minneapolis: Association for Computational Linguistics, 2019. 4171–4186.

13 Radford A, Wu J, Child R, *et al*. Language models are unsupervised multitask learners. OpenAI Blog, 2019, 1(8): 9.

14 Tay Y, Dehghani M, Bahri D, *et al*. Efficient transformers: A survey. ACM Computing Surveys. 2022: 1–27. [doi: 10.1145/3530811]

15 Beltagy I, Peters ME, Cohan A. Longformer: The long-document transformer. arXiv:2004.05150, 2020.

16 Zaheer M, Guruganesh G, Dubey KA, *et al*. Big bird: Transformers for longer sequences. Advances in Neural Information Processing Systems, 2020, 33: 17283–17297.

17 Wu CH, Wu FZ, Qi T, *et al*. Fastformer: Additive attention can be all you need. arXiv:2108.09084, 2021.

18 Wang X, He XN, Nie LQ, *et al*. Item silk road: Recommending items from information domains to social users. Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. Shinjuku: ACM, 2017. 185–194.

19 Koehn P. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. Proceedings of the 6th Conference of the Association for Machine Translation in the Americas. Washington: Springer, 2004. 115–124.

20 Ng JP, Abrecht V. Better summarization evaluation with word embeddings for ROUGE. Proceedings of 2015 Conference on Empirical Methods in Natural Language Processing. Lisbon: Association for Computational Linguistics, 2015. 1925–1930.

21 Rush AM, Chopra S, Weston J. A neural attention model for abstractive sentence summarization. Proceedings of 2015 Conference on Empirical Methods in Natural Language Processing. Lisbon: Association for Computational Linguistics, 2015. 379–389.


(校对责编: 孙君艳)