《使用预训练编码器的文本摘要》阅读报告

姓名 孙璐阳　　学号 PB21111685
姓名 赵卓　　学号 PB21111686
姓名 何昕　　学号 PB21111661

目录

1 背景介绍

近年来，预训练语言模型的出现大大提高了许多 NLP 任务的发展水平，包括情感分析、问题回答、自然语言推理、命名实体识别、文本相似性等等 NLP 任务。最先进的预训练模型包括 ELMo、GPT 以及来自 Transformer 的 BERT。在大多数情况下，这些预训练模型可以被用作句子级和段落级的自然语言理解问题的编码器，相关的问题比如预测任意的两个句子是否具有蕴含关系，比如在四个可选择的句子中选择出正确的完整句子。

然而预训练模型在文本摘要中的应用还未开发完备。这是因为文本摘要和以前的 NLP 任务相比，实现难度更大。文本摘要的目标是将文本压缩成较短的版本，同时还要保留其大部分的含义。这需要预训练模型不仅仅有对单个单词或者句子的理解能力，还要有覆盖全篇，词句相连的理解能力。具体来说，文本摘要又可以分为抽取式文本摘要和生成式文本摘要：抽取式文本摘要需要模型对原文本中的每个句子是否应该保留做出判断，最终留下的句子作为摘要；生成式文本摘要则需要模型具有语言生成能力，能够通过对原文本的理解分析，尽量不使用原文本出现过的句子生成新的文本摘要。

在这种情况下，本文探索了预训练模型 BERT 在文本摘要的应用。本文提出了一种基于 BERT 的新型文档级编码器，并基于该编码器为抽取式文本摘要和生成式文本摘要提出了一个通用的框架，三个数据集的实验结果对比表明，该框架在抽取式文本摘要和生成式文本摘要中都取得了全面领先的结果。

2 现有方法

2.1 抽取式摘要方法

现有抽取式摘要方法通过识别一个文档中最重要的句子然后连接生成摘要。神经模型是常用的模型之一，它将抽取式摘要看成是一个分类问题：由神经编码器为每个句子建立表示，然后由分类器决定哪个句子应该被选择留下作为摘要，SUMMARUNNER 是最早采用基于递归神经网络编码器的神经模型之一。此外，REFRESH 是一个基于强化学习的系统，通过全局优化 ROUGE 指标进行训练。近年来，越来越多的优秀工作通过更复杂的模型结构实现了更好的性能表现：LATENT 将抽取式摘要定义为一个潜在的变量推理问题，该模型直接最大化选定句子作为人类摘要的可能性，而不是最大化“黄金”标准标签的可能性。SUMO 利用结构化注意力的概念在预测输出摘要时引入文档的多根依赖书表示。NEUSUM 同时选择句子并评分，代表了抽取式摘要的最新技术。

2.2 生成式摘要方法

现有生成式摘要方法将摘要任务抽象为一个序列到序列的问题。编码器将源文档 $x = [x_1, \ldots, x_n]$ 中的标记序列映射到连续表示序列 $x = [z_1, \ldots, z_n]$，然后解码器以自回归的方式逐个生成摘要 $y = [y_1, \ldots, y_n]$，从而对条件概率进行建模：$p(y_1, \ldots, y_n | x_1, \ldots, x_n)$。Rush 和 Nallapati 等人是最早讲神经编码器-解码器模型用于生成式摘要的人之一。Abigail See 等人使用指针生成器网络（PT-GEN）增强此模型，该网络允许模型从源文本中复制单词，并使用覆盖机制（COV）跟踪已总结的单词。Celikyilmaz 提出了一个抽象系统，其中多个代理（编码器）表示文档并且带有用于解码的分层注意机制（在代理之上），他们的深度通信代理（DCA）模型通过强化学习进行端到端训练。Paulus 等人还提出了一个用于抽象摘要的深度强化模型（DRM），它通过内部注意机制处理覆盖问题，在这种机制中，解码器会处理先前生成的单词。Gehrmann 等人遵循自下而上的方法（BOTTOMUP），

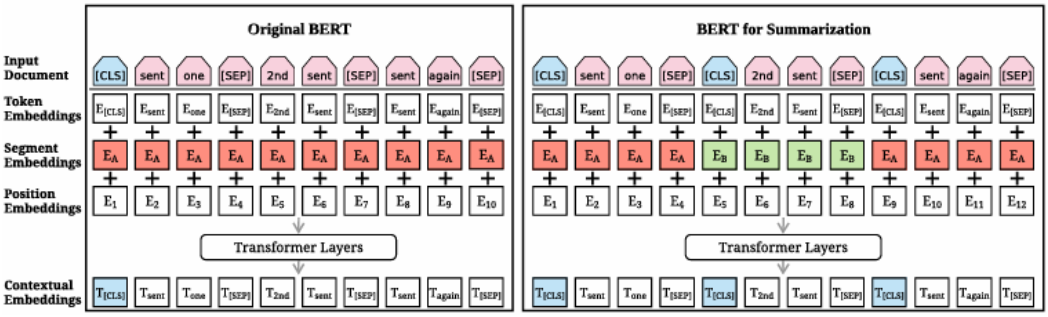

Figure 1: Architecture of the original BERT model (left) and BERTSUM (right). The sequence on top is the input document, followed by the summation of three kinds of embeddings for each token. The summed vectors are used as input embeddings to several bidirectional Transformer layers, generating contextual vectors for each token. BERTSUM extends BERT by inserting multiple [CLS] symbols to learn sentence representations and using interval segmentation embeddings (illustrated in red and green color) to distinguish multiple sentences.

内容选择器首先确定源文档中的哪些短语应该是摘要的一部分，并且在解码期间复制机制仅应用于预选短语。Narayan 等人提出了一种抽象模型，该模型特别适用于极端摘要（即单句摘要），该模型基于卷积神经网络并以主题分布（TCONVS2S）为条件。

3 本文方法及优势

3.1 BERT

BERT 是一种新的语言预训练模型，在 3300M 单词的语料库上使用掩码语言建模和“下一句预测”任务进行训练，并且可以根据不同的特定目标进行微调，在很多 NLP 任务的应用上都取得了比较好的结果。

BERT 的一般架构如图 1 的左侧部分所示。输入文本首先通过插入两个特殊标记进行预处理：$[CLS]$ 附加在正文的开头，该标记的输出表示用于聚合来自整个序列的信息；相应的，在每个句子之后插入标记 $[SEP]$ 作为句子边界的指示符。然后将修改后的文本表示为标记序列 $X = [w_1, \ldots, wn]$。每个标记 w_i 被分配了三种嵌入：标记嵌入表示每个标记的含义，分段嵌入用于区分两个句子，位置嵌入表示文本序列中的每个标记。这三个嵌入被加到一个输入向量 xi 中，并发送到具有多层的双向 Transformer：

$$\tilde{h}^l = \mathrm{LN}(h^{l-1} + \mathrm{MHAtt}(h^{l-1}))$$

$$h^l = \mathrm{LN}(\tilde{h}^l + \mathrm{FFN}(\tilde{h}^l))$$

其中 $h_0 = x$ 是输入向量，LN 是层归一化操作，$MHAtt$ 是多头注意力操作，上标 l 表示堆叠层的深度。在顶层，BERT 将为每个具有丰富上下文信息的标记生成一个输出向量 t_i。

3.2 微调 BERT 用于文本摘要

本文的工作基于 BERT 进行。如同上述，虽然 BERT 已被用于各种 NLP 任务，但它在文本摘要中的应用和其他 NLP 任务相比较为困难。现有 BERT 模型主要有以下两个问题：由于 BERT 被训练为掩码语言模型，因此输出向量基于单词而不是句子，但是在文本摘要中，需要模型在句子级向量进行操作；此外，尽管 BERT 中的分段嵌入可以代表不同的句子，但它们仅适用于一对句子的输入，但在文本摘要中，模型必须对多个句子输入进行编码和操作。

为此本文微调 BERT 后，提出一种新架构用于文本摘要。图 1 右侧展示了本文提出的用于汇总的 BERT 架构（文中称之为 BERTSUM）。为了进行句子级表示，BERTSUM 在每个句子的开头插入外部 $[CLS]$ 标记，每个 $[CLS]$ 符号收集前面句子的特征。为了对多个句子进行操作，BERTSUM 使用间隔片段嵌入来区分文档中的多个句子。对于发送的第 i 个句子 $sent_i$，BERTSUM 根据 i 是奇数或者偶数分配段嵌入 E_A 或 E_B 。例如，对于文档 $[sent_1, sent_2, sent_3, sent_4, sent_5]$ ，BERTSUM 将分配嵌入 $[E_A, E_B, E_A, E_B, E_A]$。在这种表示方式下，文档表示是分层学习的，其中较低的 Transformer 层表示相邻的句子，而较高的层与自注意力机制相结合，表示多句话语。这样就解决了 BERT 模型在文本摘要上的两个问题。

除此之外，BERTSUM 还对 BERT 进行了扩展。原始 BERT 模型中的位置嵌入最大长度为 512，BERTSUM 通过添加更多位置嵌入来克服这个限制，这些位置嵌入是随机初始化的，可以根据编码器中的其他参数进行微调。

通过 BERTSUM 模型的构建，本文解决了原 BERT 在文本摘要的两个问题，同时对位置嵌入长度进行了扩展。在进行抽取式文本摘要和生成式文本摘要时，只需要在 BERTSUM 基础上进行修改调整即可。

3.3 抽取式文本摘要

令 d 表示包含句子 $[sent_1, sent_2, \ldots, sent_m]$ 的文档，其中 $sent_i$ 表示文档中的第 i 个句子。提取式摘要可以定义为给每个句子 $sent_i$ 分配标签 $y_i \in \{0, 1\}$ ，表明该句子是否应包含在摘要中。

在 BERTSUM 中，向量 t_i 是来自顶层的第 i 个 $[CLS]$ 符号的向量，可以用作第 i 个句子 $sent_i$ 的表示。为了实现抽取式摘要，在此基础上再将几个句间 Transformer 层堆叠在 BERT 输出之上，以捕获文档级特征句子提取摘要：

$$\tilde{h}^l = \mathrm{LN}(h^{l-1} + \mathrm{MHAtt}(h^{l-1}))$$

$$h^l = \mathrm{LN}(\tilde{h}^l + \mathrm{FFN}(\tilde{h}^l))$$

其中 $h_0 = PosEmb(T)$，T 表示 BERTSUM 输出的句子向量，函数 $PosEmb$ 用于向 T 添加正弦曲线位置嵌入，指示每个句子的位置。

最后用一个 sigmoid 分类器作为输出层：

$$\hat{y}_i = \sigma(W_o h_i^L + b_o)$$

其中 h_i^L 是从 Transformer 第 L 层发送的第 i 个向量。模型的损失是预测 $\tilde{y_i}$ 相对于“黄金”标签 y_i 的二元分类熵。句间 Transformer 层也和 BERTSUM 联合微调。使用 $_1 = 0.9$ 和 $_2 = 0.999$ 的 Adam 优化器。学习率进度遵循 Vaswani 等人提出的带有预热 warming-up 的方法：
$lr = 2e^{-3} \cdot min(step^{-0.5}, step \cdot warmup^{-1.5}), warming\_up = 10000$。

在本文中，实验了 $L = 1, 2, 3$ 的 Transformer，结果表明 $L = 2$ 的 Transformer 表现最好，因此选择 $L = 2$ 并将这个模型命名为 BERTSUMEXT, 表明是在 BERTSUM 基础上进行微调得到的抽取式文本摘要模型。

3.4 生成式文本摘要

对于生成式文本摘要，本文结合了 BERTSUM 和 Abigail See 等人提出的编码器-解码器架构。编码器是预训练的 BERTSUM，解码器是随机初始化的 6 层 Transformer。显然，编码器和解码器之间是不匹配的，因为编码器是预训练的解码器是未训练的。

为了解决这个问题，本文又提出了一个新的微调方法，将编码器和解码器的优化器分开。本文使用两个 Adam 优化器，$_1 = 0.9$ 和 $_2 = 0.999$ 分别用于编码器和解码器，两个优化器分别有不同的预热步骤和学习率：

$$lr_{\mathcal{E}} = \tilde{lr}_{\mathcal{E}} \cdot \min(step^{-0.5}, \text{step} \cdot \text{warmup}_{\mathcal{E}}^{-1.5})$$

$$lr_{\mathcal{D}} = \tilde{lr}_{\mathcal{D}} \cdot \min(step^{-0.5}, \text{step} \cdot \text{warmup}_{\mathcal{D}}^{-1.5})$$

其中 $\tilde{lr_\epsilon} = 2e^{-3}$，$warmup_\epsilon = 20000$；$lr_D = 0.1$，$warmup_D = 10000$。分别对应编码器和解码器的预热步骤和学习率。这样取值的原因是编码器应该以更小的学习率和和更平滑的衰减进行微调，这样当解码器稳定时，编码器可以用更准确的梯度进行训练。

根据上述步骤得到的模型本文将其命名为 BERTSUMABS，表明这是在 BERTSUM 基础上得到的生成式文本摘要模型。

3.5 两阶段微调

在得到 BERTSUMEXT 和 BERTSUMABS 两个适用于不同文本摘要方式的模型后，本文还进行了进一步优化。Gehrmann 等人的工作表明，使用提取式模型处理后会提高生成式模型的性能。同时注意到，两个模型可以共享信息，也无需对两个模型架构进行改变，只需要进行简单的结合即可。受此启发，本文先使用 BERTSUMEXT 微调编码器，然后使用 BERTSUMABS 微调编码器，得到一种两阶段微调模型 BERTSUMEXTABS，表明这是先用 BERTSUMEXT 处理然后用 BERTSUMABS 处理得到的两阶段模型。

3.6 优势和创新

本文提出的基于 BERT 的新通用文本摘要框架主要有以下三点优势和创新：

- 最近提出的各种应用于文本摘要的技术主要通过赋值机制、强化学习来提供性能，而本文没有利用这些机制，反而着眼于文档编码，基于 BERT 编码模型微调进行文本摘要，并体现出了优于其他技术的文本摘要性能，为文本摘要提供了一条新的道路。
- 展示了如何使用预训练模型进行文本摘要，为其他预训练模型应用于文本摘要提供了可能性。
- 本文展示的基于 BERT 的文本摘要模型可以作为进一步将预训练模型应用于文本摘要的基石，所体现的性能也可以作为进一步评判新的文本摘要预训练模型性能的标准。

4 实验设置

4.1 数据集

4.1.1 来源

本文使用了三个数据集：CNN/DailyMail 亮点新闻、NYT 纽约时报注释语料库、XSum 语库。这三个数据集体现了不同的摘要风格：从内容来说，有些是新闻亮点，有些是简短的概括；从组织来说，有些是不同内容的拼凑，更适用于抽取式文本摘要，而有些则是抽象的凝练，更适用于生成式文本摘要。

4.1.2 处理方式

本文在原始数据集上进行了预处理：

- 对于 CNN/DailyMail，本文首先使用了 Hermann 等人的标准拆分。将原始数据集划分为三份：90,266/1,220/1,093 CNN 文档和 196,961/12,148/10,397DailyMail 文档，分别用于训练、验证和测试。然后用 Stanford CoreNLP 工具包拆分句子，并按照 Abigail See 等人的方法对数据集进行预处理，将输入文档截断为 512 个标记。
- 对于 NYT，本文采用 Durrett 等人的处理方式，根据发布日期将全部 110,540 篇带有抽象摘要的文章分成 100,834/9,706 个训练/测试示例，并且使用训练中的 4,000 个示例作为验证集。并且沿用了 Durrett 等人的过滤程序，从数据集中删除了摘要少于 50 个单词的文档，过滤后的测试集 (NYT50) 包括 3,452 个示例。然后使用 Stanford CoreNLP 工具包拆分句子，并按照 Durrett 等人的方法进行预处理，将输入文档截断为 800 个标记。
- XSum 包含 226,711 篇新闻文章，并附有一句话摘要，回答了“这篇文章是关于什么的?”这个问题。本文使用了 Narayan 等人的划分方法将原始数据集划分为 204,045/11,332/11,334 三份用于训练、验证和测试，并遵循他们工作中引入的预处理，将输入文档截断为 512 个标记。

处理后三个数据集的统计数据在表 1 中展示，除此之外，表 1 还报告了标准摘要中新词汇的比例，以衡量其抽象性。其中，CNN/DailyMail 和 NYT 是抽取性的，而 XSum 是高度抽象的。

Datasets	# docs (train/val/test)	avg. doc length		avg. summary length		% novel bi-grams
		words	sentences	words	sentences	in gold summary
CNN	90,266/1,220/1,093	760.50	33.98	45.70	3.59	52.90
DailyMail	196,961/12,148/10,397	653.33	29.33	54.65	3.86	52.16
NYT	96,834/4,000/3,452	800.04	35.55	45.54	2.44	54.70
XSum	204,045/11,332/11,334	431.07	19.77	23.26	1.00	83.31

Table 1: Comparison of summarization datasets: size of training, validation, and test sets and average document and summary length (in terms of words and sentences). The proportion of novel bi-grams that do not appear in source documents but do appear in the gold summaries quantifies corpus bias towards extractive methods.

4.2 实验细节

BERTSUM 是实验进行的基础。本文采用 PyTorch、OpenNMT 和“bert-base-uncased”版本的 BERT 来实现 BERTSUM。对于源文本和目标文本，都采用 BERT 的子词标记器进行标记。

4.2.1 抽取式摘要实现

抽取摘要模型在 3 个 GPU（GTX1080 Ti）上训练了 50000 步，每两步进行一次梯度累积，每 1000 步保存一次模型检查点并在验证集上对其进行评估。根据验证集上的评估损失选择了前 3 个检查点，并报告了在测试集上的平均结果。

为了训练模型，本文使用类似于 Nallapati 等人的贪心算法，获取每个文档的 oracle 摘要。该算法生成一个由多个句子组成的 oracle，该 oracle 使得相对于标准“黄金”摘要的 ROUGE-2 标准下的分数最大化。在为新文档预测摘要时，首先使用该模型获得每个句子的分数，然后按照分数从高到低对这些句子进行排序，并选择前 3 个句子作为摘要。

此外，本文使用 *TrigramBlocking* 来减少摘要冗余。给定摘要 S 和候选句子 c，如果 c 和 S 之间存在三角形重叠，我们将跳过 c。进行这种处理的灵感来源于 Carbonell 和 Goldstein 提出的最大边际相关性（MMR）。通过这种处理，可以减少当前句子和已经被选为摘要的句子之间的相似性，从而减少摘要的冗余度。

4.2.2 生成式摘要实现

在抽象摘要模型中进行了以下处理：在所有线性层之前应用了概率为 0.1 的 dropout，还使用了 Szegedy 等人提出的标签平滑方法，其中平滑因子设为 0.1。Transformer 解码器有 768 个隐藏单元，所有前馈层的隐藏大小为 2048。

模型都在 4 个 GPU（GTX1080 Ti）上训练了 200,000 步，每五步梯度累积一次。每 2,500 步保存一次模型检查点并在验证集上对其进行评估。根据它们在验证集上的评估损失选择了前 3 个检查点，并报告了测试集上的平均结果。

在解码期间，本文使用大小为 5 的波束搜索，并在验证集上将长度惩罚的 调整为 0.6 和 1 之间，解码持续到发出序列的标记结束以及重复的三角形停止出现。

值得注意的是，本文所用的解码器既不使用复制机制也不使用覆盖机制，尽管这些机制都是文本摘要中所常用的。这是因为本文试图构建一个最小要求的模型，而这些机制都需要引入额外的超参数，会使得模型更复杂，这不是本文想看到的。作为这些机制的替代，子词标记器和 $TrigramBlocking$ 起到了很好的作用，前者让词汇表之外的非合法单词的出现几近于无，后者则让摘要的冗余度大大降低。

5 实验结果

5.1 自动评估

5.1.1 评估标准

本文使用 ROUGE 标准评估摘要质量。用 ROUGE-1 和 ROUGE-2 作为评估摘要所蕴含信息质量的标准，用 ROUGE-L 作为评估摘要流畅性的标准。

5.1.2 对比模型

在三个数据集上的对比模型主要是前文所提到的现有文本摘要模型加上 ORACLE 提取系统和 LEAD-3 提取模型（简单提取每个文档的前三句）。除此在外，本文还实现了两个非预训练的 Transformer 模型分别进行抽取式文本摘要和生成式文本摘要对比:

- 一个是 TransformerEXT，它使用与 BERTSUMEXT 相同的架构，但参数更少。它是随机初始化的，只接受文本摘要的训练。TransformerEXT 有 6 层，隐藏大小为 512，前馈滤波器大小为 2,048。该模型使用与 Vaswani 等人相同的设置进行训练。
- 一个是 TransformerABS，它与 BERTSUMABS 模型具有相同的解码器，编码器是一个 6 层的 Transformer，隐藏大小为 768，前馈滤波器大小为 2048。

5.1.3 结果展示与分析

本文将 BERTSUMEXT，BERTSUMABS 以及 BERTEXTABS 三个模型和上述对比模型分别在三个数据集上测试评估，得到以下结果：

- CNN/DailyMail。表 2 展示了在 CNN/DailyMail 上各模型的得分情况。可以看到，本文提出的基于 BERT 的模型得分超过了除 ORACLE 系统外的所有现有文本摘要模型。在所有 BERT 模型中，BERTSUMEXT 得分最高，这并不难预料，因为 CNN/DailyMail 具有抽取性，更适

用于 BERTSUMEXT。此外，本文还在当前数据集增加了更大版本 BERTSUMEXT 和带有区间嵌入的 BERTSUMEXT 进行测试，结果表明增大模型会带来性能提升，而区间嵌入只会增加很小的收益。

- NYT。表 3 展示了在 NYT 上各模型的得分情况。在这个数据集上，使有限长度的 ROUGE Recall，预测摘要被截断为标准“黄金”摘要的长度。在对比模型上增加了模型 COMPRESS，这是一种是一种基于 ILP 的模型，结合了压缩和照应性约束。可以看到，本文提出的基于 BERT 的模型得分超过了除 ORACLE 系统外的所有现有文本摘要模型。在所有 BERT 模型中，BERTABS 比 BERTEXT 展现了更好的性能。对比上一个数据集，这也说明 NYT 的抽象性要比 CNN/-DayilMail 更高。而 BERTEXTABS 的性能比 BERTEXTABS 还要好，接近 ORACLE 系统的性能，这也验证了本文之前提到的两阶段微调会带来更好的表现。

- XSum。表 4 展示了在 XSum 上各模型的得分情况。由于这个数据集高度抽象，而 LEAD-3 只是简单提取文档前三句，因此在这个数据集下性能很低，不计入对比。相应的，BERTSUMEXT 也不计入对比。可以看到，在这个高度抽象的数据集上，BERTABS 和 BERTEXTABS 的性能十分接近，超过了所有现有模型。

Model	R1	R2	RL
ORACLE	52.59	31.24	48.87
LEAD-3	40.42	17.62	36.67
Extractive			
SUMMARUNNER (Nallapati et al., 2017)	39.60	16.20	35.30
REFRESH (Narayan et al., 2018b)	40.00	18.20	36.60
LATENT (Zhang et al., 2018)	41.05	18.77	37.54
NEUSUM (Zhou et al., 2018)	41.59	19.01	37.98
SUMO (Liu et al., 2019)	41.00	18.40	37.20
TransformerEXT	40.90	18.02	37.17
Abstractive			
PTGEN (See et al., 2017)	36.44	15.66	33.42
PTGEN+COV (See et al., 2017)	39.53	17.28	36.38
DRM (Paulus et al., 2018)	39.87	15.82	36.90
BOTTOMUP (Gehrmann et al., 2018)	41.22	18.68	38.34
DCA (Celikyilmaz et al., 2018)	41.69	19.47	37.92
TransformerABS	40.21	17.76	37.09
BERT-based			
BERTSUMEXT	43.25	20.24	39.63
BERTSUMEXT w/o interval embeddings	43.20	20.22	39.59
BERTSUMEXT (large)	43.85	20.34	39.90
BERTSUMABS	41.72	19.39	38.76
BERTSUMEXTABS	42.13	19.60	39.18

Table 2: ROUGE F1 results on **CNN/DailyMail** test set (R1 and R2 are shorthands for unigram and bigram overlap; RL is the longest common subsequence). Results for comparison systems are taken from the authors' respective papers or obtained on our data by running publicly released software.

Model	R1	R2	RL
ORACLE	49.18	33.24	46.02
LEAD-3	39.58	20.11	35.78
Extractive			
COMPRESS (Durrett et al., 2016)	42.20	24.90	—
SUMO (Liu et al., 2019)	42.30	22.70	38.60
TransformerEXT	41.95	22.68	38.51
Abstractive			
PTGEN (See et al., 2017)	42.47	25.61	—
PTGEN + COV (See et al., 2017)	43.71	26.40	—
DRM (Paulus et al., 2018)	42.94	26.02	—
TransformerABS	35.75	17.23	31.41
BERT-based			
BERTSUMEXT	46.66	26.35	42.62
BERTSUMABS	48.92	30.84	45.41
BERTSUMEXTABS	49.02	31.02	45.55

Table 3: ROUGE Recall results on **NYT** test set. Results for comparison systems are taken from the authors' respective papers or obtained on our data by running publicly released software. Table cells are filled with — whenever results are not available.

Model	R1	R2	RL
ORACLE	29.79	8.81	22.66
LEAD	16.30	1.60	11.95
Abstractive			
PTGEN (See et al., 2017)	29.70	9.21	23.24
PTGEN+COV (See et al., 2017)	28.10	8.02	21.72
TCONVS2S (Narayan et al., 2018a)	31.89	11.54	25.75
TransformerABS	29.41	9.77	23.01
BERT-based			
BERTSUMABS	38.76	16.33	31.15
BERTSUMEXTABS	38.81	16.50	31.27

Table 4: ROUGE F1 results on the **XSum** test set. Results for comparison systems are taken from the authors' respective papers or obtained on our data by running publicly released software.

5.2 人工评估

5.2.1 QA 评价

QA 范式是一种评估摘要质量的人工评估方式，量化了摘要模型从文档中保留关键信息的程度。首先基于标准摘要创建一组问题，尽量保证这组问题可以展现原文最关键的信息。然后要求参与者只阅读模型产生的摘要来回答这些问题。显然，参与者可以回答的问题越多，摘要对整个文档的总结就越好，摘要的质量就越高。

对于 QA 评价，本文采用采用了 Clarke and Lapata 的评分机制：正确答案标记为 1 分，部分正确答案标记为 0.5 分，否则为零分。

5.2.2 质量保证评估

模型生成的摘要可能有不流畅或者不合语法的低质量输出。为了对摘要的质量进行人工评估，可以向参与者展示源文档和模型输出的摘要，要求参与者根据信息量、流畅性和简洁性选择哪个更好。

对于质量保证评估，本文采用最佳-最差缩放比的评分机制：每个模型的评分计算为它被选为更好的次数减去它被选为更差的次数的百分比。因此，评分范围从 -1（最差）到 1（最好）。

5.2.3 结果展示

文中两个方面的人工评估都是在 Amazon Mechanical Turk 平台上进行的。对于质量保证评估，使用 CNN/DailyMail 和 NYT 两个数据集；而对于 QA 评价则使用全部三个数据集。对比模型是当前最先进的系统、LEAD 基本标准以及作为评估上限的“黄金”标准。结果分别显示在表 6 和表 7 中，其中 BERTSUM 表示本文中所有 BERT 模型中表现最好的一种。可以看到，参与者绝大多数更喜欢本文提出模型的输出。

Extractive	CNN/DM	NYT
LEAD	42.5†	36.2†
NEUSUM	42.2†	—
SUMO	41.7†	38.1†
Transformer	37.8†	32.5†
BERTSUM	58.9	41.9

Table 6: QA-based evaluation. Models with † are significantly different from BERTSUM (using a paired student t-test; $p < 0.05$). Table cells are filled with — whenever system output is not available.

	CNN/DM		NYT		XSum	
Abstractive	QA	Rank	QA	Rank	QA	Rank
LEAD	42.5†	—	36.2†	—	9.20†	—
PTGEN	33.3†	-0.24†	30.5†	-0.27†	23.7†	-0.36†
BOTTOMUP	40.6†	-0.16†	—	—	—	—
TCONVS2S	—	—	—	—	52.1	-0.20†
GOLD	—	0.22†	—	0.33†	—	0.38†
BERTSUM	56.1	0.17	41.8	-0.07	57.5	0.19

Table 7: QA-based and ranking-based evaluation. Models with † are significantly different from BERTSUM (using a paired student t-test; $p < 0.05$). Table cells are filled with — whenever system output is not available. GOLD is not used in QA setting, and LEAD is not used in Rank evaluation.

5.3 其他

5.3.1 抽象模型学习率选择

BERTSUMABS 通过对编码器和解码器应用不同的优化器实现，而两个编码器的学习率 $\tilde{l_\epsilon}$ 和 $\tilde{l_D}$ 组合的不同选择会对结果产生什么影响？什么样的组合是最佳的呢？文中通过不同组合下，BERTSUMABS 在 CNN/DailyMail 验证集上的困惑度（表 5）得出结果：$\tilde{l_\epsilon} = 2e^{-3}$，$\tilde{l_D} = 0.1$ 时，困惑度最低。

5.3.2 提取句子位置评估

对于抽取式文本摘要，摘要中每个句子在源文档中的位置也是评价摘要质量的重要指标。在一定程度上，摘要中的句子在源文档中分布越均衡，说明模型选择摘要时，对源文档的理解更深，而不是简单选择头部或者尾部句子作为摘要，摘要的质量自然也更好。

因此，本文还对 BERTSUMEXT，TransformerEXT，以及在其他对比模型中表现最好的 ORACLE 三个模型在 CNN/DailyMail 数据集上摘要结果在源文档每个位置的分布进行了统计，得到图 2 中的结果。可以看到，BERTSUMEXT 和 ORACLE 的分布接近，都很均衡，而 TransformerEXT 的分布集中于第一句和第二句。可见 BERTSUMEXT 和 ORACLE 都对文档进行了较深层的处理，而 TransformerEXT 更倾向于简单选择第一二句作为摘要。

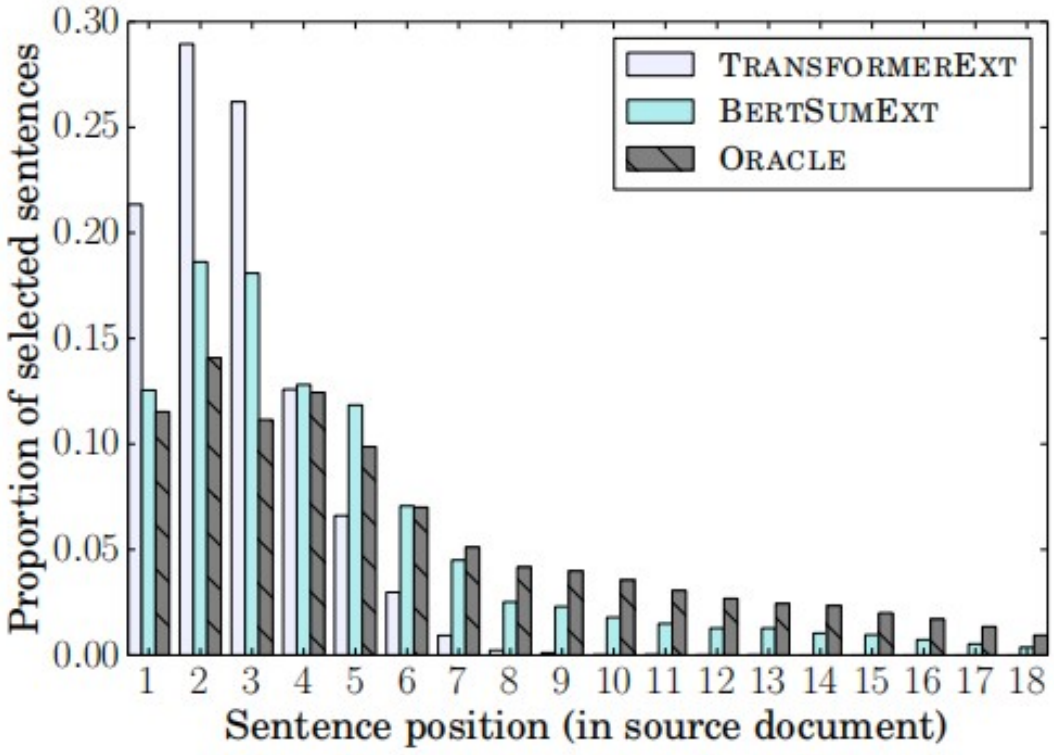

Figure 2: Proportion of extracted sentences according to their position in the original document.

5.3.3 新词汇占比评估

对于生成式文本摘要，摘要中不同于源文档的新词汇的占比也是评价摘要质量的重要指标。生成式文本摘要处理的文档一般比较抽象，没有明确的总结句，直接从文中摘取句子的效果可能不太好，需要模型对文档有深层次理解生成新的摘要。一般来说，新词汇占比越高，摘要的质量越好。

因此，本文统计了 BERTSUMABS 和 BERTSUMEXTABS 在 CNN/DailyMail 和 XSum 两个数据集上摘要结果中新词汇的占比，并且和标准摘要的占比做了对比，结果见图 3。可以看到，在 CNN/DailyMail 上，两个模型的新词汇占比相对于参考都比较低，这可能是因为 CNN/DailyMail 的文档抽取性过高，抽象性过低，因此两个模型都容易进行原文摘取；而在 XSum 上，两个模型的新词汇非常接近参考，取得了较好的效果，这是因为 XSum 的文档是高度抽象的，适用于两个抽象模型。同时可以看到，BERTSUMEXTABS 在两个数据集上的效果都比 BERTSUMABS 要略好，同样验证了文中提到的两阶段微调带来的收益。

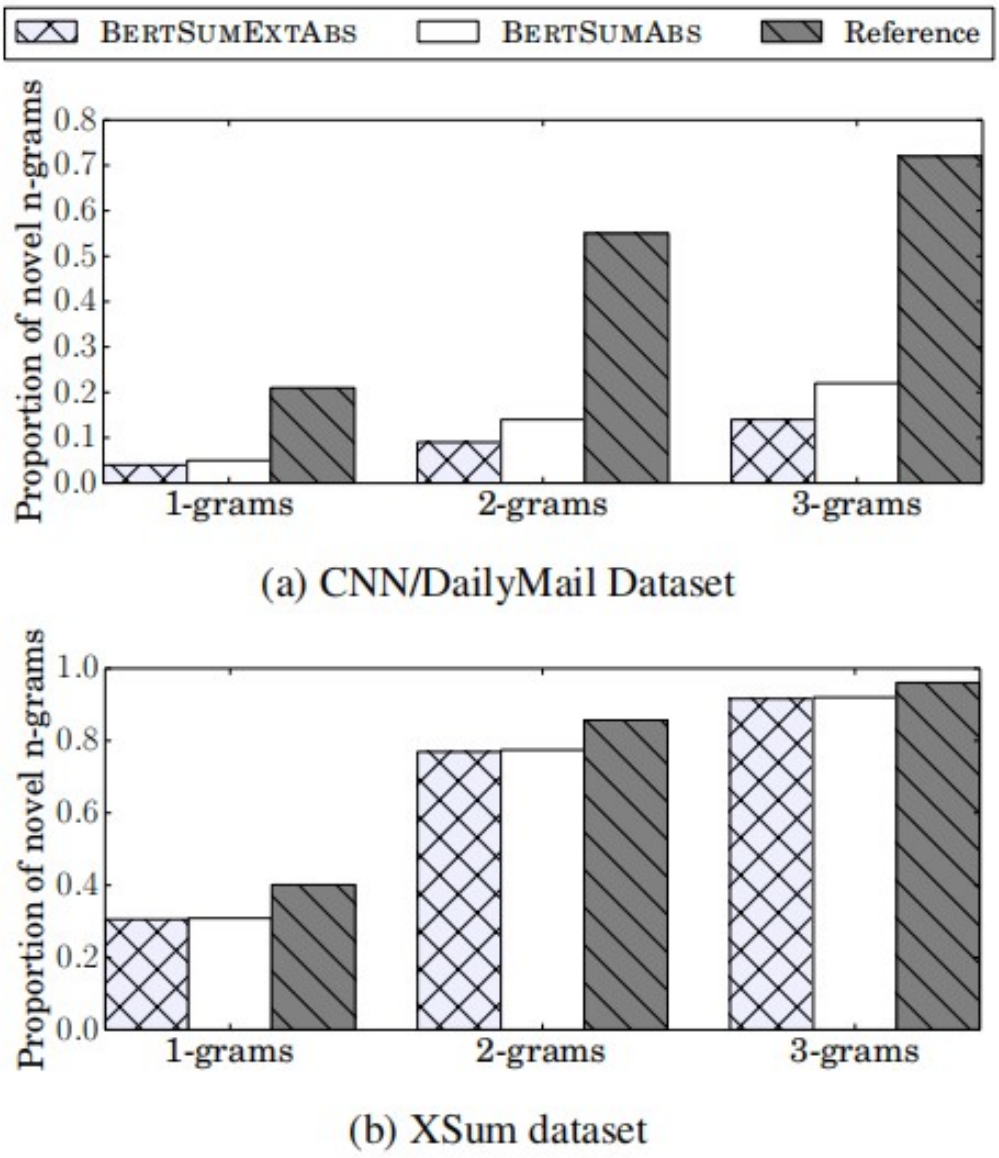

Figure 3: Proportion of novel n-grams in model generated summaries.

6 局限和扩展

本文主要展示了预训练的 BERT 模型在文本摘要中的应用，提出了一个抽取式和生成式的同用框架，在自动评估，人工评估以及其他评估标准下都取得了领先于其他模型的效果。但是该模型不具备语言生成的能力，如何将 BERT 应用于语言生成是一个可以考虑的扩展方向。

7 代码复现

7.1 实验概述

在自然语言处理（Natural Language Processing, NLP）的研究和应用中，HuggingFace 的 Transformers 库已经成为了一个不可或缺的工具。它提供了大量的预训练模型（pre-trained models）和强大的接口，极大地简化了开发者在复杂任务上的工作。Transformers 库是由 HuggingFace 团队基于 pytorchk 框架开发的一个开源库，它提供了一系列接口和预训练模型，用于处理自然语言处理任务，包括但不限于文本分类、信息抽取、问答系统等。Transformers 支持包括 Bert 等众多主流预训练模型并提供简单的接口用于加载模型、进行预测和微调且支持 PyTorch 和 TensorFlow 两大深度学习框架。因此，在本次实验中，我们将在 Pytorch 框架下借助 Transformers 库对 Bert 模型进行微调，使之完成信息抽取，qa 问答和文本摘要任务。

7.2 文本信息提取

未经过任何训练之前的原始 BERT 只能完成两种自然语言处理任务，一种是 MLM(预测被掩盖文字)，另一种是 NSP(预测语句或片段是否连续)，它并没有经过问答任务的训练。因此如果要令 BERT 完成问答任务，要使用 Transformer 对 BERT 进行微调，然后再用微调后的 BERT 模型去完成问答任务。

7.2.1 数据集选择

数据集选用 SQuAD2.0，该数据集是斯坦福大学发布的一个常用于问答任务的标准数据集，它是从维基百科里面抽取出来很多的问题和很多的答案，它每一个问题之后都接一个文本段。经过处理后的数据集结构如下所示

```
{
    "question":[" 问题"]
    "text":[" 文本"]
    "answer":[" 答案"]
}
```

对于所有的问题，其答案均在文本中，所以此问题其实是一个抽取类型的任务，即从文本中抽取与问题语义相近的词或很多词的组合。该任务与对文本直接摘要相似，区别在于对模型而言注意力集中于与问题语义相似的文本上，不需要解析全文结构。

7.2.2 实验过程

1. 数据集特征提取：将 SQuAD 2.0 数据集分割成训练集，验证集，测试集转换为 BERT 模型的输入特征，并将这些特征以 plk 格式保存到磁盘上，减少重复计算，后续直接加载数据集即可。

2. 加载 SQuAD 特征数据集：使用了 pickle 库打开保存在磁盘上的特征文件，并加载特征数据到变量 train_features 中。使用 torch.tensor() 函数将特征中的各个字段转换为 PyTorch 张量。代码使用 RandomSampler 对训练数据集进行随机采样，并使用 DataLoader 将训练数据

集转换为可迭代的数据加载器。加载器会按照指定的批次大小将数据划分为小批次进行训练。batch_size=8

3. 微调 BERT：加载原始 BERT 模型 (bert-base-uncased) 和优化器，并对 BERT 模型进行微调, 采用 AdamW 优化器对 BERT 模型的参数进行优化, 初始学习率 lr=3e-4, 训练周期 num_epochs=3。训练完成后将微调后模型存储到指定路径便于再次加载进行下一步的推理测试。
4. 加载分词器，对输入的问题和文本进行分词。从前一步中保存的模型路径中加载模型，输入问题和文本，观察给出的答案。

7.2.3 实验结果

测试示例：

> question, text = "What is the main idea? ", " Change is an immutable facet of life, an unwavering force guiding our journey through existence. While often viewed with trepidation, it serves as the catalyst for growth and renewal. Much like a river forging its course through rugged landscapes, change molds our lives, sculpting us into resilient beings capable of facing adversity. It beckons us to venture beyond our comfort zones, to explore uncharted territories, and unearth hidden potentials. Embracing change signifies our adaptability, a testament to our willingness to evolve and thrive in an ever-evolving world. In the symphony of life, let us welcome change with open arms, for within its currents lie boundless opportunities and the promise of profound transformation."

给出的回答：

> comfort zones, to explore uncharted territories, and unearth hidden potentials. embracing change

收集类似的 $nmb = 213$ 组问答数据，通过使用词袋模型结合余弦相似度 (similarity)，统计给出答案与标准答案相似度求和，计算 $acc = \sum_1^{nmb}(similarity)/nmb$ 最终在测试集上 $acc = 0.304139$

同时经测试，模型具有一定排除无意义文本和分辨相似文本能力。

示例：

> question, text = "What is the population of Shenzhen?", " The population of Shenzhen is approximately 13 million,the population of USA is approximately 200 million."

回答 13 million，做出正确回答。

从运行结果可以看到 BERT 训练完后，可以正确的理解 QA 问答并找出正确的答案。

7.3 文本摘要任务 (NLC)

Transfomer 模型在自然语言处理领域中的大多数任务中表现出了惊人的成果。迁移学习和大规模 Transformer 语言模型的结合已经成为了先进 NLP 的标准。下面我将使用 Hugging Face Transformers 库对 Bert 进行微调实现文本摘要任务。实现文本摘要有两种方式，抽取式与生成式。其中

生成式所需训练样本量极大，如 xsum 完整约 2.5G，且对算力要求较高，训练难度过大，因此选择抽取式摘要。

数据集选择：数据集存放在 nplcc.json 中，该数据集包括 5000 条中文新闻的文本和对应的摘要。结构如下所示：

```
[
{
    "title": " 演员克里斯托弗·李去世，享年 93 岁，代表角色为《指环王》《霍比特人》
              的白袍巫师萨茹曼。",
    "content": "2015-06-1120:03 新浪娱乐显示图片 ChristopherLee 新浪娱乐讯据 BBC
                报道，好莱坞演员克里斯托弗去世，享年 93 岁。他曾出演《指环王》
                《霍比特人》等影片。"
}
]
```

该数据集中，content 中存放新闻原文，title 中存放的是摘要。

7.3.1 实验过程

1. 导入和加载数据集：导入必要的库，并使用 datasets 库的 load_dataset 函数从 JSON 文件中加载数据集。然后，将数据集扁平化，创建包含”document” 和”summary” 字段的新数据集。

2. 初始化 Bert 和分词器，由于数据是中文 utf-8 格式，选择初始模型为”bert-base-chinese”。指定 BART 模型的检查点 facebook/bart-large-cnn,设定最大输入输出文本长度 max_input_length = 1024 ，max_target_length = 256。将数据集拆分为训练集、验证集和测试集并使用预处理函数对这些拆分后的数据集进行分词。

3. 复制原模型到一个新建的模型中，对新建模型进行处理避免破坏原模型，每次经过一定 step 的训练，若没发生明显梯度爆炸等问题再用新建模型替换原模型。

4. 加载用于序列到序列任务的预训练模型。设置训练参数，包括批量大小 batch_size=2、训练轮数 num_train_epochs=1、初始学习率 lr=3e-5, 两次评估间隔 logging_steps=50。定义计算评价指标的函数，使用 ROUGE 指标评估模型生成的摘要质量。初始化训练器对象，采用 Adamw 优化器并开始模型训练。

5. 总共 item 约 20500 项，训练完成后保存表现最好的模型。

7.3.2 实验结果

在测试集上采用 lawrouge 库计算文本摘要的 ROUGE-L 分数，并计算平均得分。lawrouge 是一个用于计算 ROUGE 分数的库，ROUGE 分数是用于评估自动摘要和机器翻译质量的标准指标。它主要用于比较生成的文本（预测值）与参考文本（真实值）之间的相似度。将测试集真实值和预测值放在同一列表下得到平均得分，约为 0.4412

同时，我们也采用不同类型文章进行测试，如中文科学文献 CSL 摘要数据集

测试示例：

原文：提出了 H.264 码率控制改进算法。首先, 基于帧间运动估计残差的空间和时间相关性提出了十字 MAD 预测模型; 为了平衡缓冲的占有率, 在分配帧比特时将前一帧的比特使用情况作为当前帧比特分配时的惩罚因子; 最后, 结合视频文字对象提取技术实现了基于文字对象的宏块比特分配以及量化步长的计算。实验结果表明, 改进算法能在准确控制码率的情况下, 提高输出序列的峰值信噪比。。

模型摘要：快讯: 基于帧间运动估计残差的空间和时间相关性提出了 H. 264 码率控制改进算法。

标准摘要：提出了 H.264 码率控制改进算法

通过计算 ROUGE 分数，在 100 组平均得分约为 0.4237

数据集	平均文本长度	平均摘要长度
nlucc	532.13	24.37
CSL 摘要	197.32	23.71

7.4 总结

本实验利用 Hugging Face Transformers 库对 Bert 进行了微调并式 BERT 在 qa 问答，信息摘取，摘要抽取任务中，除存在少数情况下对文本逻辑进行错误地重组等问题，对自然语言的处理理解能力有较大幅度提升。

8 工作分工

- 孙璐阳负责源代码复现和复现结果报告的撰写。
- 赵卓负责论文阅读报告的撰写。
- 何昕负责两个部分的修改，润色，整合，排版与优化。