《自然语言处理》
期末大作业--客服通话文本摘要提取

姓　　名：贾涵泽　阮心仪　郑奕霏

班　　级：2020211601　2020211602

学　　号：2021211072　2020211118　2020211151

指导教师：李荣锋

数字媒体与设计艺术学院

2023 年 6 月 12 日

目录

1 背景与简述

1.1 任务背景

客服中心每天都需要接通大量的客户来电，客户来电需要进行语音转文本，同时对文本进行概括，提取客户核心诉求，但是人工总结会增加客服工作量，降低工作效率。题目提供了用于训练深度神经网络的数据，并且希望我们使用监督学习的 AI 算法进行自动文本摘要任务。

1.2 实验简述

文本摘要是对**输入**长文本内容进行概括从而**输出**精炼的核心内容的任务 [1]，对文本或者是文本集合，抽取、总结或是精炼其中的要点信息，本质上是一种基于数据的文本生成任务。

本任务的语言文本来自于客服通话，题目提供的数据集包含了文本摘要的 Ground Truth 希望我们通过监督学习的方法来解决摘要问题。在前期调研阶段，我们对于任务内容进行分析，得出了该任务中存在以下几个难点：文本和摘要均过长、对话的前后文存在明显异质性和大量冗余信息不适合抽取式提取摘要、数据量少以及语音转文本种存在转换错误等问题。通过以上初步分析以及一部分实验，我们最终选择使用基于 transformer 神经网络的 PEGASUS[2] 模型[1]，通过对预训练模型进行微调的方法对该问题进行解决。我们通过提出一种迭代式的数据处理方法，解决了训练中因为 encoding 的 token 限制而导致无法完全读取文本的问题，同时优化了网络结构。微调时的网络输入为 train_data，并且按照 9∶1 拆分为 train 和 validation 数据，微调后使用 test_data 进行摘要输出，最终输出的结果在该任务的竞赛 A 榜上取得了 **1706** 分的成绩，在全部 988 个队伍中排名为 **19 名**。[2]

接下来我们将通过数据预处理、网络与训练、数据后处理、结果与问题这几个方面进行详细阐述与解释。

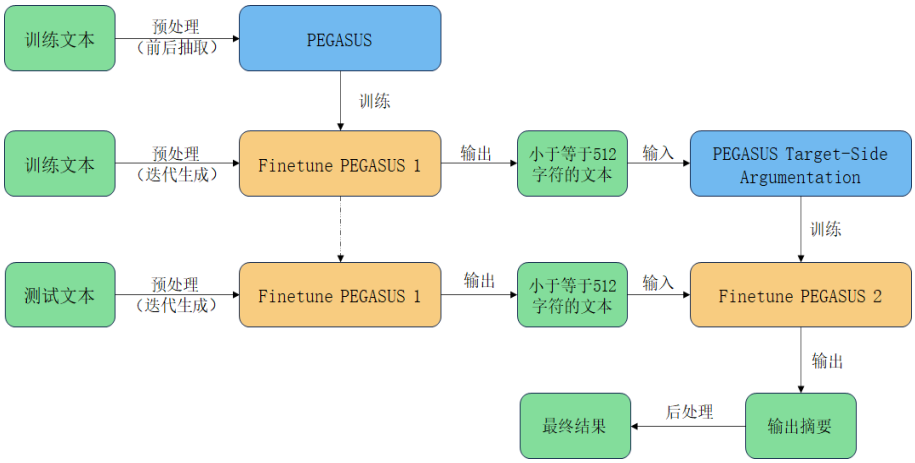


图 1: 任务流程图示

[1] https://github.com/PaddlePaddle/PEGASUS

[2] 客服通话文本摘要提取竞赛排行榜

2 数据预处理

2.1 数据分析

首先对任务提供的数据集进行数据分析，我们观察到所提供的数据集其中文本长度均值为 1215 个字符，有 90% 的数据长度在 2164 个字符以内。而摘要的均值为 105，有 90% 的数据长度在 163 个字符以内。

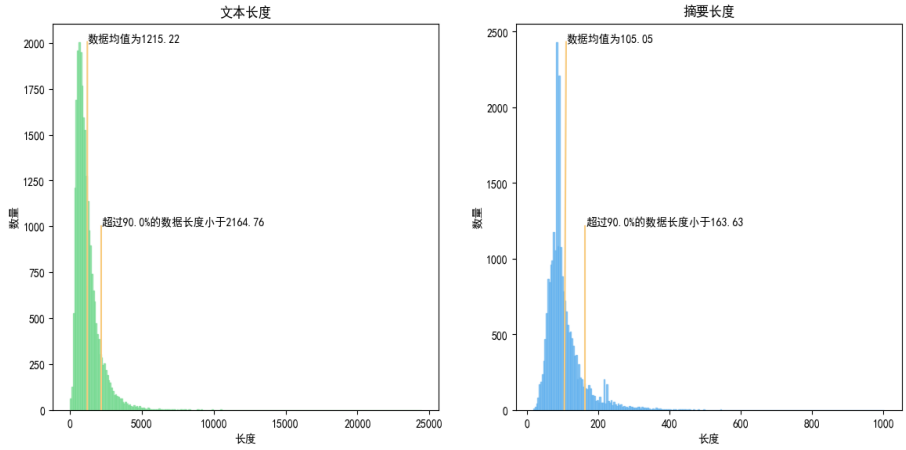


图 2: 文本与摘要长度分析

同时我们对于通话角色双方的内容进行了分析，可以看出：双方对话轮数均值为 35，坐席每轮对话的文本长度均值为 35，而客户为 32。

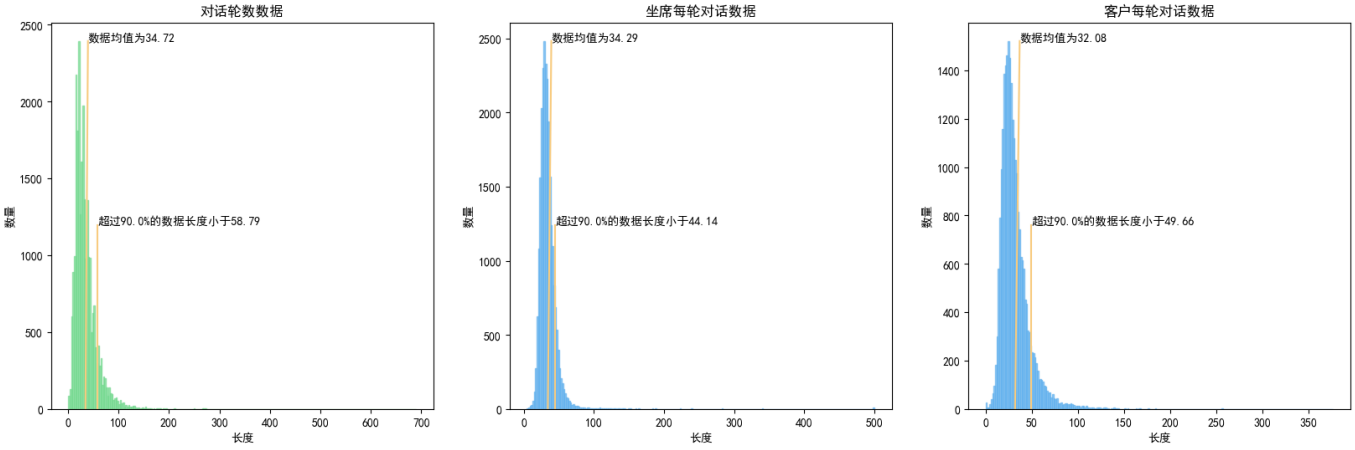


图 3: 坐席与客户对话论数与每轮对话长度分析

从以上两组数据分析可以知道给数据长度均存在过长现象，并且由于是口语文本，难免出现文本重复、冗余等现象，导致信息的冗余度增加。我们选用的模型参数为百度 Paddle 所提供的 PEGASUS 预训练模型，它的参数只支持输入最长 token 为 512，因此我们必须要缩减输入的数据长度。使其在尽可能保留原有信息的情况下，长度还要小于 512。

2.2 基于规则的冗余文本信息清理

根据通过直接分析文本内容，我们采用经验法直接对于对于一些冗余信息进行去除。

2.2.1 问候部分清理

由于通话开头会有超过两轮的问候对话，对于信息理解没有帮助，考虑清洗前几句问候的话语, 对于以【客户】开头的数据，存在 4 种语言情况：

- 标注错误。许多以【客户】开头的长句中，都包含了【坐席】的第一句话“您好，很高兴为您服务”等字眼，但这些内容没有被区分出来，而是被误放到了【客户】的语句中。这类语句的第一句长度一般大于 25，不需要去除，“您好”等错误混入的语句可以通过后面的关键词筛除步骤去除；第二句不需要去除。
- 用户直接抛出问题。用户没有等【坐席】说出基本的问候语就直接抛出问题，【坐席】紧接着进行询问和解决。这类语句的第一句长度一般大于 25，不需要去除；第二句不需要去除。
- 回访电话。回访电话是由通话公司打给用户的，因此第一句一般是用户的“喂”“你好”等回应。这类语句的第一句长度一般小于 25，需要去除；第二句说明主要问题，语句长度大于 20，不需要去除。
- 互相问候。【客户】和【坐席】彼此问候，语句都比较短，基本没有实质意义。第一句和第二句都需要去除。

基于上述 4 种语言情况，需要做的处理为：对于以【客户】开头的数据:

- 先判断第一句长度是否大于 25，是则不用去除任何内容；否则去除第一句，并进行下一步筛选。
- 判断第二句长度是否大于 20，是则不用去除；否则去除第二句。

2.2.2 停用词与替换词

考虑到虚词与语气助词的存在，我们构建了适合与该数据集的停用词表以及替换词表，同时对于标点符号等信息也进行了修正与替换。

2.2.3 去除过短的句子

根据经验观察，当处理到该步骤时，大部分字数少于 6 个汉字的语句都是无意义的语句，可以被直接去除。

2.2.4 更换前缀

由于原始每轮对话都带有前缀【坐席】【客户】进行对二者的区分，但是仍然占用了 4 个字符，我们需要减少字表示该区分的符数量。

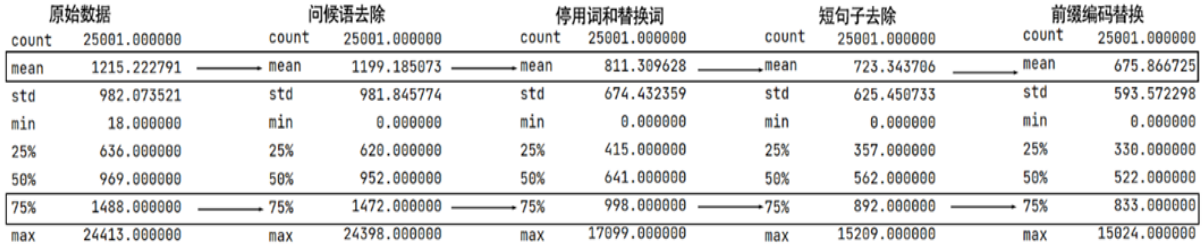


图 4: 基于规则的冗余文本信息清理过程

在经过以上处理后，注意到文本长度均值有所缩短。如图 3 所示，文本长度均值由 1215 缩短至 675，但是注意到，仍然有接近 50% 的数据字符长度大于 512，因此我们仍然需要进行进一步的文本缩短。

2.3 头尾保留法

通过文本分析不难得出，通话中的关键信息通常出现在头部于尾部，【客户】往往在一开始指出问题，【坐席】也会在结尾总结性的给出解决方案，因此比较直接简单的方式是直接取该文本的前 256 个字符与后 256 个字符组成新文本。通过这种方案我们训练得到了对应的模型，并且也取得了高于 baseline 的结果。

2.4 基于 TextRank 的文本缩短

由于文本缩短不同于生成式摘要提取，我们希望以无监督的方式来完成这一过程，因此我们尝试了使用 TextRank[3] 抽取式方法，希望通过对于对话中的句子的关键程度来抽取关键信息完成文本缩短。

TextRank 算法是由 PageRank 算法改进而来，通过构造句子节点，使用无向有权边进行连接，根据 TextRank 的相似度计算公式循环计算任意两个节点之间的相似度，设置阈值去掉两个节点之间相似度较低的边连接，构建出节点连接图，然后迭代计算每个节点的 TextRank 值，排序后选出 TextRank 值最高的几个节点对应的句子作为关键句。

在我们的方法中，我们还考虑了头尾句子关键程度信息，记文本 T 中第 i 个句子为 S_i，包含的句子为 l 个，句子 S_i 的 Rank 为 k_i，则有更新后第 i 个句子的 Rank 值为：

$$\hat{k_i} = \left((1-m)\cos(\frac{2\pi}{l}) + m\right) \times k_i \quad (0 \le m \le 1)$$

其中 m 为超参数，注意到当 $m = 1$ 时方程退化到 $\hat{k_i} = k_i$。

我们会按照更新后的 Rank 值由大到小抽取句子，直到已抽取句子的字符数刚好超过 512 为止，并按照原本的句子顺序恢复选出句子的顺序。

在实际实验中通过分析注意到，该方法虽然能够一定程度的根据词频等信息抽取出重要句子，但因为较为依赖词频影响，而客服通话为口语性质对话，往往一些冗余信息会多次出现，而关键信息出现次数不足，导致冗余句子被多次抽取，关键句子 Rank 值不够高而被忽略，因此方法仍需要改进。

2.5 迭代生成法

之后我们又考虑了使用迭代生成法，在 2.3 中我们已经训练得到了一个模型，因此我们考虑使用迭代生成的形式缩短文本。简而言之将文本每 512 个字符截断拆分一次（实际操作时考虑在完整句子处截断）。然后对每一段使用模型进行生成摘要，最终将结果进行拼接。

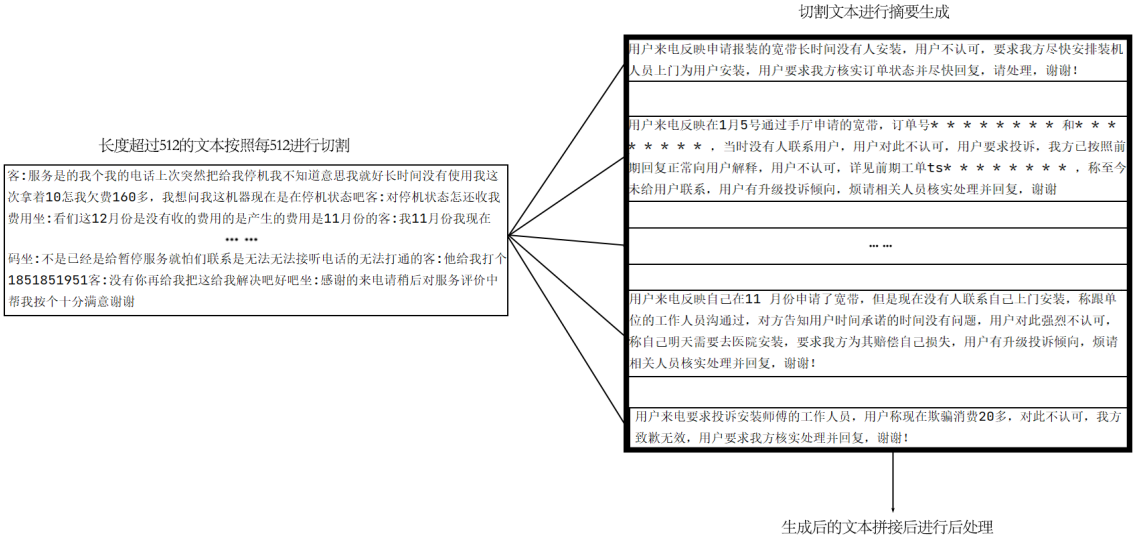


图 5: 简要的迭代生成过程

图 4 展示的为最简单形式的迭代生成，我们可以观察到，在每次迭代生成的文本均需要替换部分内容，因为生成摘要学习到的格式较为固定，比如开头都以“用户来电反映...”，结尾都以“请尽快处理... 谢谢”等固定格式，因此再拼接生成的文本时需要替换这些无用且固定的内容，因此我们这里仍然构造了停用词与替换词词典进行替换。在实际操作中，我们根据输入文本长度对实际算法进行了优化和改进，最终算法如算法 1 所示。

Algorithm 1: 迭代生成算法

Data: content,replace_dict, 函数 len(x) 求字符串 x 长度，函数 infer(x) 做字符串 x 的摘要

Result: 迭代生成提取后的 content

x← content
l← len(x)
if $x \leq 512$ **then**
    y ← x
else if $x \leq 600$ **then**
    y ← x 的前 256 和后 256 的字符拼接
else if $x \leq 2048$ **then**
    m ← x 去掉前 256 和后 256 个字符的中间部分
    n ← m 的前 256 和后 256 的字符拼接
    k ← len(infer(n))
    strNum ← $256 - k * 0.5$
    y ← 取 x 的前 strNum 个字符拼接字符串 n 再拼接 x 的后 strNum 个字符
else
    concatStr ← x
    while *len(concatStr)*≥ 2048 **do**
        m ← len(concatStr)
        cutNum ← $\frac{m}{512}$
        for *cutNum > 0* **do**
            tmp ← 依次取 x 的前 512 个字符串
            concatStr 拼接 infer(x)
            cutNum ← cutNum - 1
        end
    end
    y ← 取 concatStr 的前 256 和后 256 的字符拼接
end

Result: y

需要注意，在每次迭代生成的文本均需要替换部分内容，因为生成摘要学习到的格式较为固定，比如开头都以“用户来电反映...”，结尾都以“请尽快处理... 谢谢”等固定格式，因此再拼接生成的文本时需要替换这些无用且固定的内容，因此我们这里仍然构造了停用词与替换词词典进行替换。

2.6 对于无监督方法的评估

由于上述头尾保留法、基于 TextRank 的方法和迭代生成法均为无监督的对文本进行压缩，所以我们使用 Ground Truth 对其生成后的文本效果进行了评估。具体的评估方法使用了 ROUGE(Recall-Oriented Understudy for Gisting Evaluation)，通过 rouge-1、rouge-2 以及 rouge-l 的加权均值衡量三种方法输出与 Ground Truth 之间的距离。分数计算为 $score = rouge_1 \times 0.2 + rouge_2 \times 0.4 + rouge_l \times 0.4$ 结果如下图所示：

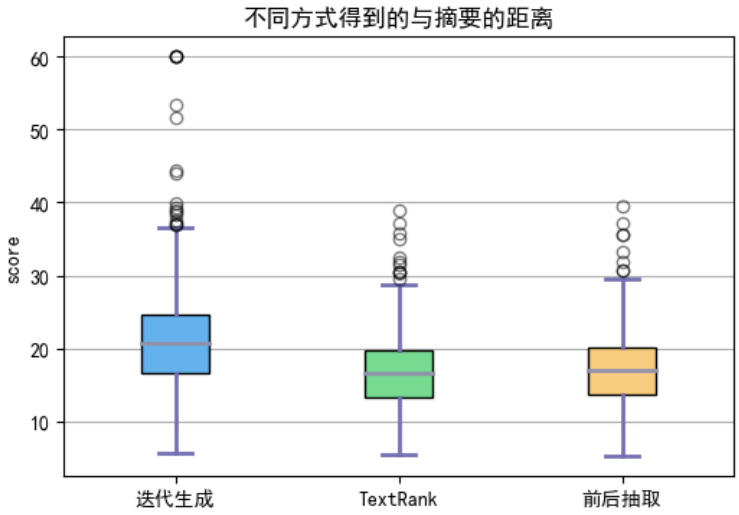


图 6: 三种方法与摘要的距离评分

可以看出迭代生成法得分的均值和整体分数最高，因此之后我们训练模型采用的方法最终确定为迭代生成法。

3 网络与训练

3.1 模型说明

PEGASUS 摘要生成模型是基于 T5-Transformer 模型改进得到的专门解决生成式摘要的预训练模型，该模型预训练的时候的目标是把生成间隙句子（GSG），所以在提取文本摘要的时候，简单的 f 微调对结果就有很大的提升，达到较好的结果。我们使用的模型是百度 Paddle 在中文数据集 LCSTS 上预训练好的模型，目前可以达到 LCSTS 数据的 SOTA 效果。

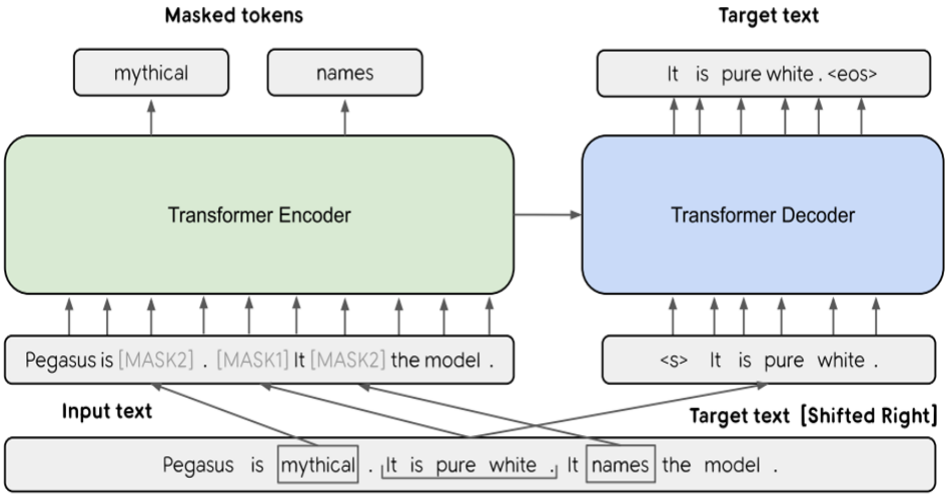


图 7: PEGSUS 模型结构

受限于运算资源限制我们选择的预训练模型参数大小如下所示：

参数	base(238M)
encoder/decoder ffn dim	3072
encoder/decoder layers	12
encoder/decoder attention heads	12
max encode length	512

3.2 target 端增强

网络模型改动是通过 target-side[4] 增强完成的，根据论文中所提出的 target 端增强的方法，即对于 encoder 端的输出根据一定比例混合 Ground Truth 的方法改进网络。其中对于 encoder 输出采用 softmax 处理，使用因子 γ 进行调配。最终输出结果为：

$$\hat{\mathbf{y}}_{\mathbf{j}}^{\mathbf{i}} = \left(\gamma \frac{\exp\left(l_j/T\right)}{\sum_{v=1}^{|v|} \exp\left(l_j[v]/T\right)} + (1-\gamma) y_j^i \right) \mathbf{W}_{\mathbf{Emb}}$$

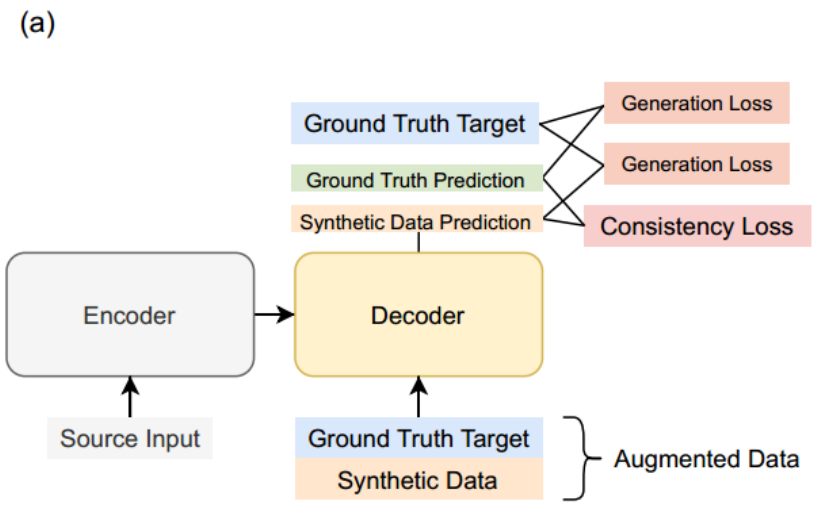


图 8: target-side 增强示意

3.3 微调预训练模型

微调预训练模型时，将训练集按照 9：1 划分为训练与验证两个部分，间隔一定步数验证一次。训练环境为：Intelő Xeon E5-2680 主频 2.4GHz ，GPU：NVIDIA Tesla P40，内存：DDR4，操作系统：Linux Ubuntu 18.04 LTS ，CUDA：11.2。

微调时使用的优化器为 Adamw，学习率采用了 warmup，batch 设置为 8，epochs 设置为 50，具体参数设置详见 train 部分代码。最终总共需要训练约 150000step，平均 0.9 step/s，在 120000step 附近收敛。

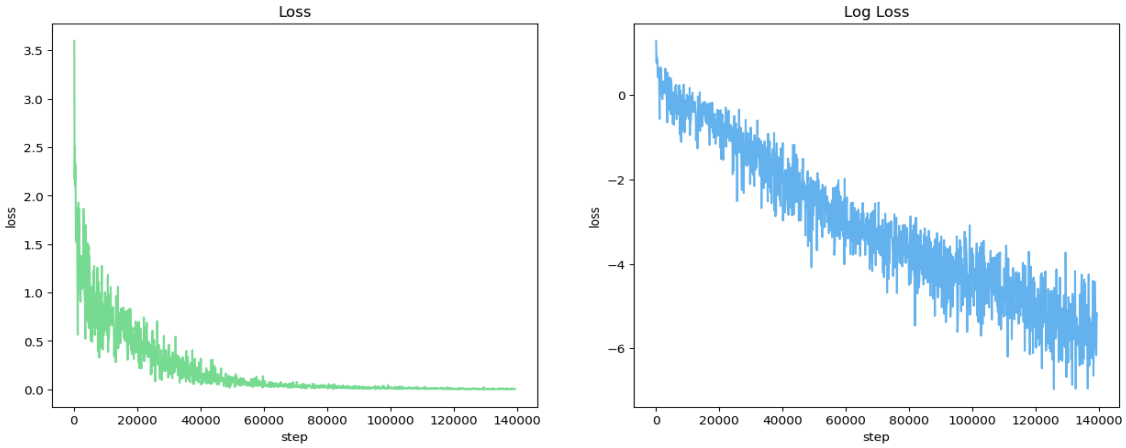


图 9: Loss 曲线

4 数据后处理

首先我们对只要的冗余信息进行了处理，对于使用 beam search 的方法，模型会在某一段连续序列中生成部分重复信息，针对这一问题我们进行了对应处理：

- 根据标点符号进行了切片
- 遍历切片对所有候选切片小于相似度阈值的切片加入到候选集中。
- 将候选集中的切片连接作为输出

其中相似度计算公式为：

$$s_{a,b} = rouge_1(a,b) \times 0.2 + rouge_2(a,b) \times 0.4 + rouge_l(a,b) \times 0.4$$

我们设定阈值为 0.75。

后续我们发现生成模型在部分标点符号上会存在 OOV 现象，有部分半角标点符号会成为未登录词出现，因此我们对于这些符号进行了后处理。最终经过以上处理，我们的结果大约有 3-5 分的涨幅。

5 结果展示

5.1 评价指标

与任务要求评价指标保持一致，选择 rouge-1、rouge-2 和 rouge-l 作为后处理前评价指标，结果经过后处理上传到 DataFoundation 官网进行打分，分数作为最终评价指标。

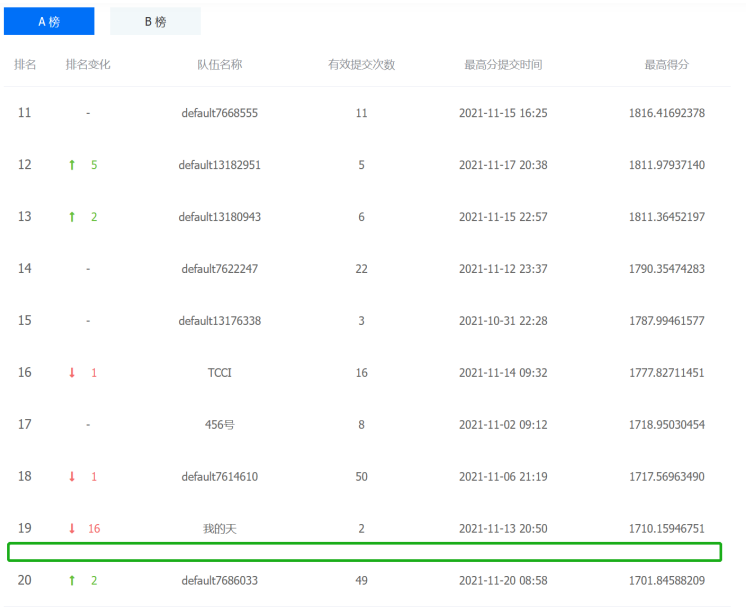
5.2 结果展示

结果如下所示，PEGASUS(LCSTS) 表示模型在 LCSTS 数据集上训练的结果，baseline 为对数据不进行任何预处理直接输入到 PEGASUS 中进行训练，可以看到分数达到了 1356，证明模型效果良好。可以看到迭代抽取 + 端增强法最终取得了最好的效果 rouge-1、rouge-2 和 rouge-l 分别提高了 12.24%、13.15% 和 10.89%，分数提高为 350 分。

	Rouge-1	**Rouge-2**	**Rouge-l**	**Score**
PEGASUS(LCSTS)	0.433	0.3008	0.4012	/
PEGASUS(baseline)	0.5358	0.3822	0.4747	1356
PEGASUS + 前后抽取法	0.5623	0.4027	0.5043	1523
PEGASUS + 前后抽取法 +target 端增强	0.5868	0.4179	0.5172	1593
PEGASUS + 迭代抽取法 +target 端增强	**0.6014**	**0.4373**	**0.5264**	**1706**

在有限算力的条件下，最终分数在竞赛排名[3]上取得了 A 榜 19/988 名的成绩

[3]客服通话文本摘要提取竞赛排行榜链接



A 榜　B 榜

排名	排名变化	队伍名称	有效提交次数	最高分提交时间	最高得分
11	-	default7668555	11	2021-11-15 16:25	1816.41692378
12	↑ 5	default13182951	5	2021-11-17 20:38	1811.97937140
13	↑ 2	default13180943	6	2021-11-15 22:57	1811.36452197
14	-	default7622247	22	2021-11-12 23:37	1790.35474283
15	-	default13176338	3	2021-10-31 22:28	1787.99461577
16	↓ 1	TCCI	16	2021-11-14 09:32	1777.82711451
17	-	456号	8	2021-11-02 09:12	1718.95030454
18	↓ 1	default7614610	50	2021-11-06 21:19	1717.56963490
19	↓ 16	我的天	2	2021-11-13 20:50	1710.15946751
20	↑ 2	default7686033	49	2021-11-20 08:58	1701.84588209

图 10: 排行榜名次

5.3 结果分析

首先能取得这样的结果很大程度来源于 PEGSUS 模型在摘要生成上的良好表现，可以说基于注意力机制的 transformer 模型在长文本信息提取上的效果远远超过了之前的任何算法模型。其次在整个实验过程中，我们小组成员也是第一次使用 transformer 结构的网络做微调，从中掌握了一些对于预训练模型微调的技巧以及方法，在数据预处理中我们尝试将传统方法结合或者通过特征观察总结规律，优化数据分布，最终结果上取得了提高。

当然使用大参数大模型的弊端就是算力限制较大，如果有想法想进行验证需要等待时间较长，并且训练上也没有太多技巧性可言，并且我们对于 transformer 的架构都算不上熟悉，很难去分析冻结哪些参数可能会有怎么样的变化或者对网络结构进行改动。

6 成员分工

成员	分工	贡献度
贾涵泽	数据预处理、模型训练、网络架构改动、报告撰写	40%
阮心仪	数据预处理、模型训练	30%
郑奕霏	数据预处理、模型训练、后处理	30%

References

[1] Narges Nazari and MA Mahdavi. “A survey on automatic text summarization”. In: *Journal of AI and Data Mining* 7.1 (2019), pp. 121–135.

[2] Jingqing Zhang et al. “Pegasus: Pre-training with extracted gap-sentences for abstractive summarization”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 11328–11339.

[3] Rada Mihalcea and Paul Tarau. “Textrank: Bringing order into text”. In: *Proceedings of the 2004 conference on empirical methods in natural language processing*. 2004, pp. 404–411.

[4] Shufang Xie et al. “Target-Side Input Augmentation for Sequence to Sequence Generation”. In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=pz1euXohm4H.