机器学习研究》杂志 21 (2020) 1-67 1/20 提交；6/20 修订；6/20 出版

用统一的文本到文本转换器探索迁移学习的极限

科林-拉斐尔* craffel@gmail.com
Noam Shazeer * NOAM@GOOGLE.COM
亚当-罗伯茨 * ADAROB@GOOGLE.COM
凯瑟琳-李 * KATHERINELEE@GOOGLE.COM
沙兰-纳兰 SHARANNARANG@GOOGLE.COM
迈克尔-马泰纳 mmatena@google.com
周彦琦 YANQIZ@GOOGLE.COM
Wei Li MWEILI@GOOGLE.CO
Peter J. Liu M
PETERJLIU@GOOGLE.COM
美国加利福尼亚州山景城 94043 谷歌公司

编辑：伊万-季托夫

摘要

迁移学习是指先在数据丰富的任务中对模型进行预训练，然后再在下游任务中对模型进行微调，它已成为自然语言处理（NLP）领域的一项强大技术。迁移学习的有效性催生了各种方法、方法论和实践。在本文中，我们引入了一个统一的框架，将所有基于文本的语言问题转换为文本到文本的格式，从而探索了 NLP 的迁移学习技术。我们的系统研究比较了数十种语言理解任务的预训练目标、架构、无标记数据集、迁移方法和其他因素。通过将我们的探索见解与规模和新的 "巨型清洁抓取语料库 "相结合，我们在摘要、问题解答、文本分类等多项基准测试中取得了最先进的结果。为了促进未来在 NLP 迁移学习方面的工作，我们发布了我们的数据集、预训练模型和代码。[1]

关键词：迁移学习、自然语言处理、多任务学习、基于注意力的模型、深度学习

1. 导言

要训练机器学习模型来执行自然语言处理（NLP）任务，通常要求模型能够以适合下游学习的方式处理文本。这可以宽泛地看作是开发通用知识，使模型能够 "理解 "文本。这种知识的范围可以很广，从低层次的（如拼写、语法等）到高层次的（如语法等）。

*.平等贡献。每位作者的贡献见附录 A。通讯 至 craffel@gmail.com。

1. https://github.com/google-research/text-to-text-transfer-transformer

或词义）到高级知识（例如，大号乐器太大，大多数背包都装不下）。在现代机器学习实践中，很少明确地提供这些知识；相反，这些知识通常是作为辅助任务的一部分来学习的。例如，历史上常见的方法是使用单词向量（Mikolov 等人，2013b,a；Pennington 等人，2014 年）将单词特性映射到连续表示中，理想情况下，相似的单词映射到相似的向量。这些向量通常是通过一个目标来学习的，例如，鼓励在连续空间中将共现词定位在附近（Mikolov 等人，2013b）。

最近，在数据丰富的任务中对整个模型进行预训练变得越来越普遍。理想情况下，这种预训练能使模型发展出通用能力和知识，然后将其迁移到下游任务中。在计算机视觉的迁移学习应用中（Oquab 等人，2014 年；Jia 等人，2014 年；Huh 等人，2016 年；Yosinski 等人，2014 年），预训练通常是通过在大型标注数据集（如 ImageNet）上进行监督学习来完成的（Russakovsky 等人，2015 年；Deng 等人，2009 年）。与此相反，NLP 中用于迁移学习的现代技术通常在无标记数据上使用无监督学习进行预训练。这种方法最近被用于在许多最常见的 NLP 基准中获得最先进的结果（Devlin 等人，2018；Yang 等人，2019；Dong 等人，2019；Liu 等人，2019c；Lan 等人，2019）。除了经验上的优势，NLP 的无监督预训练尤其具有吸引力，因为互联网提供了大量未标注的文本数据--例如，Common Crawl 项目[2] 每月从网页中提取约 20TB 的文本数据。这与神经网络天然契合，神经网络已被证明具有显著的可扩展性，即通常只需在更大的数据集上训练更大的模型，就能获得更好的性能（Hestness 等人，2017；Shazeer 等人，2017；Jozefowicz 等人，2016；Mahajan 等人，2018；Radford 等人，2019；Shazeer 等人，2018；Huang 等人，2018b；Keskar 等人，2019a）。

这种协同作用促成了近期为 NLP 开发迁移学习方法的大量工作，产生了广泛的预训练目标（Howard 和 Ruder，2018 年；Devlin 等人，2018 年；Yang 等人，2019 年；Dong 等人、2019）、无标记数据集（Yang 等人，2019；Liu 等人，2019c；Zellers 等人，2019）、基准（Wang 等人，2019b，2018；Conneau 和 Kiela，2018）、微调方法（Howard 和 Ruder，2018；Houlsby 等人，2019；Peters 等人，2019）等等。在这一新兴领域中，技术的快速进步和多样性可能会使人们难以比较不同的算法，难以区分新贡献的效果，也难以理解现有迁移学习方法的空间。出于对更严格理解的需求，我们

利用统一的转移学习方法，系统地研究不同的方 法 ， 突 破 该领域目前的极限。

我们工作的基本思路是将每个文本处理问题都视为 "文本到文本 "问题，即以文本为输入，以新文本为输出。这种方法的灵感来源于之前的 NLP 任务统一框架，包括将所有文本问题作为问题解答（McCann 等人，2018 年）、语言建模（Radford 等人，2019 年）或跨度提取 Keskar 等人（2019b）的任务。最重要的是，文本到文本框架允许我们将相同的模型、目标、训练过程和解码过程直接应用于我们考虑的每一项任务。我们利用这种灵活性，对各种基于英语的 NLP 问题进行了性能评估，其中包括问题解答、文档提取、语法分析、语义分析和语法分析。

2. http://commoncrawl.org

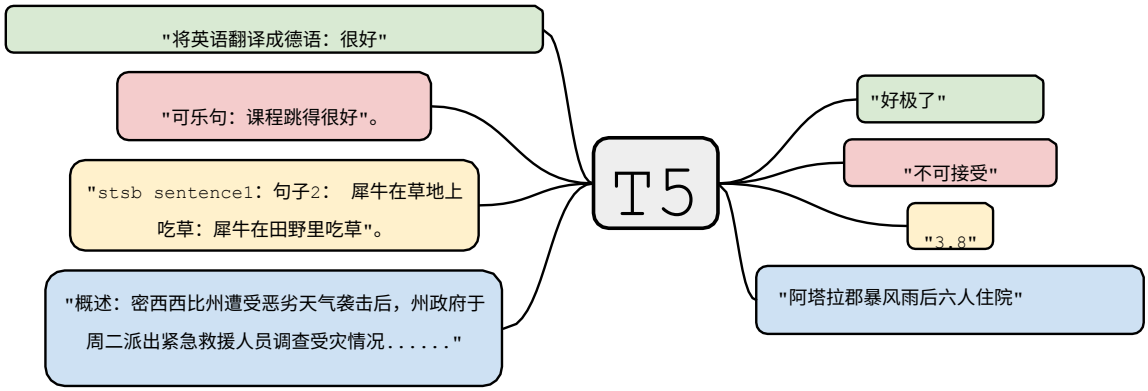

图 1：文本到文本框架图。我们考虑的每一项任务--包括翻译、问题解答和分类--都是将文本作为输入输入我们的模型，并训练它生成一些目标文本。这样，我们就能在各种任务中使用相同的模型、损失函数、超参数等。这也为我们实证调查中的方法提供了一个标准测试平台。"T5 "指的是我们的模型，我们称之为 "文本到文本转换器"。

如摘要和情感分类等。有了这种统一的方法，我们就可以比较不同迁移学习目标、无标记数据集和其他因素的有效性，同时通过扩大模型和数据集的规模来探索迁移学习在 NLP 中的局限性。

我们强调，我们的目标不是提出新的方法，而是对该领域的现状提供一个全面的视角。因此，我们的工作主要包括对现有技术的调查、探索和实证比较。我们还探索了现有方法的局限性，将系统研究（训练高达 110 亿个参数的模型）中获得的见解进行了放大，从而在我们考虑的许多任务中获得了最先进的结果。为了进行这种规模的实验，我们引入了 "巨型清洁抓取语料库"（Colossal Clean Crawled Corpus，简称 C4），这是一个由从网络上抓取的数百 GB 清洁英文文本组成的数据集。我们认识到迁移学习的主要用途是在数据稀缺的环境中利用预训练模型的可能性，因此我们发布了我们的代码、数据集和预训练模型。[1]

本文其余部分的结构如下：在下一节中，我们将讨论我们的基础模型及其实现、

我们将每个文本处理问题表述为文本到文本任务的程序，以及我们考虑的一系列任务。在第 3 节中，我们将介绍探索 NLP 迁移学习领域的大量实验。在本节末尾（第 3.7 节），我们结合系统研究的见解，在各种基准上得出了最先进的结果。最后，我们将对研究结果进行总结，并在第 4 节中展望未来。

2. 设置

在介绍我们的大规模实证研究结果之前，我们回顾了理解我们的结果所需的必要背景知识，包括 Transformer 模型架构和我们评估的下游任务。我们还介绍了我们将每个问题都视为文本到文本任务的方法，并描述了我们的 "Colossal Clean Crawled Corpus"（C4），即我们创建的基于普通抓取的数据集，作为未标记文本数据的来源。我们将我们的模型和框架称为 "文本到文本转换器"（T5）。

2.1 模型

用于 NLP 的迁移学习的早期成果利用了递归神经网络（Peters 等人，2018 年；Howard 和 Ruder，2018 年），但最近使用基于 "Transformer "架构（Vaswani 等人，2017 年）的模型变得更为常见。Transformer 最初被证明对机器翻译有效，但随后被广泛用于各种 NLP 环境（Radford 等人，2018 年；Devlin 等人，2018 年；McCann 等人，2018 年；Yu 等人，2018 年）。由于其日益普遍，我们研究的所有模型都基于 Transformer 架构。除了下文提到的细节以及我们在第 3.2 节中探讨的变体之外，我们并没有明显偏离最初提出的这一架构。我们不提供该模型的全面定义，而是请感兴趣的读者参阅原始论文（Vaswani et al., 2017）或后续教程[3,4]，以获取更详细的介绍。

转换器的主要构件是自我注意（Cheng 等人，2016 年）。自我注意是注意的一种变体（Graves，2013；Bahdanau 等人，2015），它通过用序列其余部分的加权平均值替换每个元素来处理序列。最初的 Transformer 由编码器-解码器架构组成，用于序列-序列（Sutskever 等人，2014；Kalchbrenner 等人，2014）任务。最近，使用由单个 Transformer 层栈组成的模型也变得很常见，不同形式的自我关注用于产生适合语言建模（Radford 等人，2018 年；Al-Rfou 等人，2019 年）或分类和跨度预测任务（Devlin 等人，2018 年；Yang 等人，2019 年）的架构。我们将在第 3.2 节中对这些架构变体进行实证探索。

总体而言，我们的编码器-解码器变换器实现方式与最初提出的形式（Vaswani 等人，2017 年）非常接近。首先，将输入的标记序列映射为嵌入序列，然后将其传递给编码器。编码器由一堆 "块 "组成，每个 "块 "由两个子组件组成：一个自注意层

和一个小型前馈网络。层归一化（Ba 等人，2016 年）应用于每个子组件的输入。我们使用的是简化版的层归一化，其中只对激活度进行了重新标定，而没有应用加法偏置。在层归一化之后，一个残差跳转连接（He 等人，2016 年）将每个子组件的输入添加到其输出中。在前馈网络、跳过连接、注意力权重以及整个堆栈的输入和输出中都应用了 "剔除"（Dropout，Srivastava 等人，2014 年）。解码器的结构与编码器相似，但它包括一个标准的注意权重。

3. http://nlp.seas.harvard.edu/2018/04/03/attention.html
4. http://jalammar.github.io/illustrated-transformer/

在每个自我注意层之后都有一个自我注意机制，用于注意编码器的输出。解码器中的自注意机制也使用一种自回归或因果自注意形式，它只允许模型注意过去的输出。最后解码器块的输出被送入一个具有软最大输出的密集层，其权重与输入嵌入矩阵共享。转换器中的所有注意机制都被分割成独立的 "头"，其输出在进一步处理之前会被串联起来。

由于自注意与顺序无关（即它是对集合的操作），因此通常要向变换器提供明确的位置信号。最初的 Transformer 使用正弦位置信号或学习位置嵌入，而最近使用相对位置嵌入变得越 来 越 普遍（Shaw 等人，2018；Huang 等人，2018a）。相对位置嵌入不是为每个位置使用一个固定的嵌入，而是根据自我注意机制中比较的 "关键 "和 "查询 "之间的偏移产生不同的学习嵌入。我们使用一种简化的位置嵌入形式，其中每个 "嵌入 "都是一个标量，被添加到用于计算注意力权重的相应对数中。为了提高效率，我们还在模型的所有层中共享位置嵌入参数，但在给定层中，每个注意力头使用不同的学习位置嵌入。通常情况下，我们会学习固定数量的嵌入，每个嵌入都与一系列可能的关键字查询偏移量相对应。在本研究中，我们的所有模型都使用了 32 个嵌入，嵌入范围按对数递增，直至 128 个偏移量，超过 128 个偏移量后，我们将所有相对位置分配给相同的嵌入。需要注意的是，特定层对 128 个词组以外的相对位置不敏感，但后续层可以通过结合前几层的局部信息，建立对更大偏移量的敏感度。总而言之，我们的模型与 Vaswani 等人（2017 年）提出的原始 Transformer 大致相同，只是去除了层规范偏置，将层规范化置于残差路径之外，并使用了不同的位置嵌入方案。由于这些架构变化与我们在迁移学习实证调查中考虑的实验因素是正交的，因此我们将其影响的消解留待今后的工作中进行。

作为研究的一部分，我们对这些模型的可扩展性进行了实验，即当模型的参数或层数变多时，其性能会发生怎样的变化。训练大型模型并非易事，因为这些模型可能无法在一台机器上完成，而且需要大量计算。因此，我们将模型和数据并行性结合起来，在云 TPU pod 的 "切片 "上训练模型。[5] TPU pod 是多机架 ML 超级计算机，包含 1,024 个 TPU v3 芯片，通过高速 2D 网状互连与支持 CPU 的主机相连。我们利用 Mesh TensorFlow 库（Shazeer 等人，2018 年）来轻松实现模型并行和数据并行（Krizhevsky，2014 年）。

2.2 巨大的清洁爬行语料库

以往有关 NLP 迁移学习的大部分工作都是利用大型无标记数据集进行无监督学习。在本文中，我们有兴趣衡量这种无标记数据的质量、特征和大小的影响。为了生成满足我们需要的数据集，我们利用 Common Crawl 作为从网络上抓取文本的来源。常见

5. https://cloud.google.com/tpu/

以前，Crawl 曾被用作 NLP 的文本数据源，例如用于训练 n-gram 语言模型（Buck 等人，2014 年）、常识推理的训练数据（Trinh 和 Le，2018 年）、机器翻译的平行文本挖掘（Smith 等人，2013 年）、预训练数据集（Grave 等人，2018 年；Zellers 等人，2019 年；Liu 等人，2019 年c），甚至只是作为测试优化器的巨型文本语料库（Anil 等人，2019 年）。

Common Crawl 是一个公开可用的网络档案库，它通过从抓取的 HTML 文件中去除标记和其他非文本内容来提供 "网络提取文本"。这一过程每月产生约 20TB 的抓取文本数据。遗憾的是，所产生的大部分文本都不是自然语言。取而代之的是大量的胡言乱语或模板文本，如菜单、错误信息或重复文本。此外，大量刮擦文本包含的内容对我们考虑的任何任务都不可能有帮助（攻击性语言、占位符文本、源代码等）。为了解决这些问题，我们使用了以下启发式方法来清理 Common Crawl 的网络提取文本：

- 我们只保留以结束标点符号（即句号、感叹号、问号或结束引号）结束的行。
- 我们舍弃了句子少于 3 个的页面，只保留了至少包含 5 个单词的行。
- 我们删除了任何包含 "肮脏、下流、淫秽或其他不良词语列表 "中任何词语的页面。[6]
- 许多被抓取的网页都包含警告，说明应启用 Javascript，因此我们删除了所有带有 Javascript 字样的行。
- 有些页面有 "lorem ipsum "文字占位符；我们删除了所有出现 "lorem ipsum "的页面。
- 有些页面无意中包含了代码。由于大括号"{"出现在许多编程语言（如网络上广泛使用的 Javascript）中，但不出现在自然文本中，因此我们删除了所有包含大括号的页面。
- 由于一些搜刮到的页面来自维基百科，并带有引文标记（如[1]、[需要引文]等），因此我们删除了任何此类标记。
- 许多页面都有模板式的政策声明，因此我们删除了任何包含 "使用条款"、"

隐私政策"、"cookie 政策"、"使用 cookie"、"使用 cookie "或 "使用 cookie "字符串的行。

- 为了重复数据集，我们舍弃了数据集中出现次数较多的三句话跨度，只有其中一句除外。

此外，由于我们的大部分下游任务都集中在英文文本上，因此我们使用了 `langdetect`[7] 来过滤掉任何未被归类为英文的页面，概率至少为 0.99。我们的启发式方法受到了过去使用 Common

6. https://github.com/LDNOOBW/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words
7. https://pypi.org/project/langdetect/

作为 NLP 数据源的抓取：例如，Grave 等人（2018 年）也使用自动语言检测器过滤文本并丢弃短行，Smith 等人（2013 年）和 Grave 等人（2018 年）都执行了行级重复数据删除。不过，我们选择创建一个新的数据集，因为之前的数据集使用的过滤启发式方法比较有限，没有公开提供，和/或范围不同（例如，仅限于新闻数据（Zellers 等人，2019 年；刘等人，2019c），仅包含知识共享内容（Habernal 等人，2016 年），或侧重于机器翻译的平行训练数据（Smith 等人，2013 年））。

为了建立基础数据集，我们下载了 2019 年 4 月的网络提取文本，并应用了上述过滤方法。这样生成的文本集不仅比大多数用于预训练的数据集大很多（约 750 GB），而且包含相当干净和自然的英文文本。我们将该数据集命名为 "Colossal Clean Crawled Corpus"（简称 C4），并将其作为 TensorFlow 数据集的一部分发布。[8] 我们将在第 3.4 节中考虑使用该数据集的各种替代版本的影响。

2.3 下游任务

本文的目标是测量一般语言学习能力。因此，我们研究了机器翻译、问题解答、抽象概括和文本分类等各种基准的下游性能。具体来说，我们测量了 GLUE 和 SuperGLUE 文本分类元基准、CNN/每日邮件抽象概括、SQuAD 问题解答以及 WMT 英语到德语、法语和罗马尼亚语翻译的性能。所有数据均来自 TensorFlow 数据集。[9]

GLUE（Wang等人，2018年）和SuperGLUE（Wang等人，2019年b）分别由一系列文本分类任务组成，旨在测试一般语言理解能力：

- 句子可接受性判断（CoLA（Warstadt 等人，2018 年）
- 情感分析（SST-2（Socher 等人，2013 年）
- 转述/句子相似性（MRPC（Dolan 和 Brockett，2005 年）、STS-B（Cer 等人，2017 年）、QQP（Iyer 等人，2017 年）
- 自然语言推理（MNLI（Williams 等人，2017 年）、QNLI（Rajpurkar 等人，2016 年）、RTE（Dagan 等人，2005 年）、CB（De Marneff 等人，2019 年）

- 核心参照解析（WNLI 和 WSC（Levesque 等人，2012 年）
- 句子补全（COPA（Roemmele 等人，2011 年）
- 词义消歧（WIC（Pilehvar 和 Camacho-Collados，2018 年）
- 问题解答（MultiRC（Khashabi 等人，2018 年）、ReCoRD（Zhang 等人，2018 年）、BoolQ（Clark 等人，2019 年）

8. https://www.tensorflow.org/datasets/catalog/c4
9. https://www.tensorflow.org/datasets

我们使用 GLUE 和 SuperGLUE 基准分发的数据集。为简单起见，在进行微调时，我们将 GLUE 基准中的所有任务（SuperGLUE 也是如此）视为一个任务，将所有组成数据集连接起来。根据 Kocijan 等人（2019 年）的建议，我们还将定语从句解析（DPR）数据集（Rahman 和 Ng，2012 年）纳入合并的 SuperGLUE 任务中。

CNN/Daily Mail（Hermann 等人，2015 年）数据集是作为问题解答任务引入的，但被 Nallapati 等人（2016 年）改编用于文本摘要；我们使用 See 等人（2017 年）的非匿名版本作为抽象摘要任务。SQuAD （Rajpurkar 等人，2016 年）是一个常见的问题解答基准。在我们的实验中，我们向模型输入问题及其上下文，并要求它逐个令牌生成答案。对于 WMT 英语到德语，我们使用与（Vaswani 等人，2017 年）相同的训练数据（即 News Commentary v13、Common Crawl、Europarl v7）和 `newstest2013` 作为验证集（Bojar 等人，2014 年）。对于英译法，我们使用 2015 年的标准训练数据和 `newstest2014` 作为验证集（Bojar 等人，2015 年）。对于英语到罗马尼亚语这一标准的低资源机器翻译基准，我们使用了 WMT 2016 的训练集和验证集（Bojar 等人，2016 年）。请注意，我们只在英语数据上进行预训练，因此为了学习翻译，给定的模型需要学习用一种新语言生成文本。

2.4 输入和输出格式

为了在上述各种任务中训练一个模型，我们将所有的任务都转换成了 "文本到文本 "的形式，也就是说，在一个任务中，先向模型输入一些文本作为上下文或条件，然后要求模型输出一些文本。这一框架为预训练和微调提供了一致的训练目标。具体来说，无论任务是什么，模型都是以最大似然目标（使用 "教师强迫"（Williams 和 Zipser，1989 年））进行训练的。为了指定模型应该执行的任务，我们会在原始输入序列中添加一个特定任务（文本）前缀，然后再将其输 入 模型。

例如，如果要求模型将句子 "That is good. "从英语翻译成德语，就会向模型输入 "translate English to German: That is good."（将英语翻译成德语：That is good.）的序列，然后训练模型输出 "Das ist gut"（Das ist gut）。对于文本分类任务，模型只需预测与目标标签相对应的单词即可。例如，在 MNLI 基准（Williams 等人，2017 年）中，目标是预测一个前提是暗示（"entailment"）、矛盾（"contradiction"

）还是两者都不是（"neutral"）一个假设。经过我们的预处理，输入序列变成了"mnli 前提：我讨厌鸽子：我对鸽子的感情充满敌意。"相应的目标词为"entailment"。需要注意的是，如果我们的模型在文本分类任务中输出的文本与任何可能的标签都不对应（例如，如果模型输出 "汉堡包"，而任务中唯一可能的标签是 "引申"、"中性 "或 "矛盾"），就会出现问题。在这种情况下，我们总是将模型的输出算作错误，尽管我们从未在任何训练过的模型中观察到这种行为。请注意，用于特定任务的文本前缀选择本质上是一个超参数；我们发现改变前缀的确切措辞影响有限，因此没有对不同的前缀选择进行大量实验。我们的文本到文本框架图，其中包含一些输入/输出信息

示例如图 1 所示。我们在附录 D 中提供了所研究的每一项任务的预处理输入的完整示例。

我们的文本到文本框架沿袭了之前将多个 NLP 任务转换成通用格式的工作：McCann 等人（2018 年）提出了 "自然语言十项全能"（Natural Language Decathlon）基准，该基准使用一致的问题解答格式来处理十项 NLP 任务。自然语言十项全能竞赛还规定，所有模型都必须是多任务模型，即能够同时处理所有任务。而我们则允许在每个任务中分别对模型进行微调，并使用简短的任务前缀而不是明确的问答格式。Radford 等人（2019）通过将一些输入作为前缀输入模型，然后对输出进行自回归采样，评估了语言模型的零点学习能力。例如，自动摘要的方法是在输入文档后加上 "TL;DR:"（"too long, didn't read "的缩写，一种常用缩写），然后通过自回归解码预测摘要。我们主要考虑的是先用编码器明确处理输入，然后再用单独的解码器生成输出的模型，我们关注的是迁移学习而不是零点学习。最后，Keskar 等人（2019b）将许多 NLP 任务统一为 "跨度提取"，即在输入中附加与可能的输出选择相对应的文本，然后训练模型以提取与正确选择相对应的输入跨度。相比之下，我们的框架还适用于机器翻译和抽象概括等生成任务，在这些任务中，不可能列举所有可能的输出选择。

除了 STS-B（一项回归任务，目标是预测 1 到 5 之间的相似度得分）之外，我们可以将所有考虑过的任务直接转换成文本到文本的格式。我们发现，这些分数大多以 0.2 为增量进行注释，因此我们只需将任何分数四舍五入到最接近的 0.2 增量，然后将结果转换为数字的字面字符串表示（例如，浮点数值 2.57 将被映射为字符串 "2.6"）。在测试时，如果模型输出的字符串对应的数字在 1 到 5 之间，我们就将其转换为浮点数值；否则，我们就将模型的预测视为不正确。这实际上是将 STS-B 回归问题转换成了 21 类分类问题。

另外，我们还将 Winograd 任务（GLUE 中的 WNLI、Super- GLUE 中的 WSC 以及我们添加到 SuperGLUE 中的 DPR 数据集）转换为更适合文本到文本框架的简单格式。Winograd 任务中的例子包括一段文本，其中包含一个模棱两可的代词，该代词可以指代段落中的多个名词短语。例如，这段文字可能是 "市议员拒绝向示威者发放许可证，因为他们害怕暴力"，其中包含的模糊代词 "他们 "可以指 "市议员 "或 "示威者"。我们将 WNLI、WSC 和 DPR 任务作为文本到文本问题，突出显示文本段落中的模糊

代词，并要求模型预测其所指的名词。上面提到的例子将转换为输入 "市议员拒绝向示威者发放许可证，因为他们*害怕暴力"，然后训练模型预测目标文本 "市议员"。

对于 WSC，示例包含段落、模糊代词、候选名词以及反映候选名词是否与代词匹配（忽略冠词）的 "真/假 "标签。我们只对带有 "真 "标签的示例进行训练，因为我们不知道带有 "假 "标签示例的正确名词目标。在评估时，如果出现以下情况，我们会赋予 "真 "标签

如果模型输出中的单词是候选名词短语中单词的子集（反之亦然），则赋予 "False "标签，否则赋予 "False "标签。这移除了大约一半的 WSC 训练集，但 DPR 数据集增加了大约 1,000 个代词解析示例。DPR 中的示例都标注了正确的指代名词，因此很容易以上述格式使用该数据集。

WNLI 训练集和验证集与 WSC 训练集有很大的重叠。为了避免验证示例泄漏到我们的训练数据中（这是第 3.5.2 节的多任务实验中的一个特殊问题），我们从不在 WNLI 上进行训练，也从不报告 WNLI 验证集上的结果。忽略 WNLI 验证集上的结果是标准做法（Devlin 等人，2018 年），因为相对于训练集而言，验证集具有 "对抗性"，即验证示例都是带有相反标签的训练示例的轻微扰动版本。因此，我们在报告验证集的平均 GLUE 分数时，并不包括 WNLI（第 3.7 节除外，该节介绍了测试集的结果）。将示例从 WNLI 转换为上述 "参照名词预测 "变体的过程比较复杂；我们将在附录 B 中介绍这一过程。

3. 实验

NLP 迁移学习的最新进展来自于各种各样的发展，例如新的预训练目标、模型架构、无标注数据集等等。在本节中，我们将对这些技术进行实证调查，希望能找出它们的贡献和意义。然后，我们将结合所获得的见解，在我们考虑的许多任务中达到最先进的水平。由于针对 NLP 的迁移学习是一个快速发展的研究领域，我们无法在实证研究中涵盖所有可能的技术或想法。如需更广泛的文献综述，我们推荐 Ruder 等人（2019 年）的最新调查报告。

我们采用合理的基线（见第 3.1 节），每次改变一个方面的设置，从而系统地研究这些贡献。例如，在第 3.3 节中，我们测量了不同无监督目标的性能，而实验管道的其他部分保持不变。这种 "坐标上升 "方法可能会遗漏二阶效应（例如，某些特定的无监督目标可能在比我们的基线设置更大的模型上效果最佳），但对我们研究中的所有因素进行组合探索将耗资巨大。在未来的工作中，我们希望能更全面地考虑我们所研究的各种方法的组合，这样会更有成效。

我们的目标是在尽可能多的因素保持不变的情况下，对各种不同任务的不同方

法进行比较。为了实现这一目标，在某些情况下，我们并不完全照搬现有的方法。例如，像 BERT（Devlin 等人，2018 年）这样的 "纯编码器 "模型旨在对每个输入标记或整个输入序列进行单一预测。这使得它们适用于分类或跨度预测任务，而不适用于翻译或抽象概括等生成任务。因此，我们所考虑的模型架构没有一个与 BERT 相同，也没有一个只包含编码器的结构。相反，我们测试的方法在精神上是相似的--例如，我们考虑了与 BERT 的 "遮蔽语言建模 "目标类似的目标，即

在第 3.3 节中，我们考虑了一种在文本分类任务中与 BERT 表现类似的模型架构；在第 3.2 节中，我们考虑了一种在文本分类任务中与 BERT 表现类似的模型架构。

在下一小节概述基线实验设置之后，我们将对模型架构（第 3.2 节）、无监督目标（第 3.3 节）、预训练数据集（第 3.4 节）、转移方法（第 3.5 节）和扩展（第 3.6 节）进行实证比较。在本节的最后，我们将研究结果与规模相结合，从而在我们考虑的许多任务中获得最先进的结果（第 3.7 节）。

3.1 基线

我们的基线目标是反映典型的现代实践。我们使用一个简单的去噪目标对标准变换器（见第 2.1 节）进行预训练，然后分别对每个下游任务进行微调。我们将在以下几个小节中介绍这一实验设置的细节。

3.1.1 模型

在我们的模型中，我们使用了 Vaswani 等人（2017 年）提出的标准编码器-解码器 Transformer。虽然许多用于 NLP 迁移学习的现代方法都使用仅由单个 "堆栈 "组成的 Transformer 架构（例如，用于语言建模（Radford 等人，2018 年；Dong 等人，2019 年）或分类和跨度预测（Devlin 等人，2018 年；Yang 等人，2019 年）），但我们发现，使用标准编码器-解码器结构在生成和分类任务中都取得了很好的效果。我们将在第 3.2 节中探讨不同模型架构的性能。

我们的基线模型在设计上使编码器和解码器在大小和配置上与 "$\text{BERT}_{\text{BASE}}$ "（Devlin 等人，2018 年）堆栈相似。具体来说，编码器和解码器都由 12 个块组成（每个块包括自我注意、可选的编码器-解码器注意和前馈网络）。每个区块中的前馈网络由一个输出维度为 $d_{\text{ff}} = 3072$ 的密集层和一个 ReLU 非线性层以及另一个密集层组成。所有注意力机制的 "键 "和 "值 "矩阵的内维度为 $d_{\text{kv}} = 64$，所有注意力机制都有 12 个头。所有其他子层和嵌入层的维度为 $d_{\text{model}} = 768$。这样，一个模型总共有大约 2.2 亿个参数。这大约是 $\text{BERT}_{\text{BASE}}$ 参数数量的两倍，因为我们的基线模型包含两个层栈，而不是一个。在正则化方面，我们在模型中所有应用 dropout 的地方都使用了 0.1 的 dropout 概率。

3.1.2 培训

如第 2.4 节所述，所有任务都是文本到文本的任务。这使得我们可以始终使用标准最大似然法进行训练，即使用教师强迫（Williams 和 Zipser，1989 年）和交叉熵损失。优化时，我们使用 AdaFactor（Shazeer 和 Stern，2018 年）。在测试时，我们使用贪婪解码（即在每个时间步选择概率最高的 logit）。

在微调之前，我们在 C4 上对每个模型进行了 $2^{19} = 524{,}288$ 步的预训练。我们使用的最大序列长度为 512，批量大小为 128 个序列。在可能的情况下

我们将多个序列 "打包 "到批次[10] 的每个条目中，因此我们的批次大约包含 $2^{16} = 65,536$ 个标记。总的来说，这种批次大小和步骤数相当于对 $2^{35} \approx 34$B 个词块进行预训练。这比 BERT（Devlin 等人，2018 年）和 RoBERTa（Liu 等人，2019 年 c）少得多，前者使用了大约 1.37 亿个词库，后者使用了大约 1.5 亿个词库。2.2T 标记。仅使用 2^{35} 标记可获得合理的计算预算，同时还能提供足够的预训练量，以获得可接受的性能。我们将在第 3.6 和 3.7 节中考虑预训练对更多步骤的影响。请注意，2 个^{35} 标记只涵盖了整个 C4 数据集的一小部分，因此我们在预训练过程中从不重复任何数据。

在预训练期间，我们使用 "反平方根 "学习率计划：$1/\sqrt{\max(n, k)}$ 其中，n 是当前的训练迭代次数，k 是预热步数（在所有实验中均设为 10^4 ）。这就为前 10 个^4 步设置了 0.01 的恒定学习率，然后学习率呈指数衰减，直到预训练结束。我们还尝试过使用三角形学习率（Howard 和 Ruder，2018 年），效果略好，但需要提前知道训练步骤的总数。由于我们将在某些实验中改变训练步数，因此我们选择了更通用的反平方根计划。

在所有任务中，我们的模型都是按照 $2^{18} = 262,144$ 步进行微调的。选择这个值的原因是在高资源任务（即数据集较大的任务）和低资源任务（数据集较小的任务）之间进行权衡，前者可从额外的微调中获益，而后者则很快就会过拟合。在微调过程中，我们继续使用 128 个长度为 512 的序列批次（即每个批次 2^{16} token）。微调时，我们使用 0.001 的恒定学习率。我们每 5000 步保存一个检查点，并在验证性能最高的模型检查点上报告结果。对于在多个任务上进行微调的模型，我们为每个任务独立选择最佳检查点。除第 3.7 节中的实验外，我们在验证集上报告所有实验结果，以避免在测试集上进行模型选择。

3.1.3 词汇

我们使用 SentencePiece（Kudo 和 Richardson，2018 年）将文本编码为 WordPiece 标记（Sennrich 等人，2015 年；Kudo，2018 年）。在所有实验中，我们使用了 32,000 个单词片的词汇量。由于我们最终会在英语到德语、法语和罗马尼亚语的翻译中对我们的模型进行微调，因此我们也要求我们的词汇涵盖这些非英语语言。为此，我们将 C4 中使用的 Common Crawl scrape 中的页面分类为德语、法语和罗马尼亚语。然后，我们在 10 部分英语 C4 数据与各 1 部分德语、法语或罗马尼亚语分类数据的混合数据上训练我们的 SentencePiece 模型。这一词汇在模型的输入和输出

中共享。请注意，我们的词汇表使得我们的模型只能处理一组预定的、固定的语言

。

3.1.4 无监督目标

利用无标记数据预训练我们的模型，需要一个不需要标记的目标，但（从广义上说）

要向模型传授可通用的知识，这些知识将被用于

10. https://www.pydoc.io/pypi/tensor2tensor-1.5.7/autoapi/data_generators/generator_utils/index.html#data_generators.generator_utils.pack_examples

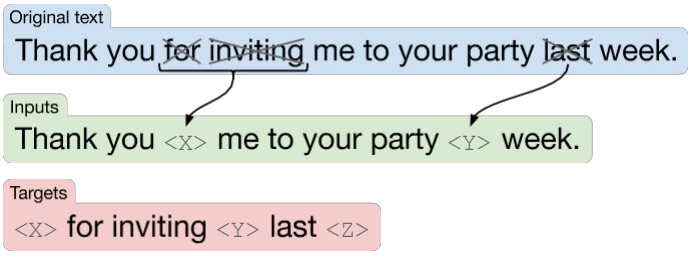

图 2：我们在基线模型中使用的目标示意图。在这个例子中，我们处理的句子是 "谢谢你上周邀请我参加你的聚会"。我们会随机选择 "感谢"、"邀请 "和 "最后"（标记为 x）这三个词进行破坏。每个连续跨度的损坏标记都由一个哨兵标记（显示为 <X> 和 <Y>）替换，该标记在整个示例中是唯一的。由于 "for "和 "inviting "是连续出现的，因此它们会被一个哨兵标记 <X> 代替。然后，输出序列由被替换掉的跨度组成，以输入中用来替换它们的哨兵标记和最后一个哨兵标记 <Z> 为界。

在下游任务中非常有用。将预训练和微调模型所有参数的迁移学习范式应用于 NLP 问题的初步工作使用因果语言建模目标进行预训练（Dai 和 Le，2015 年；Peters 等人，2018 年；Radford 等人，2018 年；Howard 和 Ruder，2018 年）。然而，最近的研究表明，"去噪 "目标（Devlin 等人，2018 年；Taylor，1953 年）（也称为 "屏蔽语言建模"）能产生更好的性能，因此很快成为标准目标。在去噪目标中，对模型进行训练，以预测输入中丢失或损坏的标记。受 BERT 的 "掩蔽语言建模 "目标和 "词丢弃 "正则化技术（Bowman 等人，2015 年）的启发，我们设计了一种目标，即随机抽样，然后丢弃输入序列中 15% 的标记。所有连续的丢弃标记都由一个哨兵标记代替。每个哨兵标记都会被分配一个序列中唯一的标记 ID。哨兵 ID 是添加到我们词汇表中的特殊标记，与任何词块都不对应。然后，目标词对应于所有被删除的跨度较大的词组，这些词组由输入序列中使用的相同哨兵词组以及最后一个哨兵词组划分开来，以标记目标序列的结束。我们之所以选择屏蔽连续的标记符跨度并只预测退出的标记符，是为了降低预训练的计算成本。我们将在第 3.3 节中对预训练目标进行深入研究。图 2 显示了应用该目标后的转换示例。在第 3.3 节中，我们将该目标与许多其他变体进行了实证比较。

3.1.5 基准性能

在本节中，我们将介绍使用上述基准实验程序得出的结果，以了解在我们的下游任务套件中可以期待什么样的性能。理想情况下，我们会多次重复研究中的每项实验，以获得结果的置信区间。遗憾的是，这样做的成本过高，因为大量的

	胶水	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
⋆ 基准平均值	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
基线标准偏差	0.235	0.065	0.343	0.416	0.112	*0.090*	0.108
无需预先培训	66.22	17.60	50.31	53.04	25.86	**39.77**	24.04

表 1：基线模型和训练程序的平均得分和标准偏差。为了便于比较，我们还报告了从零开始（即不进行任何预训练）对每个任务进行训练时的性能，训练步骤数与微调基线模型所用步骤数相同。本表（以及本文中除表 14 以外的所有表格）中的所有分数都是在每个数据集的验证集上报告的。

我们运行的实验次数。作为一种更便宜的替代方法，我们从头开始训练基准模型 10 次（即使用不同的随机初始化和数据集洗牌），并假定这些基准模型运行的方差也适用于每个实验变体。我们并不指望我们所做的大多数改变会对运行间方差产生巨大影响，因此这应该能合理地说明不同改变的重要性。另外，我们还测量了在不进行预训练的情况下，在所有下游任务上训练模型 2^{18} 步（与微调时使用的步数相同）的性能。这样，我们就能了解在基线设置中，预训练对我们的模型有多大益处。

在正文中报告结果时，为了节省篇幅和便于解释，我们只报告了所有基准的 子集分数。对于 GLUE 和 SuperGLUE，我们在 "GLUE "和 "SGLUE "标题下报告了所有子任务的平均得分（根据官方基准的规定）。对于所有翻译任务，我们报告的是 SacreBLEU v1.3.0 （Post，2018 年）提供的 BLEU 分数（Papineni 等人，2002 年），其中包含 "exp "平滑和 "intl "标记化。我们将 WMT 英语到德语、英语到法语和英语到罗马尼亚语的得分分别称为 EnDe、EnFr 和 EnRo。对于 CNN/每日邮报，我们发现模型在 ROUGE-1-F、ROUGE-2-F 和 ROUGE-L-F 指标（Lin，2004 年）上的 表现高度相关，因此我们在 "CNNDM "标题下单独报告 ROUGE-2-F 分数。同样，对于 SQuAD，我们发现 "精确匹配 "和 "F1 "得分的性能高度相关，因此我们单独报告 "精确匹配 "得分。我们在附录 E 表 16 中提供了所有实验中每项任务的每项得分。

我们的结果表均采用这样的格式，即每行对应一个特定的实验配置，每列给出每个基准的分数。我们将在大多数表格中列出基准配置的平均性能。基准配置
出现的配置，我们将用 ⋆ 标记（如表 1 第一行）。此外，我们还将用**粗体**字标出任何得分在两个标准差以内的最大（最佳）得分。

给定的实验。

我们的基准结果如表 1 所示。总体而言，我们的结果可与规模相似的现有模型相媲美。例如，$\text{BERT}_{\text{BASE}}$ 在 SQuAD 上的精确匹配分数为 80.8，在 MNLI 匹配上的准确率为 84.4，而我们分别达到了 80.88 和 84.24（见表 16）。需要注意的是，我们不能直接将我们的基线与 $\text{BERT}_{\text{BASE}}$ 进行比较，因为我们的基线是一个编码器-解码器模型，预训练的步数大约为 $1/4$。 不出所料，我们发现预训练在几乎所有基准测试中都有显著提高。唯一的例外是 WMT 英语到法语，这是一个大型的

足够多的数据集，预训练的收益往往微乎其微。我们在实验中加入了这项任务，以测试迁移学习在高资源条件下的表现。由于我们通过选择表现最好的检查点来执行早期停止，因此我们的基线与 "无预训练 "之间的巨大差距凸显了预训练对有限数据任务的性能改善程度。虽然我们在本文中没有明确衡量数据效率的 提高，但我们强调这是迁移学习范式的主要优势之一。

至于运行间差异，我们发现大多数任务的运行间标准偏差都小于任务基准分数的 1%。例外情况包括 CoLA、CB 和 COPA，它们都是 GLUE 和 SuperGLUE 基准中的低资源任务。例如，在 CB 任务中，我们的基线模型的平均 F1 得分为 91.22，标准偏差为 3.237（见表 16），部分原因可能是 CB 的验证集仅包含 56 个示例。请注意，GLUE 和 SuperGLUE 分数是根据每个基准任务的平均分数计算的。因此，我们提醒大家，CoLA、CB 和 COPA 的运行间方差较大，因此很难单独使用 GLUE 和 SuperGLUE 分数对模型进行比较。

3.2 建筑

虽然 Transformer 最初采用的是编码器-解码器架构，但现代 NLP 迁移学习的许多工作都采用了其他架构。在本节中，我们将对这些架构变体进行回顾和比较。

3.2.1 模型结构

区分不同架构的一个主要因素是模型中不同注意力机制所使用的 "掩码"。回想一下，变换器中的自我注意操作将一个序列作为输入，并输出一个相同长度的新序列。输出序列的每个条目都是通过计算输入条目的加权平均值产生的。

序列。具体来说，让 y_i 指输出序列的第 i 个元素，x_j 指输入序列的*第* j *个*条目。y_i 的计算公式为 $\sum_j w_{i,j} x_j$，其中 $w_{i,j}$ 是标量。自我注意力机制产生的权重与 x_i 和 x_j 的函数关系。注意力然后使用掩码将某些权重清零，以限制在给定的输出时间步中可以关注输入的哪些条目。我们将考虑的掩码示意图如图 3 所示。例如，因果掩码（图 3，中间）会在 $j > i$ 时将任何 $w_{i,j}$ 设为零。

我们考虑的第一个模型结构是编码器-解码器转换器，它由两层堆栈组成：编码器接收输入序列，解码器产生新的输出序列。这种结构变体的示意图如图 4 左侧

所示。

编码器使用 "完全可见 "注意力掩码。完全可见掩码允许自我注意机制在产生输出的 每个条目时，注意输入的任何条目。我们可以在图 3 左侧看到这种屏蔽模式。这种形式的掩蔽适用于关注 "前缀"，即提供给模型的某些上下文，这些上下文随后将用于预测。BERT（Devlin 等人，2018 年）也使用了完全可见的屏蔽模式，并在输入中附加了一个特殊的 "分类 "标记。BERT 的输出

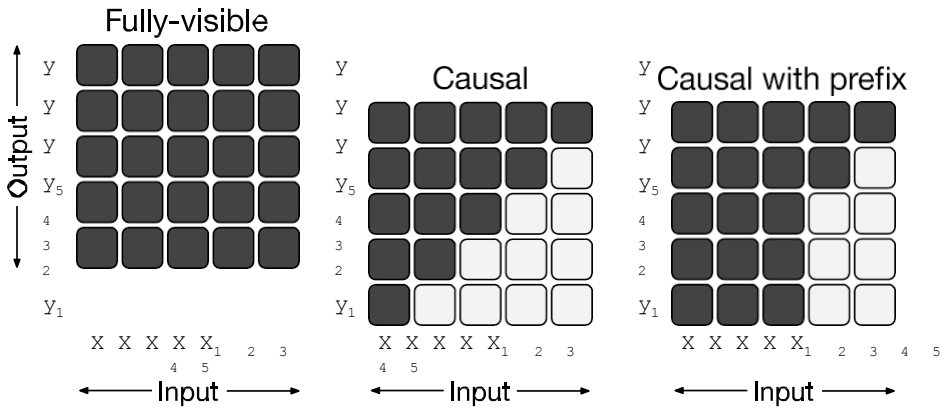

图 3：代表不同注意力掩码模式的矩阵。自我注意机制的输入和输出分别以 x 和 y *表示。第* i 行和*第* j 列的 深色单元格表示允许自我注意机制在输出时间步 i *注意*输入元素 j。左图：完全可见的掩码允许自注意机制在每个输出时间步关注全部输入。中间因果掩码可防止第 i 个输出元素依赖于 "未来 "的任何输入元素。右图带前缀的因果掩码允许自注意机制对部分输入序列使用完全可见掩码。

然后，在与分类标记相对应的时间步上，使用该标记对输入序列进行分类预测。

变换器解码器中的自我关注操作使用 "因果 "屏蔽模式。在生成输出序列的第 i *个*条目时，因果掩蔽会阻止模型关注输入序列中 j > *i 的第 j 个*条目。这在训练过程中使用，以便模型在生成输出时无法 "预见未来"。这种屏蔽模式的注意力矩阵如图 3 中所示。

编码器-解码器转换器中的解码器用于自回归产生输出序列。也就是说，在每个输出时间步，从模型的预测分布中抽取一个标记，然后将样本反馈到模型中，对下一个输出时间步进行预测，以此类推。因此，Transformer 解码器（无编码器）可用作语言模型（LM），即仅为下一步预测而训练的模型（Liu 等人，2018；Radford 等人，2018；Al-Rfou 等人，2019）。这构成了我们考虑的第二种模型结构。这种结构的示意图如图 4 中所示。事实上，早期针对 NLP 的迁移学习工作就采用了这种以语言建模为目标的架构作为预训练方法（Radford 等人，2018 年）。

语言模型通常用于压缩或序列生成（Graves，2013 年）。不过，只需将输入和目标

连接起来，它们也可[illegible]。举例来说，考虑英语到德语的翻译：如果我们有一个包含输入句子 "That is good. "和目标句子 "Das ist gut. "的训练数据点，那么我们只需在 "英译德：That is good.目标：Das ist gut. "的串联输入序列上对模型进行下一步预测训练即可：Das ist gut."如果我们想

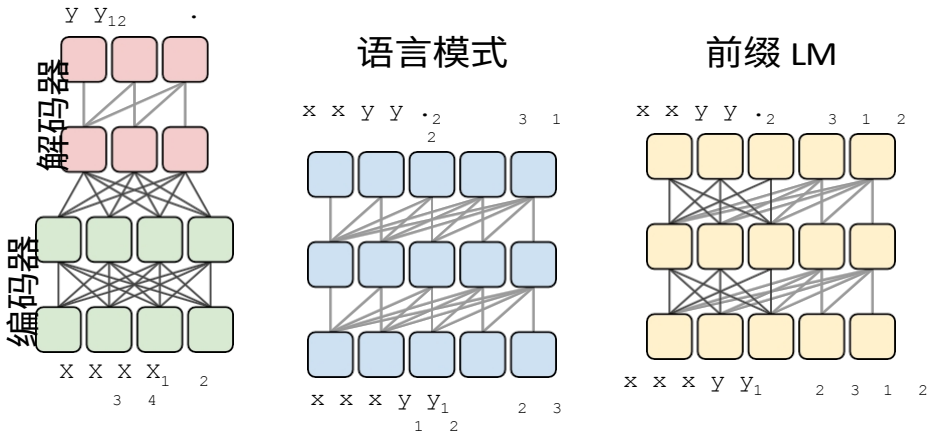

图 4：我们考虑的变形器架构变体示意图。在该图中，块代表序列中的元素，线代表注意力的可见性。不同颜色的块组表示不同的变换器层叠。深灰色线条对应完全可见遮蔽，浅灰色线条对应因果遮蔽。我们用". "表示特殊的序列结束标记，代表预测的结束。输入和输出序列分别用 x 和 y 表示。左图：标准的编码器-解码器架构在编码器和编码器-解码器注意力中使用完全可见掩码，在解码器中使用因果掩码。中间语言模型由单个变换器层栈组成，输入是输入和目标的连接，全程使用因果掩码。右图在语言模型中添加前缀相当于允许对输入进行完全可见的屏蔽。

在此示例中，要获得模型的预测结果，需要向模型输入前缀 "将英语翻译成德语：很好，目标："，并要求模型自回归生成序列的其余部分。这样，模型就能根据输入预测输出序列，从而满足文本到文本任务的需求。这种方法最近被用于证明语言模型可以在没有监督的情况下学习执行某些文本到文本的任务（Radford 等人，2019）。

在文本到文本（text-to-text）设置中使用语言模型的一个基本且经常被提及的缺点是，因果掩蔽迫使模型对输入序列*第 i 个*条目的表征只能依赖于*第 i 个条目*之前的条目。在完全因果掩蔽的情况下，模型对前缀状态的表示只能取决于前缀的先前条目。因此，在预测输出条目时，模型将关注前缀的表征，而这是不必要的限制。类似的论点也反对在序列到序列模型中使用单向递归神经网络编码器（Bahdanau

等人，2015 年）。 探索迁移学习的极限

在基于变换器的语言模型中，只需改变屏蔽模式就能避免这一问题。我们不使用因果掩码，而是在序列的前缀部分使用完全可见的掩码。图 3 和图 4 的最右侧面板分别展示了这种屏蔽模式和由此产生的 "前缀 LM"（我们考虑的第三种模型结构）的示意图。在上文提到的英译德例子中，"将英语翻译成德语：That is good. target: "的前缀将采用完全可见屏蔽，而因果屏蔽将在预测目标词 "Das ist gut "的训练中使用。在文本到文本框架中使用前缀 LM 最初是由 Liu 等人（2018）提出的。最近，Dong 等人（2019）的研究表明，这种架构在各种文本到文本任务中都很有效。这种架构类似于编码器-解码器模型，编码器和解码器共享参数，编码器-解码器的注意力被输入和目标序列的完全注意力所取代。

我们注意到，在遵循文本到文本框架时，前缀 LM 架构与分类任务的 BERT（Devlin 等人，2018 年）非常相似。要了解原因，请看 MNLI 基准中的一个例子，前提是 "我讨厌鸽子"，假设是 "我对鸽子的感情充满敌意"，正确标签是 "entailment"。要将此示例输入语言模型，我们需要将其转换为 "mnli premise: I hate pigeons. hypothesis：目标：谓词"。在这种情况下，完全可见的前缀将对应整个输入序列，直到 "target: "一词，这可以被看作是类似于 BERT 中使用的 "分类 "标记。因此，我们的模型可以完全看到整个输入，然后通过输出 "entailment "一词来完成分类任务。根据任务前缀（本例中为 "mnli"），模型很容易学会输出其中一个有效的类标签。因此，前缀 LM 与 BERT 架构的主要区别在于，分类器被简单地集成到了前缀 LM 中变换解码器的 输出层中。

3.2.2 比较不同的模型结构

为了在实验中比较这些架构变体，我们希望我们所考虑的每个模型都能以某种有意义的方式等效。我们可以说，如果两个模型的参数数量相同，或者处理给定的（输入-序列、目标-序列）对所需的计算量大致相同，那么这两个模型就是等价的。遗憾的是，我们无法同 时 根据这两个标准来比较一个编码器-解码器模型和一个语言模型架构（由一个转换器栈组成）。要了解原因，首先要注意的是，在编码器中包含 L *层*、在解码器中包含 L *层的编码*器-解码器模型，其参数数量与包含 2L 层的语言模型大致相同。然而，相同的 $L + L$ 编码器-解码器模型的计算成本与*只有* L *层*的语言模型大致相同。这是因为语言模型中的 L 层必须*同时*应用于输入和输出序列，而编码器只应用于输

入序列，解码器只应用于输出序列。请注意，这些等价关系是近似的--由于编码器-解码器注意，解码器中存在一些额外参数，而且注意层中也存在一些计算成本，这些成本是 序列长度的二次方。然而，在实际应用中，我们观察到几乎相同的步长。

*L 层*语言模型与 *L* + *L 层*编码器-解码器模型相比，计算成本大致相当。此外，就我们考虑的模型规模而言，编码器-解码器注意层的参数数约为总参数数的 10%，因此我们做了一个简化假设，即 *L* + *L* 层编码器-解码器模型的参数数与 *2L 层语言模型的*参数数相同。

为了提供合理的比较方法，我们考虑了编码器-解码器模型的多种配置。我们将把 $\text{BERT}_{\text{BASE}}$ 大小的层堆栈中的层数和参数分别称为 *L* 和 *P*。我们将使用 *M 来表示 L* + *L 层*编码器-解码器模型或仅 *L 层*解码器模型处理给定输入-目标对所需的 FLOP 数量。总的来说，我们将比较

- 编码器-解码器模型，编码器有 *L* 层，解码器有 *L* 层。该模型有 *2P 个*参数，计算成本为 *M* FLOP。
- 一个等效模型，但编码器和解码器共享参数，导致 *P 个*参数和 *M* -FLOP 计算成本。
    - 编码器-解码器模型的编码器和解码器各有 L/2 层，因此 *P* 参数和 M/2-FLOP 成本。
- 纯解码器语言模型有 *L* 层和 *P 个*参数，计算成本为 *M* FLOPs。
- 仅有解码器的前缀 LM 具有相同的 结构（因此参数数量和计算成本也相同），但对输入具有完全可见的自注意力。

3.2.3 目标

作为无监督目标，我们将同时考虑基本语言建模目标和第 3.1.4 节中描述的基线去噪目标。由于语言建模目标历来被用作预训练目标（Dai 和 Le，2015 年；Ramachandran 等人，2016 年；Howard 和 Ruder，2018 年；Radford 等人，2018 年；Peters 等人，2018 年），而且与我们考虑的语言模型架构天然契合，因此我们将语言建模目标包括在内。对于在预测前摄取前缀的模型（编码器-解码器模型和前缀 LM），我们从未标明的数据集中采样一段文本，并随机选择一个点将其分成前缀和目标部分。对于标准语言模型，我们训练该模型来预测从头到尾的整个跨度。我们的无监督去噪目标是专为文

本到文本模型设计的；为了将其用于语言模型，我们将按照第 3.2.1 节所述的方法连接输入和目标。

3.2.4 成果

表 2 列出了我们比较的每种架构所取得的分数。在所有任务中，以去噪为目标的编码器-解码器架构表现最佳。这一变体的参数数最多（2P），但计算成本与仅有 P 个参数的解码器模型相同。令人惊讶的是，我们发现编码器和解码器共享参数的效果几乎一样好。相比之下，将编码器和解码器的层数减半

建筑学	目标	参数	费用	胶水	CNNDM	SQuAD	SGLUE	EnDe	EnFr	恩罗
⋆ 编码器-解码器	去噪	*2P*	*M*	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, 共享	去噪	*P*	*M*	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
编码-解码，6 层	去噪	*P*	M/2	80.88	18.97	77.59	68.42	26.38	38.40	26.95
语言模式	去噪	*P*	*M*	74.70	17.93	61.14	55.02	25.09	35.28	25.86
前缀 LM	去噪	*P*	*M*	81.82	18.61	78.94	68.11	26.43	37.98	27.39
编码器-解码器	LM	*2P*	*M*	79.56	18.59	76.02	64.29	26.27	39.17	26.86
Enc-dec, 共享	LM	*P*	*M*	79.60	18.13	76.35	63.50	26.62	39.17	27.05
编码-解码，6 层	LM	*P*	M/2	78.67	18.26	75.32	64.06	26.13	38.42	26.89
语言模式	LM	*P*	*M*	73.78	17.54	53.81	56.51	25.23	34.31	25.38
前缀 LM	LM	*P*	*M*	79.68	17.84	76.87	64.86	26.28	37.51	26.76

表 2：第 3.2.2 节所述不同架构变体的性能。我们用 *P* 表示 12 层基本变换器层栈中的参数数，用 *M* 表示使用编码器-解码器模型处理序列所需的 FLOP。我们使用去噪目标（见第 3.1.4 节）和自回归目标（通常用于训练语言模型）来评估每种结构变体。

在编码器和解码器堆栈中共享参数会显著降低性能。同时进行的工作（Lan 等人，2019 年）还发现，在 Transformer 块之间共享参数可以有效降低总参数数，而不会牺牲太多性能。XLNet 与以去噪为目标的共享编码器-解码器方法也有一些相似之处（Yang 等人，2019）。我们还注意到，共享参数编码器-解码器的性能优于仅有解码器的前缀 LM，这表明增加明确的编码器-解码器注意力是有益的。最后，我们证实了一个广为流传的观点，即与语言建模目标相比，使用去噪目标总是能带来更好的下游任务表现。Devlin 等人（2018 年）、Voita 等人（2019 年）以及 Lample 和 Conneau（2019 年）等人都曾提出过这一观点。我们将在下一节对无监督目标进行更详细的探讨。

3.3 无监督目标

无监督目标的选择至关重要，因为它提供了一种机制，通过这种机制，模型可以获得通用知识，并应用于下游任务。因此，人们开发了各种各样的预训练目标（Dai 和 Le，2015；Ramachandran 等人，2016；Radford 等人，2018；Devlin 等人，2018；Yang 等人，2019；Liu 等人，2019b；Wang 等人，2019a；Song 等人，2019；Dong 等人，

2019；Joshi 等人，2019）。在本节中，我们将对无监督目标的空间进行程序化探索。在很多情况下，我们不会完全照搬现有的目标--有些目标会进行修改，以适应我们的文本到文本编码器-解码器框架， 而在其他情况下，我们会使用结合了多种常见方法概念的目标。

总之，我们的所有目标都会从无标记文本数据集中摄取与标记化文本跨度相对应的标记 ID 序列。令牌序列经过处理后会产生一个（损坏的）输入序列和一个相应的目标。然后，像往常一样训练模型

目标	输入	目标
前缀语言建模	感谢您的邀请	我上周参加了你的聚会 .
BERT-style Devlin 等人（2018 年）	谢谢你<M><M>我参加了你的苹果周聚会。	*(原文)*
除尘	最后的乐趣，你邀请周 谢谢	*(原文)*
MASS-style Song et al.	谢谢你<M><M>我到你的党<M>周 。	*(原文)*
内径噪声，更换跨度	谢谢你 <X> 我参加了你的聚会 <Y> 周 。	<X>为邀请<Y>最后<Z>
I.i.d. 噪音，丢弃代币	感谢我参加你们的派对周。	为邀请最后
随机跨度	感谢 <X> 至 <Y> 周 。	<X>感谢你上次<Z>邀请我参加<Y>你的聚会

表 3：我们认为应用于输入文本 "谢谢你邀请我参加你上周的聚会 "的一些无监督目标所产生的输入和目标示例。请注意，我们的所有目标都处理*标记化*文本。对于这个特定句子，所有单词都被我们的词汇表映射为一个标记。<M> 表示共享掩码标记，<X>、<Y> 和 <Z> 表示被分配了唯一标记 ID 的哨兵标记。BERT 类型的目标（第二行）包括一种破坏，其中一些标记被随机标记 ID 所取代；我们通过灰色的单词苹果来表示这种情况。

以最大似然法预测目标序列。表 3 举例说明了我们所考虑的许多目标。

3.3.1 不同的高层方法

首先，我们比较了受常用目标启发但在方法上有显著差异的三种技术。首先，我们采用了第 3.2.3 节中使用的基本 "前缀语言建模 "目标。这种技术将一段文本分成两部分，一部分作为编码器的输入，另一部分作为解码器预测的目标序列。其次，我们考虑了受 BERT 中使用的 "掩码语言建模"（MLM）目标启发的目标（Devlin 等人，2018 年）。MLM 采用文本跨度，破坏 15% 的标记。90% 被破坏的标记被一个特殊的掩码标记替换，10% 被随机标记替换。由于 BERT 仅是一个编码器模型，因此它在预训练期间的目标是在编码器的输出端 重建掩码标记。在编码器-解码器情况下，我们只需将整个未损坏序列作为目标。请注意，这与我们的基线目标不同，后者只将损坏的标记作为目标；我们将在第 3.3.2 节中比较这两种方法。最后，我们还考虑了一个基本的解扰目标，例如（刘等人，2019a）将其应用于去噪序列自动编码器。这种方法采用标记序列，对其进行洗码，然后将原始的去洗码序列作为目标。我们在表 3 的前三行提供了这三种方法的输入和目标示例。

这三个目标的性能如表 4 所示。总体而言，尽管前缀语言建模目标在翻译任务中的表现类似，但我们发现 BERT 式目标的表现最好。事实上，BERT 目标的动机是超越基于语言模型的预训练。去修辞目标的表现比前缀语言建模目标和 BERT 式目 标 都要差很多。

目标	胶水	CNNDM	SQuAD	SGLUE	EnDe	EnFr	恩罗
前缀语言建模	80.69	18.94	77.99	65.27	**26.86**	39.73	**27.49**
BERT 式（Devlin 等人，2018 年）	**82.96**	**19.17**	**80.65**	**69.85**	**26.78**	**40.03**	**27.41**
除尘	73.17	18.59	67.61	58.47	26.11	39.30	25.62

表 4：第 3.3.1 节所述三种不同预培训目标的性能。

目标	胶水	CNNDM	SQuAD	SGLUE	EnDe	EnFr	恩罗
BERT 式（Devlin 等人，2018 年）	82.96	19.17	**80.65**	69.85	26.78	**40.03**	27.41
MASS-style（Song 等人，2019 年）	82.32	19.16	80.10	69.28	26.79	**39.89**	27.55
⋆ 替换损坏的跨度	83.28	**19.24**	**80.88**	**71.36**	**26.98**	39.82	**27.65**
丢弃损坏的代币	**84.44**	**19.31**	**80.52**	68.67	**27.07**	39.76	**27.82**

表 5：BERT 式预训练目标变体的比较。在前两个变体中，训练模型是为了重建未损坏的原始文本片段。在后两个变体中，模型只预测损坏的标记序列。

3.3.2 简化 BERT 目标

基于前一节的结果，我们现在将重点探讨对 BERT 式去噪目标的 修改。该目标最初是作为一种预训练技术提出的，用于为分类和跨度预测而训练的纯编码器模型。因此，有可能对其进行修改，使其在我们的编码器-解码器文本-文本设置中表现更好或更有效。

首先，我们考虑了 BERT 式目标的一个简单变体，即不包含随机标记交换步骤。由此产生的目标只是用一个屏蔽标记替换了输入中 15%的标记，然后训练模型重建未被破坏的原始序列。Song 等人（2019）也使用了类似的屏蔽目标，并将其称为 "MASS"，因此我们称这种变体为 "MASS-style "目标。其次，我们想知道是否有可能避免预测整个未损坏文本跨度，因为这需要解码器对长序列进行自我关注。为此，我们考虑了两种策略：首先，我们不使用掩码标记替换每个损坏标记，而是使用唯一的掩码标记替换损坏标记的每个连续跨度的全部内容。然后，目标序列就变成了 "损坏 "跨度的连接，每个跨度的前缀都是输入中用来替换它的掩码标记。这就是我们在基线中使用的预训练目标，详见第 3.1.4 节。其次，我们还考虑了一种变体，即从输入序列中完全删

除损坏的标记，然后让模型按顺序重建被删除的标记。这些方法的示例见表 3 的第五和第六行。

表 5 显示了原始 BERT 式目标与这三种备选方案的经验比较。我们发现，在我们的环境中，所有这些变体的表现都差不多。唯一的例外是，完全丢弃已损坏的标记会使 GLUE 得分略有提高，这要归功于 CoLA 得分的显著提高（60.04，与我们的

腐败率	胶水	CNNDM	SQuAD	SGLUE	EnDe	EnFr	恩罗
10%	**82.82**	19.00	**80.38**	69.55	**26.87**	39.28	**27.44**
⋆ 15%	**83.28**	19.24	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
25%	**83.00**	**19.54**	**80.96**	70.48	**27.04**	**39.83**	**27.47**
50%	81.27	19.32	79.80	70.33	**27.01**	**39.90**	**27.49**

表 6： 不同腐败率下 i.i.d. 腐败目标的绩效。

表 16）。这可能是由于 CoLA 涉及对给定句子在语法和句法上是否可接受进行分类，而判断标记符是否丢失与检测可接受性密切相关。然而，在 SuperGLUE 中，完全丢弃标记比用哨兵标记来替换标记的效果要差。不需要预测完整原始序列的两种变体（"替换已损坏的跨度 "和 "丢弃已损坏的跨度"）都具有潜在的吸引力，因为它们可以缩短目标序列，从而加快训练速度。今后，我们将探索用哨兵标记替换已损坏跨度并只预测已损坏标记的变体（如我们的基线目标）。

3.3.3 改变腐败率

到目前为止，我们一直在损坏 15% 的标记，这是 BERT（Devlin 等人，2018 年）中使用的值。同样，由于我们的文本到文本框架与 BERT 的不同，我们有兴趣看看不同的损坏率是否对我们更有效。我们在表 6 中比较了 10%、15%、25% 和 50% 的损坏率。总体而言，我们发现腐败率对模型性能的影响有限。唯一的例外是，我们考虑的最大损坏率（50%）导致 GLUE 和 SQuAD 的性能显著下降。使用较大的损坏率还会导致目标时间变长，从而有可能减慢训练速度。基于这些结果和 BERT 的历史先例，我们今后将使用 15% 的损坏率。

3.3.4 腐蚀跨度

我们现在的目标是通过预测更短的目标来加快训练速度。我们迄今为止使用的方法是对每个输入标记进行 i.i.d. 判断，决定是 否 对其进行破坏。当多个连续标记被破坏时，它们将被视为一个 "跨度"，并使用一个唯一的掩码标记来替换整个跨度。用单个标记替换整个跨度会将未标记的文本数据处理成更短的序列。由于我们使用的是 i.i.d. 破坏策略，因此并不总是会连续出现大量被破坏的标记。因此，我们可以通过对标记的跨度进行特定的破坏，而不是以 i.i.d. 的方式对单个标记进行破坏，来获得额外的速度

提升。在此之前，我们也曾考虑过将破坏跨度作为 BERT 的预训练目标，结果发现它可以提高性能（Joshi 等人，2019 年）。

为了验证这一想法，我们考虑了一个专门破坏连续、随机间隔的标记间隔的目标。这个目标的参数可以是被破坏的标记的比例和被破坏跨度的总数。然后选择跨度长度

跨度长度	胶水	CNNDM	SQuAD	SGLUE	EnDe	EnFr	恩罗
⋆ 基线 (i.i.d.)	**83.28**	19.24	80.88	71.36	**26.98**	**39.82**	**27.65**
2	**83.54**	19.39	**82.09**	**72.20**	**26.76**	**39.99**	**27.63**
3	**83.49**	**19.62**	**81.84**	**72.53**	**26.86**	39.65	**27.62**
5	**83.40**	19.24	**82.05**	**72.23**	**26.88**	39.40	**27.53**
10	82.85	19.33	**81.84**	70.44	**26.79**	39.49	**27.69**

表 7：不同平均跨度长度下跨度破坏目标的性能（受 Joshi 等人（2019）的启发）。在所有情况下，我们破坏了 15%的原始文本序列。

以满足这些指定参数的要求。例如，如果我们正在处理一个包含 500 个标记的序列，并指定 15%的标记应被损坏，且总跨度应为 25，那么被损坏的标记总数为 500 × 0.15 = 75，平均跨度长度为 *75/25* = 3。请注意，在给定原始序列长度和损坏率的情况下，我们可以用平均跨度长度或跨度总数来等价地对这一目标进行参数化。

我们在表 7 中比较了跨度腐败目标和 i.i.d 腐败目标。我们在所有情况下都使用 15% 的腐败率，并使用 2、3、5 和 10 的平均跨度长度进行比较。我们再次发现，这些目标之间的差异有限，尽管在某些情况下，平均跨度为 10 的版本略低于其他值。我们还特别发现，在大多数非翻译基准上，使用平均跨度长度为 3 的目标略微（但明显）优于 i.i.d 目标。幸运的是，与 i.i.d. 噪声方法相比，跨度破坏目标在训练过程中也提供了一定的速度，因为跨度破坏平均会产生更短的序列。

3.3.5 讨论

图 5 显示了我们在探索无监督目标过程中所做选择的流程图。总体而言，我们观察到的最显著的性能差异是，去噪目标在预训练中的表现优于语言建模和去重。在我们探索的多种去噪目标中，我们没有观察到明显的差异。但是，不同的目标（或目标的参数化）会导致不同的序列长度，从而导致不同的训练速度。这就意味着，在我们所考虑的去噪目标中进行选择时，应主要考虑其计算成本。我们的结果还表明，对于我们所考虑的任务和模型来说，对与我们在此考虑的目标类似的目标进行额外探索可能不会带来显著的收益。相反，探索完全不同的利用无标记数据的方法可能是偶然的。

3.4 预训练数据集

与无监督目标一样，预训练数据集本身也是迁移学习管道的 重要组成部分。然而，与目标和基准不同的是，新的预训练数据集通常不会被视为对自身有重大贡献，而且往往不会

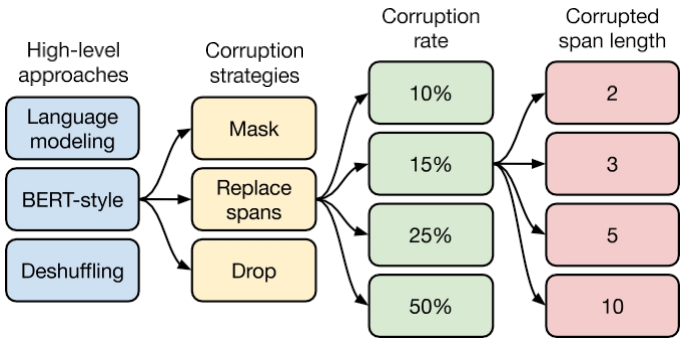

图 5：我们探索无监督目标的流程图。在第 3.3.1 节中，我们首先考虑了几种不同的方法，发现 BERT 类型的去噪目标表现最佳。然后，我们在第 3.3.2 节中考虑了各种简化 BERT 目标的方法，使其产生更短的目标序列。鉴于用哨兵标记替换掉线跨度的效果很好，并能产生较短的目标序列，我们在第 3.3.3 节中试验了不同的损坏率。最后，我们将在第 3.3.4 节中对故意破坏连续标记的目标进行评估。

与预训练模型和代码一起发布。相反，它们通常是在介绍一种新方法或模型的过程中引入的。因此，不同预训练数据集之间的比较相对较少，也缺乏用于预训练的 "标准 " 数据集。最近一些值得注意的例外（Baevski 等人，2019；Liu 等人，2019c；Yang 等人，2019）比较了在新的大型（通常是 Common Crawl 源）数据集和使用较小的已有数据集（通常是维基百科）上进行预训练的情况。为了更深入地探究预训练数据集对性能的影响，我们在本节中比较了 C4 数据集的变体和其他潜在的预训练数据源。我们将考虑的所有 C4 数据集变体作为 TensorFlow Datasets 的一部分发布。[11]

3.4.1 无标签数据集

在创建 C4 的过程中，我们开发了各种启发式方法来过滤从 Common Crawl 中提取的网络文本（参见第 2.2 节的描述）。除了与其他过滤方法和常见的预训练数据集进行比较外，我们还想衡量这种过滤方法是否能提高下游任务的性能。为此，我们在以下数据集上比较了预训练后基线模型的性能：

C4 作为基线，我们首先考虑在第 2.2 节所述的拟议无标记数据集上进行预训练。

未经过滤的 C4 为了衡量我们在创建 C4 时使用的启发式过滤（重复数据删除、删除坏词、只保留句子等）的效果，我们还生成了一个放弃这种过滤的 C4 替代版本。需要注意的是，我们仍然使用 `langdetect`

11. https://www.tensorflow.org/datasets/catalog/c4

来提取英文文本。因此，我们的 "未过滤 "变体仍然包含一些过滤，因为 `langdetect` 有时会对非自然英语文本赋予较低的概率。

RealNews-like 最近的研究使用了从新闻网站中提取的文本数据（Zellers 等人，2019；Baevski 等人，2019）。为了与这种方法进行比较，我们通过对 C4 进行额外过滤，使其仅包含来自 "RealNews "数据集（Zellers 等人，2019 年）中使用的一个域的内容，从而生成另一个未标记的数据集。请注意，为了便于比较，我们保留了 C4 中使用的启发式过滤方法；唯一不同的是，我们表面上省略了任何非新闻内容。

类似于 **WebText** 同样，WebText 数据集（Radford 等 人，2019 年）只使用提交给内容聚合网站 Reddit 并获得至少 3 分 "评分 "的网页内容。使用 Reddit 分数作为质量信号的理念是，该网站的用户只会对高质量的文本内容进行向上投票。为了生成一个可比较的数据集，我们首先尝试从 C4 中删除所有不是来自 OpenWebText 所准备的列表中出现的 URL 的内容。[12] 不过，这样做的结果是内容相对较少，只有大约 2 GB，因为大多数网页从未在 Reddit 上出现过。回想一下，C4 是基于一个月的 Common Crawl 数据创建的。为了避免使用过小的数据集，我们从 Common Crawl 下载了从 2018 年 8 月到 2019 年 7 月的 12 个月数据，对 C4 应用启发式过滤，然后应用 Reddit 过滤。这产生了一个 17GB 的 WebText 类数据集，其大小与原始的 40GB WebText 数据集相当（Radford 等人，2019 年）。

维基百科 维基百科网站由数百万篇合作撰写的百科全书文章组成。网站上的内容受严格的质量准则约束，因此被用作干净自然文本的可靠来源。我们使用来自 TensorFlow Datasets 的英文维基百科文本数据，[13] ，其中省略了文章中的任何标记或参考文献部分。

维基百科 + 多伦多图书语料库 使用维基百科预训练数据的一个缺点是，它只代表了自然文本的一个可能领域（百科全书文章）。

为了缓解这一问题，BERT（Devlin 等人，2018 年）将维基百科的数据与多伦多图书语料库（TBC）（Zhu 等人，2015 年）相结合。TBC 包含从电子书中提取

的文本，代表了不同的自然语言领域。BERT 的流行使得维基百科 + TBC 的组合被许多后续研究采用。

表 8 显示了在每个数据集上进行预训练后取得的结果。第一个显而易见的结论是，去掉 C4 中的启发式过滤会一致降低性能，并使未经过滤的变体在每项任务中表现最差。除此之外，我们还发现，在某些情况下，具有更严格领域的预训练数据集的性能优于多样化的 C4 数据集。例如，使用维基百科 + TBC 语料库

12. https://github.com/jcpeterson/openwebtext
13. https://www.tensorflow.org/datasets/catalog/wikipedia

数据集	尺寸	胶水	CNNDM	SQuAD	SGLUE	EnDe	EnFr	恩罗
★ C4	745GB	83.28	**19.24**	80.88	71.36	**26.98**	**39.82**	**27.65**
C4，未经过滤	6.1TB	81.46	19.14	78.78	68.04	26.55	39.34	27.21
类似真实新闻	35GB	**83.83**	**19.23**	80.39	72.38	**26.75**	**39.90**	**27.48**
类 WebText	17GB	**84.03**	**19.31**	**81.42**	71.40	**26.80**	**39.74**	**27.59**
维基百科	16GB	81.85	**19.31**	81.29	68.01	**26.94**	39.69	**27.67**
维基百科 + 待定	20GB	83.65	**19.28**	**82.08**	**73.24**	**26.77**	39.63	**27.57**

表 8：在不同数据集上进行预训练所产生的性能。前四个变量基于我们新的 C4 数据集。

SuperGLUE 的得分为 73.24，超过了基线得分（使用 C4）71.36。这几乎完全归功于性能从 25.78（基线，C4）提高到了
50.93（维基百科 + TBC）（见表 16）。MultiRC 是一个阅读理解数据集，其最大的数据来源是小说书籍，而这正是 TBC 所覆盖的领域。同样，使用类似于 RealNews 的数据集进行预训练，ReCoRD 的精确匹配得分从 68.16 分提高到 73.72 分，ReCoRD 是一个测量新闻文章阅读理解能力的数据集。最后一个例子是，使用来自维基百科的数据在 SQuAD 上产生了显著的（但不太明显的）收益，SQuAD 是一个包含维基百科段落的问题解答数据集。之前的研究也发现了类似的情况，例如，Beltagy 等人（2019 年）发现，在研究论文的文本上对 BERT 进行预训练提高了它在科学任务上的表现。这些发现背后的主要启示是，*对域内未标记数据进行预训练可以提高下游任务的性能*。如果我们的目标是预训练一个能够快速适应任意领域语言任务的模型，那么这一点就不足为奇，但也不能令人满意。Liu 等人（2019c）还观察到，在更多样化的数据集上进行预训练可以提高下游任务的性能。这一观察结果也推动了自然语言处理领域适应性的并行研究；有关该领域的调查，请参阅如 Ruder (2019)；Li (2012)。

只在单一领域进行预训练的一个缺点是，所产生的数据集往往要小得多。同样，在我们的基线设置中，WebText-like 变种的表现与 C4 数据集相当甚至更好，而基于 Reddit 的过滤产生的数据集比 C4 小了约 40 倍，尽管它基于的 Common Crawl 数据多了 12 倍。不过，请注意，在我们的基线设置中，我们只在 $2^{35} \approx$ 34B 标记上进行了预训练，这比我们考虑的最小预训练数据集只大了约 8 倍。我们将在

下一节中探讨使用较小的预训练数据集会带来什么问题。

3.4.2 预训练 数据集大小

我们用来创建 C4 的管道旨在创建超大的预训练数据集。有了如此多的数据，我们就可以在不重复示例的情况下对模型进行预训练。目前还不清楚在预训练过程中重复示例对下游性能是有帮助还是有害，因为我们的预训练目标本身就是随机的，可以帮助防止模型多次看到完全相同的数据。

代币数	重复次数	胶水	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
完整数据集	0		*19.24*		***71.36***	**26.98**	**39.82**	**27.65**
2^{29}	64	**82.87**	**19.19**	**80.97**	**72.03**	**26.83**	**39.74**	**27.63**
2^{27}	256	82.62	**19.20**	79.78	69.97	**27.02**	**39.71**	27.33
2^{25}	*1,*024	79.55	18.57	76.27	64.76	26.38	39.56	26.80
2^{23}	*4,*096	76.34	18.33	70.92	59.29	26.37	38.84	25.81

表 9：测量预训练期间重复数据的效果。在这些实验中，我们只使用了 C4 中的前 N 个标记（第一列中显示了不同的 N 值），但仍对 2^{35} 个标记进行了预训练。这导致数据集在预训练过程中不断重复（第二列显示了每次实验的重复次数），从而可能导致记忆（见图 6）。

为了测试有限的未标记数据集的影响，我们在人为截断的 C4 版本上对基线模型进行了预训练。回想一下，我们在 $2^{35} \approx 34$B 标记（C4 总规模的一小部分）上对基线模型进行了预训练。我们考虑在由 2^{29} 、2 、2^{2725} 和 2^{23} 标记组成的 C4 截断变体上进行训练。这些大小相当于在预训练过程中分别重复数据集 64、256、1024 和 4096 次。

由此产生的下游性能如表 9 所示。不出所料，随着数据集规模的缩小，性能也会下降。我们认为这可能是由于模型开始记忆训练前的数据集。为了衡量这种情况是否属实，我们在图 6 中绘制了每种数据集大小的训练损失。事实上，随着预训练数据集的缩小，模型的训练损失明显减少，这表明可能存在记忆。Baevski 等人（2019）同样观察到，截断预训练数据集的大小会降低下游任务的性能。

我们注意到，当预训练数据集仅重复 64 次时，这些影响是有限的。这表明，重复一定量的预训练数据可能并无害处。不过，考虑到额外的预训练可能是有益的（正如我们将在第 3.6 节中展示的那样），而且获取额外的未标记数据既便宜又容易，我们建议尽可能使用大型预训练数据集。我们还注意到，对于较大的模型，这种影响可能会更明显，也就是说，对于较小的预训练数据集，较大的模型可能更容易过度拟合。

3.5 培训战略

到目前为止，我们已经考虑了这样一种情况：先在无监督任务中对模型的所有参数进行预训练，然后再在单个有监督任务中进行微调。虽然这种方法简单明了，但也有人提出了在下游/监督任务上训练模型的各种替代方法。在本节中，除了在多个任务上同时训练模型的方法外，我们还将比较不同的模型微调方案。

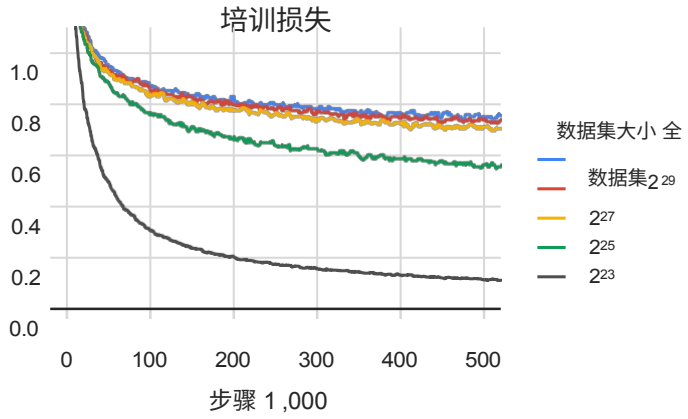

图 6：原始 C4 数据集和 4 个人为截断版本的预训练损失。列出的大小指的是每个数据集中的标记数。所考虑的四种规模对应于在预训练过程中重复数据集 64 到 4,096 次。使用较小的数据集会导致较小的训练损失值，这可能表明未标注数据集有一定的记忆性。

3.5.1 微调方法

有人认为，微调模型的所有参数会导致次优结果，尤其是在低资源任务中（Peters 等人，2019）。针对文本分类任务的迁移学习的早期成果主张只微调一个小型分类器的参数，该分类器由一个固定的预训练模型生成句子嵌入（Subra- manian 等人，2018 年；Kiros 等人，2015 年；Logeswaran 和 Lee，2018 年；Hill 等人，2016 年；Conneau 等人，2017 年）。这种方法不太适用于我们的编码器-解码器模型，因为整个解码器必须经过训练才能输出特定任务的目标序列。相反，我们将重点放在两种可供选择的微调方法上，它们只更新编码器-解码器模型的一个参数子集。

第一种是 "适配器层"（Houlsby et al.适配层是额外的密集 ReLU 密集块，添加在 Transformer 每个块中的每个前馈网络之后。这些新的前馈网络在设计上使其输出维度与输入维度相匹配。这样就可以将它们插入网络，而无需对结构或参数进行额外更改。微调时，只需更新适配器层和层归一化参数。这种方法的主要超参数是前馈网络的内维度 d，它会改变添加到模型中的新参数数量。我们尝试了不同的 d 值。

我们考虑的第二种微调方法是所谓的渐进解冻"（Howard 和 Ruder，2018 年）。在渐进解冻法中，随着时间的推移，越来越多的模型参数被微调。渐进解冻最初应用于由单层 堆栈组成的语言模型架构。在这种情况下，微调开始时只有

微调方法	胶水	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
⋆ 所有参数	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
适配器层，$d = 32$	80.52	15.08	79.32	60.40	13.84	*17.88*	15.54
适配器层，$d = 128$	81.51	16.62	79.47	63.03	19.83	27.50	22.63
适配器层，$d = 512$	81.54	17.78	79.18	64.30	23.45	33.98	25.81
适配器层，$d = 2048$	81.51	16.62	79.47	63.03	19.83	27.50	22.63
逐渐解冻	82.50	18.95	79.17	**70.79**	26.71	39.02	26.93

表 10：只更新模型参数子集的不同微调方法比较。对于适配器层，d 指的是适配器的内维度。

更新最后一层的参数，然后在训练一定次数的更新后，倒数第二层的参数也会被更新，以此类推，直到整个网络的参数被微调。为了使这种方法适用于我们的编码器-解码器模型，我们并行地逐步解冻编码器和解码器中的层，两种情况都从顶层开始。由于输入嵌入矩阵和输出分类矩阵的参数是共享的，因此我们会在整个微调过程中更新它们。回想一下，我们的基线模型由编码器和解码器各 12 层组成，微调步长为 2^{18} 。因此，我们将微调过程细分为 12 集，每集 $2^{18}/12$ 步，并在*第 n* 集中从*第 12 - n* 层训练到*第 12 层。*我们注意到，Howard 和 Ruder（2018 年）建议在每个训练历时后再微调一层。然而，由于我们的监督数据集大小差异很大，而且我们的一些下游任务实际上是许多任务的混合体（GLUE 和 SuperGLUE），因此我们采用了更简单的策略，即每 $2^{18}/12$ 步后微调一个额外层。

这些微调方法的性能比较见表 10。对于适配器层，我们使用 32、128、512、2048 的内维度 d *来*报告性能。根据以往的结果（Houlsby 等人，2019；Bapna 等人，2019），我们发现像 SQuAD 这样的低资源任务在使用较小的 d 值时效果很好，而高资源任务则需要较大的维度才能达到合理的性能。这表明，只要维度与任务规模成适当比例，适配器层可能是一种在较少参数上进行微调的有前途的技术。需要注意的是，在我们的案例中，我们将 GLUE 和 SuperGLUE 视作单个 "任务"，将它们的组成数据集合并在一起，因此尽管它们包含一些低资源数据集，但合并后的数据集足够大，因此有必要使用较大的 d 值。如果更仔细地调整解冻计划，可能会取得更好的结果。

3.5.2 多任务学习

到目前为止，我们都是先在单个无监督学习任务上对模型进行预训练，然后再在每个下游任务上分别对模型进行微调。另一种被称为 "多任务学习 "的方法（Ruder，2017；Caruana，1997）是同 时 在多个任务上训练模型。这种方法的目标通常是训练一个模型，该模型可以同时

同时执行多项任务，即模型及其大部分参数在所有任务中共享。我们在一定程度上放宽了这一目标，转而研究同时在多个任务上进行训练的方法，以便最终产生在每个任务上都表现良好的独立参数设置。例如，我们可以在多个任务上训练一个模型，但在报告性能时，允许我们为每个任务选择不同的检查点。这就放宽了多任务学习框架，使其与我们迄今为止所考虑的 "预训练--再微调 "方法相比更加平衡。我们还注意到，在我们统一的文本到文本框架中，"多任务学习 "仅仅对应于将数据集混合在一起。因此，在使用多任务学习时，我们仍然可以将无监督任务视为混合在一起的任务之一，从而在无标记数据上进行训练。相比之下，多任务学习在 NLP 中的大多数应用都会添加特定任务的分类网络，或为每个任务使用不同的损失函数（Liu 等人，2019b）。

正如 Arivazhagan 等人（2019 年）所指出的，多任务学习中一个极其重要的因素是，模型应该在每个任务中接受多少数据的训练。我们的目标是不对模型进行过少或过多的训练，也就是说，我们希望模型能从给定任务中看到足够多的数据，从而能很好地完成任务，但又不希望看到太多的数据，以至于模型记住了训练集。如何准确设定来自每个任务的数据比例取决于各种因素，包括数据集的大小、学习任务的 "难度"（即模型必须看到多少数据才能有效执行任务）、正则化等。另外一个问题是潜在的 "任务干扰 "或 "负迁移"，即在一项任务上取得好成绩可能会影响另一项任务的成绩。考虑到这些问题，我们首先探讨了设定来自每个任务的数据比例的各种策略。Wang 等人（2019a）也进行了类似的探索。

示例-按比例混合 模型对特定任务的过拟合速度主要取决于任务数据集的大小。因此，设置混合比例的一种自然方法是根据每个任务数据集的大小按比例进行采样。这等同于将所有任务的数据集合并，然后从合并后的数据集中随机抽取示例。但请注意，我们的无监督去噪任务所使用的数据集比其他任务的数据集要大得多。因此，如果我们只是按照每个数据集的大小比例进行采样，那么模型所看到的绝大部分数据都将是未标记的，它在所有有监督任务上的训练都会不足。即使没有无监督任务，有些任务（如 WMT 英语到法语）的规模也非常大，同样会挤掉大部分批次的数据。为了解决这个问题，我们在计算比例之前对数据集大小设置了一个人为的 "限制"。具体来说，如果每个

我们 N 个任务的数据集为 $e_n, n \in \{1, ., N\}$，那么我们将训练期间从第 m 个任务中抽取一个示例的概率设为 $r_m = \min(e_m, K) / \sum \min(e_n, K)$ 其中 K 是人工数据集的大小限制。

温标混合法 减轻数据集大小之间巨大差异的另一种方法是调整混合率的 "温度"。

多语种 BERT 就采用了这种方法，以确保模型在低资源语言上得到充分训练。[14] 为了用温度 T 实现温度缩放，我们提高了

14. https://github.com/google-research/bert/blob/master/multilingual.md

每项任务的混合率 r_m 到 $1/T$ 的幂，并对混合率进行重归一化，使其总和为 1。当 $T = 1$ 时，这种方法等同于实例比例混合，随着 T 的增加，比例变得更接近于等比例混合。我们保留了数据集大小限制 K（用于在温度缩放之前获得 r_m），但将其设置为一个大值 $K = 2^{21}$。我们使用较大的 K 值是因为温度升高会降低最大数据集的混合率。

平等混合 在这种情况下，我们以相同的概率从每个任务中抽取示例。具体来说，每批中的每个示例都是从我们训练的数据集之一中均匀随机抽样的。这很可能是一种次优策略，因为模型在低资源任务中会很快过拟合，而在高资源任务中则会过拟合。我们主要将其作为一个参考点，说明当比例设置为次优时可能会出现什么问题。

为了将这些混合策略与我们的 "预训练-微调 "基线结果进行平等比较，我们对多任务模型进行了总步数相同的训练：$2^{19} + 2^{18} = 786\,432$。结果如表 11 所示。

总的来说，我们发现在大多数任务中，多任务训练的效果都不如预训练后再进行微调的效果好。特别是 "等量 "混合策略会导致性能急剧下降，这可能是因为低资源任务过度拟合，高资源任务没有看到足够的数据，或者模型没有看到足够的未标记数据来学习通用语言能力。对于例子-比例混合，我们发现对于大多数任务，K 值都有一个 "甜蜜点"，在这个点上，模型可以获得最佳性能，而 K 值越大或越小，性能往往越差。但 WMT 英法互译任务是个例外（在我们考虑的 K 值范围内），它是一项高资源任务，因此混合比例越大越好。最后，我们注意到，温标混合也是在大多数任务中获得合理性能的一种方法，在大多数情况下，$T = 2$ 的性能最佳。Arivazhagan 等人（2019 年）和 McCann 等人（2018 年）等人之前已经观察到，多任务模型的性能优于在每个单独任务上训练的单独模型，不过也有研究表明，多任务设置可以在非常相似的任务中带来好处 Liu 等人（2019b）；Ratner 等人（2018 年）。在下面的章节中，我们将探讨如何缩小多任务训练与预训练然后微调方法之间的差距。

3.5.3 多任务学习与微调相结合

回想一下，我们正在研究多任务学习的宽松版本，即在混合任务上训练单一模型，但允许使用模型的不同参数设置（检查点）来评估性能。我们可以扩展这种方法，考虑在所有任务上同时对模型进行预训练，然后在单个监督任务上对模型进行微调的情况。这就是 "MT-DNN"（Liu 等人，2015，2019b）所使用的方法，该方法一经推出就在 GLUE 和其他基准上取得了最先进的性能。我们考虑了这种方法的三种变体：在第一种变体中，我们只是在一个实例比例混合模型上对模型进行预训练，人工数据集大小限制为 $K = 2^{19}$ ，然后再在每个单独的下游任务上对模型进行微调。这有助于我们衡量在预训练过程中，是否将有监督任务与无监督目标同时考虑在内。

混合策略	胶水	CNNDM	SQuAD	SGLUE	EnDe	EnFr	恩罗
⋆ 基准线（预培训/微调）	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
平等	76.13	19.02	76.51	63.37	23.89	34.31	26.78
示例-比例，$K = 2^{16}$	80.45	19.04	77.25	69.95	24.35	34.99	27.10
示例-比例，$K = 2^{17}$	81.56	19.12	77.00	67.91	24.36	35.00	27.25
示例-比例，$K = 2^{18}$	81.67	19.07	78.17	67.94	24.57	35.19	27.39
示例-比例，$K = 2^{19}$	81.42	**19.24**	79.78	67.30	25.21	36.30	**27.76**
示例-比例，$K = 2^{20}$	80.80	**19.24**	**80.36**	67.38	25.66	36.93	**27.68**
示例-比例，$K = 2^{21}$	79.83	18.79	79.50	65.10	25.82	37.22	27.13
温标，$T = 2$	81.90	**19.28**	79.42	69.92	25.42	36.72	27.20
温标，$T = 4$	80.56	**19.22**	77.99	69.54	25.04	35.82	27.45
温标，$T = 8$	77.21	19.10	77.14	66.07	24.55	35.35	27.17

表 11：使用不同混合策略进行多任务训练的比较。示例比例混合是指根据每个数据集的总大小从每个数据集中抽取示例，并人为限制数据集的最大大小 (K)。温标混合法是通过温度 T 对采样率进行重新缩放。对于温标混合，我们使用的人为数据集大小限制为 $K = 2^{21}$ 。

让模型在早期接触下游任务时获得一些有益的经验。我们可能还希望，混合多种监督来源可以帮助预训练模型在适应单个任务之前获得一套更广泛的 "技能"（宽泛地说）。为了直接衡量这一点，我们考虑了第二种变体，即在相同的示例-比例混合物（$K = 2^{19}$ ）上对模型进行预训练，只是在预训练混合物中省略了一个下游任务。然后，我们在预训练时忽略的任务上对模型进行微调。对于我们考虑的每个下游任务，我们都会重复上述步骤。我们称这种方法为 "遗漏式 "多任务训练。这模拟了现实世界中的情况，即在预训练期间，预训练过的模型会在未见过的任务上进行微调。需要注意的是，多任务预训练提供了多种有监督任务。由于其他领域（如计算机视觉（Oquab 等人，2014 年；Jia 等人，2014 年；Huh 等人，2016 年；Yosinski 等人，2014 年））使用有监督数据集进行预训练，因此我们有兴趣了解从多任务预训练混合物中省略无监督任务是否仍能产生良好效果。因此，对于我们的第三种变体，我们在 $K = 2^{19}$ 的所有监督任务的实例比例混合物上进行预训练。在所有这些变体中，我们都遵循我们的标准程序，即先进行 2 步^{19} 的预训练，然后再进行 2 步^{18} 的微调。

表 12 比较了这些方法的结果。为了便于比较，我们还列出了基线（先预训练再微调

）和标准多任务学习（不进行微调）在 $K = 2^{19}$ 的示例比例混合物上的结果。我们发现，在多任务预训练后进行微调的结果与我们的基准线性能相当。这表明，在多任务学习后进行微调有助于减轻第 3.5.2 节所述的不同混合率之间的一些权衡。有趣的是，"leave-one-out "训练的性能只略微差一些，这表明在多种任务中训练过的模型仍然可以适应新任务（即多任务预训练可能不会导致剧烈的任务干扰）。最后，除翻译任务外，有监督的多任务预训练在所有情况下都表现较差。这

培训战略	胶水	CNNDM	SQuAD	SGLUE	EnDe	EnFr	恩罗
⋆ 无监督预训练 + 微调	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	39.82	27.65
多任务培训	81.42	**19.24**	79.78	67.30	25.21	36.30	27.76
多任务预培训 + 微调	**83.11**	**19.12**	**80.26**	**71.03**	**27.08**	39.80	**28.07**
留白式多任务训练	81.98	19.05	79.97	**71.68**	**26.93**	39.79	**27.87**
有监督的多任务预培训	79.93	18.96	77.38	65.36	26.81	**40.13**	**28.04**

表 12：无监督预培训、多任务学习和各种形式的 多任务预培训的比较。

这可能表明，翻译任务从（英语）预培训中获益较少，而无监督预培训则是其他任务的一个重要因素。

3.6 缩放

机器学习研究的 "惨痛教训 "认为，能够利用额外计算的通用方法最终会战胜依赖人类专业知识的方法（Sutton，2019；Hestness 等人，2017；Shazeer 等人，2017；Jozefowicz等人，2016；Mahajan等人，2018；Shazeer等人，2018，2017；Huang等人，2018b；Keskar等人，2019a）。最近的研究结果表明，这可能适用于 NLP 中的迁移学习（Liu 等人，2019c；Radford 等人，2019；Yang 等人，2019；Lan 等人，2019），也就是说，与更精心设计的方法相比，扩大规模会产生更好的性能。然而，有多种可能的扩展方法，包括使用更大的模型、对模型进行更多步骤的训练以及集合。在本节中，我们将针对以下前提对这些不同方法进行比较："你刚刚获得了 4 倍的计算能力。你应该如何使用它？

我们从基线模型开始，该模型有 2.2 亿个参数，分别按 2^{19} 步和 2^{18} 步进行预训练和微调。编码器和解码器的大小与 "$\text{BERT}_{\text{BASE}}$ "相似。为了尝试增加模型规模，我们遵循 "$\text{BERT}_{\text{LARGE}}$ " Devlin 等人（2018 年）的指导原则，使用 $d_{\text{ff}} = 4096$、$d_{\text{model}} = 1024$、$d_{\text{kv}} = 64$ 和 16 头注意力机制。然后，我们在编码器和解码器中分别生成了 16 层和 32 层的两个变体，产生的模型参数数量分别是原始模型的 2 倍和 4 倍。这两个变体的计算成本也分别为原来的 2 倍和 4 倍。利用我们的基线和这两个更大的模型，我们考虑了三种使用 4 倍计算量的方法：以 4 倍的步数进行训练，以 2 倍的步数

用 2 倍大的模型进行训练，以及以 "基线 "步数训练 4 倍大的模型。在增加训练步数时，为了简单起见，我们同时增加了预训练和微调步数。请注意，在增加预训练步数时，我们实际上包含了更多的预训练数据，因为 C4 非常大，即使训练 2^{23} 步，我们也无法完成对数据的一次传递。

让模型查看 4 倍数据量的另一种方法是将批处理量增加 4 倍，由于并行化效率更高，这有可能加快训练速度。不过，使用 4 倍大的批处理量进行训练，与使用 4 倍大的批处理量进行训练，可能会产生不同的结果。

扩展战略	胶水	CNNDM	SQuAD	SGLUE	EnDe	EnFr	恩罗
⋆ 基准线	83.28	19.24	80.88	71.36	26.98	39.82	27.65
1× 大小，*4*× 训练步骤	85.33	19.33	82.45	74.72	27.08	40.66	27.93
1 倍大小，*4* 倍批量	84.60	19.42	82.52	74.64	27.07	40.60	27.84
2 倍大小，2 *倍*训练步骤	**86.18**	19.66	**84.18**	77.18	27.52	**41.03**	28.19
4 倍大小，*1* 倍训练步骤	**85.91**	19.73	**83.86**	**78.04**	27.47	40.71	28.10
4× 组合	84.77	**20.10**	83.09	71.74	**28.05**	40.53	**28.57**
4× 组合，仅微调	84.05	19.57	82.36	71.55	27.55	40.22	28.09

表 13：扩大基线模型的不同方法比较。除集合微调模型外，所有方法的计算量都是基线模型的 4 倍。"大小 "指的是模型中参数的数量，"训练时间 "指的是用于预训练和微调的步骤数量。

4 倍的步骤（Shallue 等人，2018 年）。为了比较这两种情况，我们还进行了一项额外的实验，用大 4 倍的批次规模来训练基线模型。

在我们考虑的许多基准中，通常的做法是通过使用模型集合进行训练和评估，以获得额外的性能。这提供了一种使用额外计算的正交方法。为了将其他扩展方法与集合方法进行比较，我们还测量了由 4 个单独预训练和微调的模型组成的集合的性能。在将对数输入输出软最大非线性之前，我们先平均整个集合的对数，以获得综合预测结果。与预训练 4 个单独的模型相比，一个更便宜的替代方法是使用一个预训练模型，并生成 4 个单独的微调版本。虽然这并没有使用我们全部的 4 倍计算预算，但我们也加入了这种方法，看看它是否能产生与其他缩放方法相媲美的性能。

表 13 显示了采用这些不同缩放方法后所取得的性能。不难看出，增加训练时间和/或模型大小都能持续改善基线。在训练步骤增加 4 倍或使用 4 倍大的批次规模之间没有明显的优胜者，尽管两者都有好处。一般来说，与单纯增加训练时间或批次规模相比，增加模型规模会带来额外的性能提升。在我们研究的任何任务中，我们都没有观察到训练 2 倍大的模型和训练 4 倍大的模型在训练时间上有很大差别。这表明，增加训练时间和增大模型规模是提高性能的互补手段。我们的研究结果还表明，集合是通过规模提高性能的一种正交而有效的方法。在某些任务（CNN/DM、WMT 英语到德语以及 WMT 英语到罗马尼亚语）中，4 个完全单独训练的模型的集合性能明显优于其他任何扩展方法。将预先训练好的模型集合在一起，但分别进

行微调，其性能也比基线有大幅提高，这表明这是一种提高性能的更经济的方法。唯一的例外是 SuperGLUE，它的集合方法都没有比基线方法有显著提高。

我们注意到，不同的缩放方法有不同的权衡，这与它们的性能是分开的。例如，使用更大的模型会使下游微调和

推理成本更高。与此相反，如果将一个小型模型应用于许多下游任务，那么预训练较长时间的成本就会被有效摊销。另外，我们注意到，集合 N 个独立模型的成本类似于使用一个计算成本高 N 倍的模型。因此，在选择缩放方法时，必须对模型的最终用途有所考虑。

3.7 将所有内容整合在一起

现在，我们将利用系统研究中获得的洞察力，确定我们能在流行的 NLP 基准上将性能提升到什么程度。我们还有兴趣通过在大量数据上训练更大的模型来探索 NLP 目前迁移学习的极限。我们从基线训练方法开始，并做出以下改变：

目标 我们将基线中的 i.i.d. 去噪目标换成了第 3.3.4 节中描述的跨度破坏目标，其灵感来源于 SpanBERT（Joshi 等人，2019 年）。具体来说，我们使用的平均跨度长度为 3，破坏原始序列的 15%。我们发现，这一目标的性能略有提高（表 7），同时由于目标序列长度更短，计算效率略高。

更长时间的训练 我们的基线模型使用了相对较少的预训练量（$1/4$，相当于 BERT（Devlin 等，2018 年）的预训练量；$1/16$，相当于 XLNet（Yang 等，2019 年）的预训练量；$1/64$，相当于 RoBERTa（Liu 等，2019c）的预训练量等）。幸运的是，C4 足够大，我们可以在不重复数据的情况下训练更长的时间（如第 3.4.2 节所示，重复数据可能是有害的）。我们在第 3.6 节中发现，额外的预训练确实有帮助，而增加批量大小和增加训练步数都能带来这种好处。因此，我们在长度为 512 的 2^{11} 序列上对模型进行了 100 万步的预训练，相当于总计约 1 万亿个预训练标记（约为基线的 32 倍）。在第 3.4.1 节中，我们展示了在类 RealNews、类 WebText 和维基百科 + TBC 数据集上进行的预训练在一些下游任务上优于在 C4 上进行的预训练。然而，这些数据集的变体非常小，以至于在对 1 万亿个词组进行预训练的过程中会重复数百次。由于我们在第 3.4.2 节中指出这种重复可能是有害的，因此我们选择继续使用 C4 数据集。

模型大小 在第 3.6 节中，我们还展示了扩大基线模型大小如何提高性能。然而，在计

算资源有限，无法进行微调或推理的情况下，使用较小的模型可能会有所帮助。基于这些因素，我们训练了各种规模的模型：

- **基线模型**这是我们的基准模型，其超参数在第 3.1.1 节中有所描述。它大约有 2.2 亿个参数。
- **小。**我们考虑了一个更小的模型，它通过使用 $d_{\text{model}} = 512$、$d_{\text{ff}} = 2{,}048$、8 头注意力以及编码器和解码器各只有 6 层来缩小基线。这个变体有大约 6000 万个参数。

- **大。**由于基线使用的是 $\text{BERT}_{\text{BASE}}$ 大小的编码器和解码器，我们还考虑了编码器和解码器在大小和结构上都与 $\text{BERT}_{\text{LARGE}}$ 相似的变体。具体而言，该变体使用 $d_{\text{model}} = 1{,}024$、$d_{\text{ff}} = 4{,}096$、$d_{\text{kv}} = 64$、16 头注意力以及编码器和解码器各 24 层，从而产生约 7.7 亿个参数。

- **3B 和 11B。**为了进一步探索使用更大模型时可能产生的性能，我们考虑了另外两种变体。在这两种情况下，我们使用 $d_{\text{model}} = 1024$、24 层编码器和解码器，以及 $d_{\text{kv}} = 128$。对于 "3B "变体，我们使用 $d_{\text{ff}} = 16{,}384$ 和 32 头注意力，这将导致约
  28 亿个参数；对于 "11B"，我们使用 $d_{\text{ff}} = 65{,}536$ 和 128 头注意力，从而产生一个拥有约 110 亿个参数的模型。我们选择扩大 d_{ff}，特别是因为现代加速器（如我们训练模型所使用的 TPU）对于大型密集矩阵乘法（如 Transformer 的前馈网络中的矩阵乘法）最为有效。

多任务预训练 在第 3.5.3 节中，我们展示了在微调之前对无监督任务和有监督任务的多任务混合预训练与单独对无监督任务的预训练效果一样好。这就是 "MT-DNN"（Liu 等人，2015 年，2019b）所倡导的方法。这种方法还有一个实际好处，那就是可以在整个训练过程中监控 "下游 "性能，而不仅仅是在微调期间。因此，我们在最后一组实验中使用了多任务预训练。我们假设，训练时间更长的大型模型可能会从更大比例的未标注数据中获益，因为它们更有可能对较小的训练数据集进行过拟合。不过，我们也注意到，第 3.5.3 节的结果表明，多任务预训练后的微调可以减轻因选择次优比例的无标记数据而可能产生的一些问题。基于这些想法，我们在使用标准示例比例混合法（见第 3.5.2 节）之前，用以下人工数据集大小代替了未标记数据：710,000（小）、2,62*0*,000（基础）、8,66*0,000*（大）、33,500,000（3B）和 133,*000*,000（11B）。对于所有模型变体，我们还在预训练期间将 WMT English to French 和 WMT English to German 数据集的有效数据集大小限制为 100 万例。

对单个 GLUE 和 SuperGLUE 任务进行微调 到目前为止，在对 GLUE 和 SuperGLUE

进行微调时，我们将每个基准测试中的所有数据集合并在一起，这样我们只需对 GLUE 和 SuperGLUE 模型分别微调一次。这种方法使我们的研究在逻辑上更加简单，但我们发现，与分别对任务进行微调相比，这种方法在某些任务上会牺牲少量性能。在单个任务上进行微调的一个潜在问题是，我们可能会对资源较少的任务迅速过度适应，而在所有任务上同时进行训练则可以缓解这一问题。例如，我们使用 2^{11} length-512 序列的大容量批次，会导致整个数据集在许多低资源 GLUE 和 SuperGLUE 任务的每个批次中出现多次。因此，我们在对每个 GLUE 和 SuperGLUE 任务进行微调时，使用了长度为 8 个 512 序列的较小批次。我们还每隔 1000 步保存检查点，而不是每隔 5000 步保存检查点，以确保我们能在模型过拟合之前获得其参数。

波束搜索 我们之前报告的所有结果都是使用贪婪解码。对于输出序列较长的任务，我们发现使用波束搜索可以提高性能（Sutskever 等人，2014 年）。具体来说，我们在 WMT 翻译和 CNN/DM 摘要任务中使用了 4 的波束宽度和 $\alpha = 0.6$ 的长度惩罚（Wu 等人，2016 年）。

测试集 由于这是最后一组实验，我们报告的是测试集而非验证集的结果。对于 CNN/《每日邮报》，我们使用随数据集分发的标准测试集。对于 WMT 任务，这相当于使用 `newstest2014` 来测试英语-德语，使用 `newstest2015` 来测试英语-法语，使用 `newstest2016` 来测试英语-罗马尼亚语。对于 GLUE 和 SuperGLUE，我们使用基准评估服务器计算官方测试集分数。[15,16] 对于 SQuAD，在测试集上进行评估需要在基准服务器上运行推理。遗憾的是，该服务器上的计算资源不足以从我们最大的模型中获得预测结果。因此，我们继续在 SQuAD 验证集上报告性能。幸运的是，在 SQuAD 测试集上性能最高的模型也在验证集上报告了结果，因此我们仍然可以与表面上最先进的模型进行比较。

除上述变化外，我们还使用了与基线相同的训练程序和超参数（AdaFactor 优化器、用于预训练的反平方根学习率计划、用于微调的恒定学习率、Dropout 正则化、词汇量等）。这些细节将在第 2 节中介绍，以供参考。

最后一组实验结果如表 14 所示。总体而言，在我们考虑的 24 项任务中，我们在 18 项任务上取得了最先进的性能。不出所料，在所有任务中，我们的最大（110 亿参数）模型在模型规模变体中表现最佳。我们的 T5-3B 模型变体确实在少数任务中超越了之前的技术水平，但将模型规模扩展到 110 亿参数是实现最佳性能的最重要因素。现在我们来分析各个基准的结果。

我们的 GLUE 平均得分为 90.3 分，达到了最先进水平。值得注意的是，在自然语言推理任务 MNLI、RTE 和 WNLI 方面，我们的表现大大优于之前的先进水平。RTE 和 WNLI 是机器性能历来落后于人类性能的两个任务，人类性能分别为 93.6 和 95.9（Wang 等人，2018 年）。就参数数量而言，我们的 11B 模型变体是已提交给 GLUE 基准的最大模型。然而，大多数得分最高的提交模型都使用了大量的集合和计算来产生预测结果。例如，表现最好的 ALBERT 变体（Lan 等人，2019 年）使用的模型在规模和架构上与我们的 3B 变体相似（尽管由于巧妙的参数共享，它的参数数量大大减少）。为了在 GLUE 上取得令人印象深刻的性能，ALBERT 的作者根据任务的不同，将 "6 到 17 个 "模型组合在一起。这可能会导致使用 ALBERT 集合进行预测的计算成本比使用 T5-11B 时更高。

就 SQuAD 而言，我们在精确匹配得分上比之前的先进水平（ALBERT（Lan 等人，2019 年））高出 1 分多。SQuAD 是一个历史悠久的基准

15. http://gluebenchmark.com
16. http://super.gluebenchmark.com

模型	胶水平均	CoLA 马太福音	SST-2 准确性	MRPC F1	MRPC 准确性	STS-B 皮尔森	STS-B 斯皮尔曼
上一个最佳	89.4[a]	69.2[b]	97.1[a]	**93.6**[b]	**91.5**[b]	92.7[b]	92.3[b]
T5-小型	77.4	41.0	91.8	89.7	86.6	85.6	85.0
T5 基座	82.7	51.1	95.2	90.7	87.5	89.4	88.6
T5 大号	86.4	61.2	96.3	92.4	89.9	89.9	89.2
T5-3B	88.5	67.1	97.4	92.5	90.0	90.6	89.8
T5-11B	**90.3**	**71.6**	**97.5**	92.8	90.4	**93.1**	**92.8**

模型	QQP F1	QQP 准确性	MNLI-m 准确性	MNLI-mm 准确性	QNLI 准确性	RTE 准确性	WNLI 准确性
上一个最佳	74.8[c]	**90.7**[b]	91.3[a]	91.0[a]	**99.2**[a]	89.2[a]	91.8[a]
T5-小型	70.0	88.0	82.4	82.3	90.3	69.9	69.2
T5 基座	72.6	89.4	87.1	86.2	93.7	80.1	78.8
T5 大号	73.9	89.9	89.9	89.6	94.8	87.2	85.6
T5-3B	74.4	89.7	91.4	91.2	96.3	91.1	89.7
T5-11B	**75.1**	90.6	**92.2**	**91.9**	96.9	**92.8**	**94.5**

模型	SQuAD EM	SQuAD F1	SuperGLUE 平均	BoolQ 准确性	CB F1	CB 准确性	COPA 准确性
上一个最佳	90.1[a]	95.5[a]	84.6[d]	87.1[d]	90.5[d]	95.2[d]	90.6[d]
T5-小型	79.10	87.24	63.3	76.4	56.9	81.6	46.0
T5 基座	85.44	92.08	76.2	81.4	86.2	94.0	71.2
T5 大号	86.66	93.79	82.3	85.4	91.6	94.8	83.4
T5-3B	88.53	94.95	86.4	89.9	90.3	94.4	92.0
T5-11B	**91.26**	**96.22**	**88.9**	**91.2**	**93.9**	**96.8**	**94.8**

模型	MultiRC F1a	MultiRC EM	ReCoRD F1	ReCoRD 准确性	RTE 准确性	WiC 准确性	WSC 准确性
上一个最佳	84.4[d]	52.5[d]	90.6[d]	90.0[d]	88.2[d]	69.9[d]	89.0[d]
T5-小型	69.3	26.3	56.3	55.4	73.3	66.9	70.5
T5 基座	79.7	43.1	75.0	74.2	81.5	68.3	80.8
T5 大号	83.3	50.7	86.8	85.9	87.8	69.3	86.3
T5-3B	86.8	58.3	91.2	90.4	90.7	72.1	90.4
T5-11B	**88.1**	**63.3**	**94.1**	**93.4**	**92.5**	**76.9**	**93.8**

模型	WMT EnDe BLEU	WMT EnFr BLEU	WMT EnRo BLEU	CNN/DM ROUGE-1	CNN/DM 红宝石-2	CNN/DM ROUGE-L
上一个最佳	**33.8**[e]	**43.8**[e]	**38.5**[f]	43.47[g]	20.30[g]	40.63[g]
T5-小型	26.7	36.0	26.8	41.12	19.56	38.35
T5 基座	30.9	41.2	28.0	42.05	20.34	39.40
T5 大号	32.0	41.5	28.1	42.50	20.68	39.75
T5-3B	31.8	42.6	28.2	42.72	21.02	39.94
T5-11B	32.1	43.4	28.1	**43.52**	**21.55**	**40.69**

表 14：我们的 T5 变体在我们研究的每个任务上的性能。Small、Base、Large、3B 和 11B 分别指参数为 6000 万、2.2 亿、7.7 亿、30 亿和 110 亿的模型配置。在每个表格的第一行，我们报告了该任务的最新成果（截至 2019 年 10 月 24 日），上标表示其来源，参考文献列于本标题末尾。除 SQuAD 使用验证集

外，所有结果都是在测试集上报告的。([a]Lan 等人，2019） [b]（Wang 等人，2019c） （[c]Zhu 等人，2019） （[d]Liu 等人，2019c） （[e]Edunov 等人，2018） （[f]Lample 和 Conneau，2019） （[g]Dong 等人，2019）

我们注意到，当报告测试集的结果时，它们通常是基于集合模型和/或利用外部数据集（如 TriviaQA（Joshi 等，2017 年）或 NewsQA（NewsQA，2017 年））。我们注意到，当报告测试集的结果时，它们通常是基于一组模型和/或利用外部数据集（如 TriviaQA（Joshi 等人，2017 年）或 NewsQA（Trischler 等人，2016 年））来增强 SQuAD 的小型训练集。据估计，人类在 SQuAD 上的表现分别为 82.30 分和 82.30 分。
精确匹配和 F1 指标分别为 91.22（Rajpurkar 等人，2016 年），因此尚不清楚对这一基准的进一步改进是否有意义。

就 SuperGLUE 而言，我们在最先进系统的基础上有了很大提高（从平均 84.6 分（Liu 等人，2019c）提高到 88.9 分）。SuperGLUE 在设计上包含了 "超出当前最先进系统范围，但大多数受过大学教育的英语使用者都能解决"（Wang 等人，2019b）的任务。我们的成绩几乎与人类的 89.8 分不相上下（Wang 等人，2019b）。有趣的是，在阅读理解任务（MultiRC 和 ReCoRD）上，我们的表现远远超过人类，这表明这些任务所使用的评估指标可能偏向于机器预测。另一方面，人类在 COPA 和 WSC 上的准确率都达到了 100%，明显优于我们模型的表现。这表明，我们的模型在语言任务方面仍然难以做到尽善尽美，尤其是在低资源环境下。

在任何一项 WMT 翻译任务中，我们都没有达到最先进的性能。部分原因可能是我们使用了纯英语的无标注数据集。我们还注意到，这些任务中的大多数最佳结果都使用了反向翻译（Edunov 等人，2018 年；Lample 和 Conneau，2019 年），这是一种复杂的数据增强方案。低资源英语到罗马尼亚语基准的最新技术也使用了额外形式的跨语言无监督训练（Lample 和 Conneau，2019 年）。我们的研究结果表明，规模和英语语言预训练可能不足以与这些更复杂的方法相媲美。更具体地说，英语到德语 `newstest2014` 集的最佳结果使用了 WMT 2018（Edunov et al.

最后，在 CNN/Daily Mail 上，我们达到了最先进的性能，尽管在 ROUGE-2-F 分数上仅有显著提高。事实证明，ROUGE 分数的提高并不一定会带来更连贯的摘要（Paulus 等人，2017 年）。此外，虽然 CNN/《每日邮报》被设定为抽象总结基准，但纯粹的提取方法也被证明效果良好（Liu，2019）。也有人认为，用最大似然法训练的生成模型容易产生重复性摘要（See 等人，2017 年）。尽管存在这些

潜在问题，但我们发现我们的模型确实能生成连贯且基本正确的摘要。我们在附录 C 中提供了一些非樱桃采摘验证集示例。

T5 将我们的实验研究成果与前所未有的规模相结合，取得了显著的成果。请注意，在第 3.6 节中，我们发现扩大基线模型的预训练量或规模会产生巨大的收益。有鉴于此，我们有兴趣衡量一下我们在 T5 中引入的 "非扩展 "变化对其强劲性能的贡献程度。因此，我们进行了最后一次实验，比较了以下三种配置：第一种是标准基线模型，该模型在 $2^{35} \approx 34$B 标记上进行了预训练；第二种是在约 1 万亿标记（即与 T5 的预训练量相同）上训练的基线，我们称之为 "基线-1T"；第三种是 T5-基线。请注意，基线-1T 和 T5-Base 之间的差异包括

模型	胶水	CNNDM	SQuAD	SGLUE	EnDe	EnFr	恩罗
⋆ 基准线	83.28	19.24	80.88	71.36	26.98	39.82	27.65
基线-1T	84.80	19.62	83.01	73.90	27.46	40.30	28.34
T5 基座	**85.97**	**20.90**	**85.44**	**75.64**	**28.37**	**41.37**	**28.98**

表 15：T5-Base 与本文其余部分使用的基线实验装置的性能比较。结果在验证集上报告。"基线-1T "指的是在 1 万亿个代币（与 T5 模型变体使用的数量相同）而不是 $2^{35} \approx 34$B 代币（与基线使用的数量相同）上对基线模型进行预训练所取得的性能。

我们在设计 T5 时所作的 "非扩展 "改动。因此，通过比较这两个模型的性能，我们可以具体衡量系统性研究的见解所产生的影响。

这三种模型配置的性能如表 15 所示。与第 3.6 节中的研究结果一致，我们发现额外的预训练比基线性能有所提高。尽管如此，T5-Base 在所有下游任务中的表现都大大优于基线-1T。这表明，规模并不是 T5 取得成功的唯一因素。我们假设，大型模型不仅得益于其规模的扩大，还得益于这些非规模因素。

4. 反思

在完成系统研究之后，我们首先总结了一些最重要的发现。我们的研究结果提供了一些高层次的视角，让我们了解哪些研究途径可能更有前途，哪些研究途径不太有前途。最后，我们概述了一些我们认为可以为进一步推动该领域发展提供有效方法的课题。

4.1 收获

文本到文本 我们的文本到文本框架提供了一种简单的方法，可以使用相同的损失函数和解码程序在各种文本任务中训练单一模型。我们展示了这种方法如何成功应用于生成任务（如抽象概括）、分类任务（如自然语言推理），甚至回归任务（如 STS-B）。尽管简单，但我们发现文本到文本框架的性能可与特定任务架构相媲美，并在与规模相结合时最终产生了最先进的结果。

架构 虽然一些针对 NLP 的迁移学习工作考虑了 Transformer 的架构变体，但我们发现原始的编码器-解码器形式在我们的文本-文本框架中效果最佳。虽然编码器-解码器模型使用的参数是 "纯编码器"（如 BERT）或 "纯解码器"（语言模型）架构的两倍，但其计算成本却相差无几。我们还证明，共享编码器和解码器中的参数不会导致性能大幅下降，同时将总参数数减半。

无监督目标 总体而言，我们发现大多数 "去噪 "目标（训练模型重建随机损坏的文本）在文本到文本的设置中表现类似。因此，我们建议使用能产生较短目标序列的目标，这样无监督预训练的计算效率更高。

数据集 我们引入了 "Colossal Clean Crawled Corpus"（C4），它包括从 Common Crawl 网络转储中启发式清理的文本。在将 C4 与使用额外过滤的数据集进行比较时，我们发现，使用域内无标记数据进行训练可以提高一些下游任务的性能。不过，局限于单一领域通常会导致数据集较小。我们还发现，当未标注数据集足够小，以至于在预训练过程中重复多次时，性能就会下降。这就促使我们在通用语言理解任务中使用像 C4 这样的大型多样化数据集。

训练策略 我们发现，在微调过程中更新所有预训练模型参数的基本方法优于更新较少参数的方法，尽管更新所有参数的成本最高。我们还尝试了在多个任务上同时训练模型的各种方法，在我们的文本到文本设置中，这种方法简单地说就是在构建批次时混合来自不同数据集的示例。多任务学习的首要问题是设定每个任务的训练比例。我们最终没有找到一种设定混合比例的策略，能与无监督预训练后再进行监督微调的基本方法的性能相匹配。不过，我们发现，在混合任务上进行预训练后再进行微调，效果与无监督预训练相当。

扩展 我们比较了利用额外计算能力的各种策略，包括在更多数据上训练模型、训练更大的模型以及使用模型集合。我们发现，每种方法都能显著提高性能，但在更多数据上训练较小模型的效果往往优于在较少步骤中训练较大模型的效果。我们还发现，集合模型能提供比单一模型好得多的结果，这提供了一种利用额外计算的正交方法。与完全单独预训练和微调所有模型相比，由同一基础预训练模型微调的集合模型表现更差，不过仅微调集合模型的表现仍大大优于单一模型。

挑战极限 我们结合上述见解，训练了更大的模型（多达 110 亿个参数），在我们考虑的许多基准测试中取得了最先进的结果。在无监督训练中，我们从 C4 数据集中提取文本，并应用去噪目标来破坏连续的词组。在对单个任务进行微调之前，我

们对多任务混合进行了预训练。总体而言，我们的模型在超过 1 万亿个词组上进行了训练。为了便于复制、扩展和应用我们的成果，我们发布了我们的代码、C4 数据集以及每个 T5 变体的预训练模型权重。[1]

4.2 展望

大型模型带来的不便 我们的研究得出了一个不足为奇但很重要的结果，即大型模型往往性能更好。用于运行这些模型的硬件不断变得更便宜、更强大，这一事实表明，扩大规模可能仍然是实现更高性能的一种有前途的方法（Sutton，2019 年）。不过，在某些应用和场景中，使用较小或较便宜的模型总是有帮助的，例如在执行客户端推理或联合学习时（Konečny`等人，2015年，2016年）。与此相关，迁移学习的一个有益用途是在低资源任务中获得良好性能的可能性。低资源任务（顾名思义）通常发生在缺乏资产来标注更多数据的环境中。因此，低资源应用通常也只能获得有限的计算资源，从而产生额外的成本。因此，我们主张研究能用更便宜的模型实现更高性能的方法，这样迁移学习就能应用到影响最大的地方。目前在这方面开展的一些工作包括蒸馏（Hinton 等人，2015 年；Sanh 等人，2019 年；Jiao 等人，2019 年）、参数共享（Lan 等人，2019 年）和条件计算（Shazeer 等人，2017 年）。

更高效的知识提取 回顾一下，预训练的目标之一是（粗略地说）为模型提供通用 "知识"，以提高其在下游任务中的性能。我们在这项工作中使用的方法，也是目前常见的做法，是训练模型对损坏的文本跨度进行去噪。我们怀疑这种简单化的技术可能不是教授模型通用知识的有效方法。更具体地说，如果我们不需要先在 1 万亿个文本标记上训练模型，就能获得良好的微调性能，那将是非常有用的。一些同时进行的工作正是沿着这一思路，通过预训练模型来区分真实文本和机器生成的文本，从而提高了效率（Clark 等人，2020 年）。

正式确定任务之间的相似性 我们发现，对未标记的域内数据进行预训练可以提高下游任务的性能（第 3.4 节）。这一发现主要依赖于基本的观察结果，例如 SQuAD 是使用维基百科的数据创建的。如果能对前期训练和下游任务之间的 "相似性 " 提出一个更严格的概念，我们就能对使用何种非标记数据源做出更有原则性的选择。在计算机视觉领域，已经有一些早期的实证工作沿着这一思路展开（Huh 等人，2016 年；Kornblith 等人，2018 年；He 等人，2018 年）。更好地了解任务的相关性也有助于选择*有监督的*预训练任务，这已被证明对 GLUE 基准有帮助（

Phang 等人，2018 年）。

语言无关模型 我们失望地发现，在我们研究的翻译任务中，纯英语的预训练并没有取得最先进的结果。我们还希望避免需要提前指定词汇表可以编码哪些语言的后勤困难。为了解决这些问题，我们有兴趣进一步研究语言无关模型，即无论文本使用哪种语言，都能出色完成特定 NLP 任务的模型。这尤其是

鉴于英语不是世界上大多数人的母语，这是一个相关的问题。

我们撰写这篇论文的动力来自于最近在 NLP 迁移学习方面的大量工作。在我们开始这项工作之前，这些进展已经在一些基于学习的方法尚未被证明有效的环境中实现了突破。我们很高兴能够延续这一趋势，例如，我们在 SuperGLUE 基准测试中的表现几乎达到了人类水平，而这是一项专门为现代迁移学习管道设计的困难任务。我们的成果来自于一个直接统一的文本到文本框架、新的 C4 数据集以及系统研究的深入分析。此外，我们还提供了该领域的实证概述以及对该领域现状的看法。我们很高兴看到继续利用迁移学习来实现一般语言理解的目标。

致谢

感谢 Grady Simon、Noah Fiedel、Samuel R.Bowman、Augustus Odena、Daphne Ippolito、Noah Constant、Orhan Firat、Ankur Bapna 和 Sebastian Ruder 对本稿的意见；Zak Stone 和 TFRC 团队的支持；Austin Tarango 对数据集创建的指导；Melvin Johnson、Dima Lepikhin、Katrin Tomanek、Jeff Klingner 和 Naveen Arivazhagan 对多任务机器翻译的见解；Neil Houlsby 对适配器层的意见；Olga Wichowska、Ola Spyra、Michael Banfield、Yi Lin 和 Frank Chen 在基础架构方面的帮助；Etienne Pot、Ryan Sepassi 和 Pierre Ruyssen 在 TensorFlow 数据集方面的合作；Rohan Anil 在 Common Crawl 下载管道方面的帮助；Robby Neale 和 Taku Kudo 在 SentencePiece 方面的工作；Jeffrey Li 指出了 C4 创建过程中缺失的细节；以及 Google Brain 团队其他成员的讨论和见解。

附录 A。捐款

科林设计了本项目的范围并撰写了本文，运行了第 3.1 至 3.6 节中的所有实验，并贡献了我们代码库的大部分内容。诺姆贡献了许多想法，包括文本到文本框架、无监督目标和数据集混合策略；实现了我们的基础 Transformer 模型及其架构变体；并运行了第 3.7 节中的实验。亚当负责该项目的所有工程方面，创建了 C4 数据集，实施了我们的数据集管道，并添加了各种基准数据集。凯瑟琳负责协调实验、撰写和更新文档、进行实验以帮助设计我们的基线，并为我们代码库的 许多部分做出了贡献。Sharan 提供了一些所需的数据集和预处理器，并进行了各种初步实验，此外还共同领导了代码库的开源工作。迈克尔拥有维诺格拉德数据集的所有方面，采集了我们使用的许多数据集，对我们的基础架构进行了各种改进和修正，并进行了一些初步实验。严琦（Yanqi）进行了实验并实施了方法，帮助确定了合理的基线，并帮助对第 3.7 节中的模型进行了最终微调。Wei 也帮助进行了最后的微调，并改进了我们的一些预处理器。Peter 制作了早期版本的预训练数据集原型，并解决了与 SQuAD 和 CNN/DM 任务相关的问题。所有作者都帮助我们确定了这项工作的范围和研究方向。

附录 B.将 WNLI 转换为我们的文本到文本格式

请注意，如第 2.4 节所述，我们并没有在 WNLI 的任何数据上进行训练。相反，在对 WNLI 测试集（第 3.7 节中的结果）进行评估时，我们将 WNLI 测试集转换为 "指代名词预测 "的文本到文本格式，这样我们就可以使用在 WSC 和 DPR 上训练的模型进行评估。我们的 WNLI 预处理器受到 He 等人（2019）提出的预处理器的启发。回顾一下，WNLI 中的示例由一个前提、一个假设和一个表示假设是真还是假的标签组成。使用第 2.4 节中的例子，假设是 "市议员拒绝向示威者发放许可证是因为他们害怕暴力"，前提是 "示威者害怕暴力"，标签是 "False"。我们首先找出前提中所有代词的位置（在我们的例子中为 "他们"）。然后，我们找出每个代词之前或之后作为假设（示例中的 "惧怕暴力"）子串的最大单词数，忽略大小写和标点符号。当前提包含多个候选代词时，我们会选择假设中最大子串前或后的代词。然后，我们用星号将前提中的代词突出显示出来。对于候选名词（与我们的模型预测进行比

较，以获得 "真 "或 "假 "标签），我们会从假设中移除匹配的子串，并选择性地使其成为非所有格（结果是 "the demonstrators"）。

附录 C。CNN/每日邮报预测示例

为了证明我们的模型能够生成流畅的摘要，我们在验证集上加入了几个来自最佳模型（T5-11B）的解码示例，以及地面实况摘要。这些示例是随机选取的，并非精挑细选。

1. **真相：在**南非克鲁格国家公园的战斗中，豹子在尖嘴猴腮的动物拒绝退让后放弃了。
   野生动物爱好者 Lisl
   豹子和豪猪从灌木丛中翻滚出来，在路边开始搏斗 - 穆尔曼女士在一旁观看。

   41 岁的野生动物爱好者 Lisl Moolman 在路上用相机捕捉到了这场奇异的战斗。

2. **地面真相：**研究人员称住宅是未被发现物种的温床
   对北卡罗来纳州 40 所住宅的研究发现了 8000 个细菌和古生物类群
   .加利福尼亚州的 11 座房屋揭示了我们中间存在着种类繁多的真菌 .

   预测：研究人员估计，室内生物群落约占无冰陆地的 0.5%，或约 24.7 万平方英里，几乎相当于地球的大小。
   对北卡罗来纳州 40 所房屋进行的一 项 研究利用基因组技术记录了 8000 多个细菌和古细 菌分类群。
   在对加利福尼亚州的 11 所房屋进行的检查中，我们发现了种类繁多的真菌。

3. 凯莉-吉森达纳（Kelly Gissendaner）的死刑执行被第二次推迟，因为她被发现注射的戊巴比妥（Pentobarbital）致死剂量出现混浊。
   乔治亚州总检察长萨姆-奥伦斯说："必须以符合宪法的方式执行死刑"。

   预言：星期一，格鲁吉亚死囚牢中唯一的女性凯莉-吉森达纳的死刑执行第二次被推迟。行刑小组发现，致命的戊巴比妥剂量出现混浊。

4. 本周，阿尔维斯与前队友约瑟夫-平托（Jose Pinto）共同发行了一首慈善单曲。
   西班牙媒体报道称，他将在今夏签约巴黎圣日耳曼。
   .

   预测：达尼-阿尔维斯没有入选巴西对阵法国和智利的友谊赛。这位巴塞罗那右后卫与何塞-平托一起发行了一首慈善单曲。

拉斐尔、沙泽尔、罗伯茨、李、纳兰、马特纳、周、李和刘

点击此处查看所有巴塞罗那最新新闻。

附录 D.预处理示例

本节将举例说明我们对每个数据集的预处理。

D.1 CoLA

原始输入：

句子约翰让比尔成为自己的主人。

已处理输入：可乐句子：约翰让比尔成为自己的主人。

原目标：1

加工目标：可接受

D.2 RTE

原始输入：

第 1 句：南斯拉夫的意大利人中有较小一部分定居在斯洛文尼亚（在 1991 年的全国人口普查中，约有 3000 名斯洛文尼亚居民宣称自己是意大利族人）。

句子 2：斯洛文尼亚有 3 000 名居民。

已处理输入：RTE sentence1：南斯拉夫的意大利人中有一小部分定居在斯洛文尼亚（在 1991 年的全国人口普查中，约有 3000 名斯洛文尼亚居民宣称自己是意大利族人）。 句子 2：斯洛文尼亚有 3,000 名居民。

原目标：1

已处理目标：not_entailment

D.3 MNLI

原始输入：

假设：圣路易斯红雀队一直在赢。

前提：是的，输球就是输球，我的意思是，我来自圣路易斯，圣路易斯红雀队在那里的时

候，大部分时间都是一支输球的球队。

处理输入： MNLI 假设：圣路易斯红雀队一直在赢球。 前提： 是啊，输球是我的本能，我的意思是我来自圣路易斯，圣路易斯红雀队在那里的时候，大部分时间都在输球，但现在他们赢了。

拉斐尔、沙泽尔、罗伯茨、李、纳兰、马特纳、周、李和刘

原定目标：2

处理目标：矛盾

D.4 MRPC

原始输入：

第 1 句：拉姆斯菲尔德说："我们之所以采取行动，是因为我们从'9-11'事件的经验中看到了新的证据。

第 2 句：相反，美国之所以采取行动，是因为美国政府 "从新的角度，通过我们在 9 月 9 日的经历，看到了现有的证据"。
11 " .

已处理输入： mrpc sentence1：我们采取行动是因为我们看到了现有证据
拉姆斯菲尔德说："通过我们在'9-11'事 件 中的经验，我们对现有证据有了新的认识：相反，美国之所以采取行动，是因为美国政府 "通过我们在 9 月 11 日的经历，从新的角度看待了现有证据"。

原目标：1

加工目标：等效

D.5 QNLI

原始输入：

问题Jebe 死在哪里?

句子不久，成吉思汗将苏布台召回蒙古，而哲别则死在了返回撒马尔罕的路上。

已处理输入： qnli 问题：哲别死在哪里？成吉思汗不久将苏布台召回蒙古，而哲别则死在了返回撒马尔罕的路上。

原目标：0

处理过的目标：引申

D.6 QQP

原始输入：

问题 1：在古罗马，哪些特质会让你备受青睐?

问题 2： 我如何获得作为应届生加入 IT 公司的机会？

已处理输入： qqp 问题1：在古罗马，你的哪些特质会让你非常受欢迎？ 问题2：我如何获得作为应届生加入 IT 公司的机会？

拉斐尔、沙泽尔、罗伯茨、李、纳兰、马特纳、周、李和刘

原目标：0

已处理目标：not_duplicate

D.7 SST2

原始输入：

句子：它证实了芬奇作为电影制作人的地位，他巧妙地将技术诀窍用于心理洞察。

处理过的输入：SST2 句子：它证实了芬奇作为电影制作人的地位，他巧妙地将技术诀窍运用到心理洞察上
.

原目标：1

处理目标：正

D.8 STSB

原始输入：

第 1 句：周三，我们无法立即联系到 Puretunes 的代表发表评论。

第 2 句：周四，无法找到 Puretunes 的代表就该诉讼发表评论。

已处理输入：stsb sentence1：周三无法立即联系到 Puretunes 的代表置评：Puretunes 代表无法
周四，该公司未能就该诉讼发表评论。

原定目标：3.25

处理目标：3.2

D.9 CB

原始输入：

假设价值有助于
前提：虚无的大脑瓦伦斯，贤惠的男仆瓦伦斯。为什么不能
那个家伙选择了他自己的那部分泰坦尼克号解剖图作为轴心？他以为自己是在帮忙吗？

处理过的输入：CB 假设：Valence 是帮助前提： Valence 虚空脑、
贤内助瓦伦斯这家伙怎么就不能选择自己的那部分泰坦尼克式的解剖结构作为轴心呢？他

以为自己是在帮忙吗?

原目标：1

处理目标：矛盾

拉斐尔、沙泽尔、罗伯茨、李、纳兰、马特纳、周、李和刘

D.10 COPA

原始输入：

问题： 效果

前提：全国爆发政治暴力。

选择 1：许多市民搬迁到国会大厦。

选择 2：许多公民到其他地区避难。

选择 1：许多公民迁往首都：前提：全国爆发了政治暴力。 问题：影响

原目标：1

已处理目标：正确

D.11 MultiRC

原始输入：

请回答：只有馅饼可吃，而不是传统的早餐食品

段落：寄语 1：从前，有一只松鼠叫乔伊。
寄语 2：乔伊喜欢到外面和他的表弟吉米一起玩。

生3：乔伊和吉米一起玩愚蠢的游戏，总是笑个不停。
生4：有一天，乔伊和吉米一起去朱莉阿姨家的池塘游泳。

寄语6：除了馅饼，他找不到任何吃的！
寄语7：通常，乔伊早餐会吃麦片、水果（一个梨）或燕麦粥。
送8：吃完饭后，他和吉米去池塘玩。
送9：在去池塘的路上，他们看见了他们的朋友

第10句：他们潜入水中，游了几个小时。
第11句：太阳出来了，但微风很冷。
第12句：乔伊和吉米从水里出来，开始步行回家。
第13句：他们的毛湿了，微风吹得他们很冷。

寄语15：乔伊穿上了一件有红点和绿点的蓝色衬衫：
两只松鼠吃了乔伊的妈妈茉莉做的食物，就去睡觉了。

问题乔伊起床吃早餐的那天早上，他为什么会感到惊讶?

处理过的输入：多路问题：为什么乔伊早上起来吃早餐时会感到惊讶？只有馅饼可吃，而不是传统的馅饼

早餐食品段：寄语1：从前，有一只松鼠叫乔伊。
寄语2：乔伊喜欢到外面和他的表弟吉米一起玩。
寄语 3：乔伊和吉米一起玩愚蠢的游戏，总是笑个不停。
寄语 4：有一天，乔伊和吉米一起去游泳。

拉斐尔、沙泽尔、罗伯茨、李、纳兰、马特纳、周、李和刘

第5句：乔伊一大早就起床，想在出发前吃点东西。
寄语6：除了馅饼，他找不到任何东西吃！
寄语7：通常，乔伊早餐会吃麦片、水果（一个梨）或燕麦粥。
寄语 8：吃完饭后，他和吉米来到池塘边。
寄语 9：在去池塘的路上，他们看见了他们的朋友杰克兔。
寄语 10：他们跳进水里，游了好一会儿。

发送11：太阳出来了，但微风很冷。
发送12：乔伊和吉米下了水，开始走回家。
13：他们的皮毛湿透了，微风吹得他们浑身发冷。

寄语15：乔伊穿上了一件有红点和绿点的蓝色衬衫。
两只松鼠吃了乔伊的妈妈茉莉做的食物后就去睡觉了。

原目标： 1

已处理目标： 正确

D.12 WiC

原始输入：

POS： N

第 1 句： 他的行为是经过深思熟虑的，具有侮辱性.....。

第 2 句： 陪审团的审议.

词： 审议

处理过的输入： WIC POS: N 句子 1: 他的行为的审议是侮辱性的: 陪审团的商议 ... word: deliberation

原目标： 0

已处理目标： 假

D.13 WSC 和 DPR

原始输入：

跨度 2 文本： 它 **跨**

度 1 文本： 稳定 **跨**

探索迁移学习的极限

度 2 指数：20

跨度 1 指数：1

正文马厩非常宽敞，有四个很好的牲口棚；院子里有一扇大窗户，让人感到舒适通风。

处理输入：WSC：马厩非常宽敞，有四个很好的牲口棚；院子里有一扇大窗户，窗外空气清新，令人心旷神怡。

拉斐尔、沙泽尔、罗伯茨、李、纳兰、马特纳、周、李和刘

原目标：1

已处理目标：稳定

D.14 美国有线电视新闻网/每日邮报

马鲁阿内-费莱尼和阿德南-贾努扎伊继续向世界展示，他们不仅是队友，还是最好的伙伴。

在周三对阵纽卡斯尔的比赛之前，曼联和比利时双雄都在周一晚上发布了自己在一家餐厅的照片。

阿德南-贾努扎伊和马鲁阿内-费莱尼与朋友在舞池中合影留念

曼彻斯特联队和比利时双星法莱尼和贾努扎伊在场内外都是好朋友 ... 曼彻斯特联队王牌法莱尼跑到替补席上，与好友贾努扎伊一起庆祝对阵 Qpr 的进球 ... 背景中的迪斯科效果增加了理论依据，但贾努扎伊似乎并不介意，因为

他们随后在舞池中与其他朋友一起摆出各种姿势。

不过，周三对阵纽卡斯尔的胜利至少会让主帅范加尔感到欣慰。

贾努扎伊和范佩西与法莱尼一起在西布朗球场曼彻斯特联队球迷面前庆祝 . 贾努扎伊接受了曼彻斯特联队荷兰籍主教练路易斯-范加尔（Louis van gaal）的谆谆教诲 . 贾努扎伊和法莱尼与一些朋友一起在纽卡斯尔比赛前走上舞池 .

处理的输入：概述：马鲁安-费莱尼和阿德南-贾努扎伊继续向世界展示他们不仅是队友，也是最佳拍档。在周三对阵纽卡斯尔的比赛之前，曼联和比利时双雄都在周一晚上发布了自己在一家餐厅的照片。阿德南-贾努扎伊和马鲁安-费莱尼在舞池中与朋友合影。

曼彻斯特联队和比利时双雄法莱尼和贾努扎伊在场内外都是好朋友 ... 曼彻斯特联队王牌法莱尼跑到替补席上，与好友贾努扎伊一起庆祝对阵 Qpr 的进球 ... 背景中的迪斯科效果增加了理论依据，但贾努扎伊似乎并不介意，因为

他们随后在舞池中与其他朋友一起摆出各种姿势。

不过，周三对阵纽卡斯尔的胜利至少会让主帅范加尔感到欣慰。

贾努扎伊和范佩西与法莱尼一起在西布朗球场曼彻斯特联队球迷面前庆祝。

曼彻斯特联队的荷兰籍主教练路易斯-范加尔（Louis van gaal）在与纽卡斯尔队的比赛开始前，与贾努扎伊和费莱尼一起在舞池中跳舞。

曼彻斯特联队将在周三的英超联赛中迎战纽卡斯尔队，红魔将争取七场联赛客场胜利中的第二场胜利，范加尔的球队目前领先第四名利物浦两分。

曼彻斯特联队将在周三的英超联赛中迎战纽卡斯尔队，红魔将寻求七场联赛客场胜利中的第二场胜利，范加尔的球队目前领先第四名利物浦两分。

D.15 SQuAD

原始输入：

问题病人肺部氧气浓度增加会取代什么？

背景：高压氧（高压）医学使用特殊的氧气舱
以增加病人周围的氧气分压，必要时也可增加医务人员周围的氧气分压。一氧化碳中毒、气性坏疽和减压病（"弯腰症"）有时都需要使用这些设备进行治疗。增加肺中的氧气浓度有助于将一氧化碳从血红蛋白的血红素组中置换出来。氧气对导致气性坏疽的厌氧细菌有毒，因此增加氧气分压有助于杀死它们。
潜水员在潜水后减压过快时会出现减压病。
潜水时，惰性气体（主要是氮气和氦气）在血液中形成气泡。尽快增加 O 2 的压力是治疗的一部分。

处理过的输入：问题：病人肺部氧气浓度增加会取代什么？ 背景：高压氧（高压）医学使用特殊的氧舱可增加病人周围的氧气分压，必要时也可增加医务人员周围的氧气分压。一氧化碳中毒、气性坏疽和减压病（"弯腰症"）有时都需要使用这些设备进行治疗。增加肺中的氧气浓度有助于将一氧化碳从血红蛋白的血红素组中置换出来。氧气对导致气性坏疽的厌氧菌有毒，因此增加氧气分压有助于杀死它们。潜水员如果在潜水后减压过快，会在血液中形成惰性气体气泡，主要是氮气和氦气，从而导致减压病。尽快增加 O 2 的压力是治疗方法之一。

拉斐尔、沙泽尔、罗伯茨、李、纳兰、马特纳、周、李和刘

原始目标：一氧化碳

加工目标：一氧化碳

D.16 WMT 英语到德语

原始输入："她写道："路易吉经常对我说，他从来不希望兄弟俩走上法庭。

已处理输入：将英文翻译成德文："路易吉经常对我说，他从来不希望兄弟俩走上法庭，"
她写道。

原文目标："Luigi 经常对我说，他不希望看到 Brüder 在 Gericht 降落"，她写道。

处理后的目标："路易吉经常对我说，他不愿意让布鲁德的孩子们离开他的家"，她写道。

D.17 WMT 英语到法语

原始输入：这幅由斯皮策望远镜拍摄的红外图像显示了无数代恒星的 "全家福"：最古老的恒星
而更难识别的是恒星产房中粉红色的 "新生儿"。

已处理输入：将英语翻译成法语：这张由斯皮策望远镜拍摄的红外图像显示了无数代恒星的 "全家福
"：最古老的恒星被看作是蓝色的小点，而更难辨认的是恒星产房里粉红色的 "新生儿"。

原始目标斯皮策望远镜拍摄的这张红外照片显示了无数代陨石的 "家庭肖像"：最古老的陨石是
蓝色的，而最难拍摄的陨石是玫瑰色的。
à identifier, sont les "nouveau-nés" dans la salle d'accouchement de l'univers.

处理后的目标：斯皮策望远镜拍摄的红外图像显示了无数代陨石的 "家庭肖像"：最古老的陨石
是蓝色的，而最难拍摄的陨石是玫瑰色的。
à identifier, sont les "nouveau-nés" dans la salle d'accouchement de l'univers.

D.18 WMT 英语到罗马尼亚语

原始输入：塔可钟（Taco Bell）表示，计划到 2022 年在美国新增 2000 家分店。

已处理输入：将英语翻译成罗马尼亚语：塔可钟（Taco Bell）说，它计划到 2022 年在美国增
加 2000 家分店。

原始目标：塔可钟（Taco Bell）确认，到 2022 年，意向，将在澳大利亚减少 2000 家餐厅。

加工目标：塔可钟（Taco Bell）确认，到 2022 年，，。

拉斐尔、沙泽尔、罗伯茨、李、纳兰、马特纳、周、李和刘

附录 E。所有实验中每项任务的得分

下表列出了第 3.2 至 3.6 节所述实验中每项任务的得分情况。

		胶水																	超级胶水														WMT		
		得分	CoLA	SST-2	MRPC	MRPC	STSB	STSB	QQP	QQP	MNLIm	MNLImm	QNLI	RTE	CNN/DM			SQuAD		得分	BoolQ	CB	CB	COPA	MultiRC	MultiRC	ReCoRD	ReCoRD	RTE	WiC	WSC	EnDe	EnFr	恩罗
表格	实验	平均	MCC	Acc	F1	Acc	PCC	SCC	F1	Acc	Acc	Acc	Acc	Acc	R-1-F	R-2-F	R-L-F	EM	F1	平均	Acc	F1	Acc	Acc	F1	EM	F1	EM	Acc	Acc	Acc	BLEU	BLEU	BLEU
1	⋆ 基准平均值	83.28	53.84	92.68	92.07	88.92	88.02	87.94	88.67	91.56	84.24	84.57	90.48	76.28	41.33	19.24	38.77	80.88	88.81	71.36	76.62	91.22	91.96	66.20	66.13	25.78	69.05	68.16	75.34	68.04	78.56	26.98	39.82	27.65
1	基线标准偏差	0.235	1.111	0.569	0.729	1.019	0.374	0.418	0.108	0.070	0.291	0.231	0.361	1.393	0.065	0.065	0.058	0.343	0.226	0.416	0.365	3.237	2.560	2.741	0.716	1.011	0.370	0.379	1.228	0.850	2.029	0.112	0.090	0.108
1	无需预先培训	66.22	12.29	80.62	81.42	73.04	72.58	72.97	81.94	86.62	68.02	67.98	75.69	58.84	39.19	17.60	36.69	50.31	61.97	53.04	65.38	71.61	76.79	62.00	59.10	0.84	20.33	17.95	54.15	54.08	65.38	25.86	39.77	24.04
2	⋆ 编码/解码、去噪	83.28	53.84	92.68	92.07	88.92	88.02	87.94	88.67	91.56	84.24	84.57	90.48	76.28	41.33	19.24	38.77	80.88	88.81	71.36	76.62	91.22	91.96	66.20	66.13	25.78	69.05	68.16	75.34	68.04	78.56	26.98	39.82	27.65
2	编码/解码、共享、去噪	82.81	55.24	91.86	91.58	88.24	87.43	87.58	88.69	91.60	83.88	84.01	90.23	73.65	41.11	18.78	38.48	80.63	88.49	70.73	77.13	95.04	96.43	65.00	66.16	22.98	68.95	68.09	70.76	68.18	75.96	26.72	39.03	27.46
2	编码/解码，6 层，去噪	80.88	46.26	92.09	91.51	87.99	87.01	86.76	87.93	90.97	82.20	82.41	88.83	71.48	40.83	18.97	38.31	77.59	86.07	68.42	73.79	91.70	92.86	67.00	61.02	19.62	61.26	60.33	72.20	65.99	75.00	26.38	38.40	26.95
2	语言模型、去噪	74.70	24.50	90.60	86.08	78.92	85.22	85.42	85.40	88.99	76.72	77.05	86.02	64.62	39.49	17.93	36.91	61.14	71.37	55.02	65.47	60.08	71.43	58.00	43.03	2.94	53.35	52.31	53.07	58.62	63.46	25.09	35.28	25.86
2	前缀 LM、去噪	81.82	49.99	92.43	91.43	88.24	87.20	86.98	88.41	91.39	82.32	82.93	88.71	74.01	40.46	18.61	37.90	78.94	87.31	68.11	75.50	93.37	91.07	60.00	63.43	21.20	65.03	64.11	71.48	65.67	73.08	26.43	37.98	27.39
2	Enc/dec，LM	79.56	42.03	91.86	91.64	88.24	87.13	87.00	88.21	91.15	81.68	81.66	88.54	65.70	40.67	18.59	38.13	76.02	84.85	64.29	72.23	85.74	89.29	57.00	60.53	16.26	59.28	58.30	65.34	64.89	70.19	26.27	39.17	26.86
2	编码/解码、共享、LM	79.60	44.83	92.09	90.20	85.78	86.03	85.87	87.77	91.02	81.74	82.29	89.16	65.34	40.16	18.13	37.59	76.35	84.86	63.50	70.49	91.41	87.50	55.00	60.21	16.89	57.83	56.73	63.54	63.48	70.19	26.62	39.17	27.05
2	编码/解码，6 层，LM	78.67	38.72	91.40	90.40	86.52	86.82	86.49	87.87	91.03	80.99	80.92	88.05	65.70	40.29	18.26	37.70	75.32	84.06	64.06	71.38	85.25	89.29	60.00	57.56	16.79	55.22	54.30	66.79	63.95	71.15	26.13	38.42	26.89
2	语言模型，LM	73.78	28.53	89.79	85.23	78.68	84.22	84.00	84.88	88.70	74.94	75.77	84.84	58.84	38.97	17.54	36.37	53.81	64.55	56.51	64.22	59.92	71.43	64.00	53.04	1.05	46.81	45.78	58.84	56.74	69.23	25.23	34.31	25.38
2	前缀 LM、LM	79.68	41.26	92.09	90.11	86.27	86.82	86.32	88.35	91.35	81.71	82.02	89.04	68.59	39.66	17.84	37.13	76.87	85.39	64.86	71.47	93.37	91.07	57.00	58.67	16.89	59.25	58.16	64.26	66.30	71.15	26.28	37.51	26.76
4	带前缀的语言建模	80.69	44.22	93.00	91.68	88.48	87.20	87.18	88.39	91.41	82.66	83.09	89.29	68.95	40.71	18.94	38.15	77.99	86.43	65.27	73.55	83.95	87.50	55.00	59.65	18.89	61.76	60.76	68.59	65.67	73.08	26.86	39.73	27.49
4	BERT 式（Devlin 等人，2018 年）	82.96	52.49	92.55	92.79	89.95	87.68	87.66	88.47	91.44	83.60	84.05	90.33	75.45	41.27	19.17	38.72	80.65	88.24	69.85	76.48	94.37	94.64	61.00	63.29	25.08	66.76	65.85	72.20	69.12	75.00	26.78	40.03	27.41
4	除尘	73.17	22.82	87.16	86.88	81.13	84.03	83.82	86.38	89.90	76.30	76.34	84.18	58.84	40.75	18.59	38.10	67.61	76.76	58.47	69.17	63.70	78.57	56.00	59.85	12.70	45.52	44.36	57.04	64.89	68.27	26.11	39.30	25.62
5	BERT 式（Devlin 等人，2018 年）	82.96	52.49	92.55	92.79	89.95	87.68	87.66	88.47	91.44	83.60	84.05	90.33	75.45	41.27	19.17	38.72	80.65	88.24	69.85	76.48	94.37	94.64	61.00	63.29	25.08	66.76	65.85	72.20	69.12	75.00	26.78	40.03	27.41
5	MASS-style（Song 等人，2019 年）	82.32	47.01	91.63	92.53	89.71	88.21	88.18	88.58	91.44	82.96	83.67	90.02	77.26	41.16	19.16	38.55	80.10	88.07	69.28	75.08	84.98	89.29	63.00	64.46	23.50	66.71	65.91	72.20	67.71	78.85	26.79	39.89	27.55
5	⋆ 替换损坏的跨度	83.28	53.84	92.68	92.07	88.92	88.02	87.94	88.67	91.56	84.24	84.57	90.48	76.28	41.33	19.24	38.77	80.88	88.81	71.36	76.62	91.22	91.96	66.20	66.13	25.78	69.05	68.16	75.34	68.04	78.56	26.98	39.82	27.65
5	丢弃损坏的代币	84.44	60.04	92.89	92.79	89.95	87.28	86.85	88.56	91.54	83.94	83.92	90.74	79.42	41.27	19.31	38.70	80.52	88.28	68.67	75.90	96.02	94.64	56.00	65.06	23.92	65.54	64.60	71.12	67.40	74.04	27.07	39.76	27.82
6	腐败率 = 10%	82.82	52.71	92.09	91.55	88.24	88.19	88.15	88.47	91.40	83.50	84.51	90.33	75.45	41.05	19.00	38.53	80.38	88.36	69.55	74.98	92.37	92.86	62.00	66.04	24.66	67.93	67.09	70.76	67.24	75.96	26.87	39.28	27.44
6	⋆腐败率 = 15%	83.28	53.84	92.68	92.07	88.92	88.02	87.94	88.67	91.56	84.24	84.57	90.48	76.28	41.33	19.24	38.77	80.88	88.81	71.36	76.62	91.22	91.96	66.20	66.13	25.78	69.05	68.16	75.34	68.04	78.56	26.98	39.82	27.65
6	腐败率 = 25	83.00	53.47	93.00	92.44	89.46	87.36	87.36	88.68	91.53	84.44	84.15	90.77	74.01	41.69	19.54	39.14	80.96	88.61	70.48	76.39	93.02	92.86	68.00	65.46	24.66	68.20	67.39	73.65	67.87	72.12	27.04	39.83	27.47
6	腐败率 = 50%	81.27	46.26	91.63	91.11	87.99	87.87	87.64	88.70	91.57	83.64	84.10	90.24	70.76	41.51	19.32	38.89	79.80	87.76	70.33	75.02	93.05	92.86	68.00	62.97	24.13	64.94	64.13	72.20	68.50	77.88	27.01	39.90	27.49
7	⋆ 基线 (i.i.d.)	83.28	53.84	92.68	92.07	88.92	88.02	87.94	88.67	91.56	84.24	84.57	90.48	76.28	41.33	19.24	38.77	80.88	88.81	71.36	76.62	91.22	91.96	66.20	66.13	25.78	69.05	68.16	75.34	68.04	78.56	26.98	39.82	27.65
7	平均跨度 = 2	83.54	53.82	92.20	93.05	90.44	87.85	87.71	88.42	91.40	84.28	84.46	90.88	77.62	41.23	19.39	38.69	82.09	89.69	72.20	77.06	90.43	91.07	70.00	66.28	26.13	71.34	70.61	75.45	68.34	78.85	26.76	39.99	27.63
7	平均跨度 = 3	83.49	53.90	92.43	92.25	89.46	87.49	87.53	88.72	91.51	84.85	84.84	90.99	77.26	41.50	19.62	38.94	81.84	89.66	72.53	76.85	94.37	94.64	70.00	67.64	28.75	70.84	69.90	74.73	67.71	77.88	26.86	39.65	27.62
7	平均跨度 = 5	83.40	52.12	93.12	92.63	89.71	88.70	88.47	88.84	91.64	84.32	84.29	90.79	76.90	41.39	19.24	38.82	82.05	89.79	72.23	77.06	83.06	89.29	69.00	68.16	30.12	71.36	70.53	75.81	69.91	79.81	26.88	39.40	27.53
7	平均跨度 = 10	82.85	50.11	92.09	91.95	88.97	88.45	88.22	88.86	91.63	84.34	84.28	91.07	76.17	41.38	19.33	38.80	81.84	89.39	70.44	76.45	87.40	89.29	65.00	66.87	29.59	69.82	68.94	72.56	67.55	75.96	26.79	39.49	27.69
8	⋆ C4	83.28	53.84	92.68	92.07	88.92	88.02	87.94	88.67	91.56	84.24	84.57	90.48	76.28	41.33	19.24	38.77	80.88	88.81	71.36	76.62	91.22	91.96	66.20	66.13	25.78	69.05	68.16	75.34	68.04	78.56	26.98	39.82	27.65
8	C4，未经过滤	81.46	48.01	91.63	92.72	89.95	87.79	87.60	88.31	91.27	82.30	82.34	88.71	72.20	41.09	19.14	38.54	78.78	87.04	68.04	75.75	89.17	91.07	62.00	65.52	25.60	62.42	61.58	69.68	67.08	72.12	26.55	39.34	27.21
8	类似真实新闻	83.83	56.55	92.66	92.06	88.97	87.71	87.37	88.51	91.49	84.35	84.46	90.61	78.34	41.38	19.23	38.84	80.39	88.50	72.38	77.00	93.09	94.64	66.00	65.92	23.82	74.56	73.72	75.81	66.61	80.77	26.75	39.90	27.48
8	类 WebText	84.03	56.38	93.12	92.31	89.22	88.69	88.68	88.65	91.56	84.70	84.84	90.83	77.62	41.23	19.31	38.70	81.42	89.15	71.40	76.88	83.08	89.29	66.00	64.10	24.24	72.24	71.36	75.45	68.03	82.69	26.80	39.74	27.59
8	维基百科	81.85	45.53	92.32	91.67	88.24	85.62	86.40	88.37	91.34	82.61	83.25	90.96	77.26	41.39	19.31	38.81	81.29	89.18	68.01	76.12	56.03	80.36	67.00	65.01	25.92	69.03	68.06	74.73	67.08	76.92	26.94	39.69	27.67
8	维基百科 + 待定	83.65	55.53	92.78	92.41	89.22	86.67	86.27	89.47	92.29	84.38	83.45	91.94	76.90	41.22	19.28	38.67	82.08	89.70	73.24	76.22	95.40	92.86	69.00	51.59	50.93	69.53	68.51	77.62	66.93	81.73	26.77	39.63	27.57
9	⋆ 完整数据集	83.28	53.84	92.68	92.07	88.92	88.02	87.94	88.67	91.56	84.24	84.57	90.48	76.28	41.33	19.24	38.77	80.88	88.81	71.36	76.62	91.22	91.96	66.20	66.13	25.78	69.05	68.16	75.34	68.04	78.56	26.98	39.82	27.65
9	2^{29} （64 次重复）	82.87	53.82	92.78	91.79	88.73	87.56	87.58	88.73	91.54	84.07	84.21	90.59	73.65	41.18	19.19	38.67	80.97	88.90	72.03	76.76	92.96	92.86	66.00	65.11	26.76	69.35	68.49	75.81	67.24	82.69	26.83	39.74	27.63
9	2^{27} （256 次重复）	82.62	50.60	92.32	92.07	88.73	87.83	87.60	88.65	91.54	83.43	84.37	90.12	75.81	41.24	19.20	38.70	79.78	87.63	69.97	75.29	93.42	91.07	63.00	61.82	23.61	66.27	65.39	73.65	66.30	80.77	27.02	39.71	27.33
9	2^{25} （重复 1 024 次）	79.55	43.84	91.28	89.32	85.05	85.92	85.74	88.05	91.09	81.29	81.72	87.90	69.31	40.66	18.57	38.13	76.27	84.58	64.76	72.63	83.97	82.14	64.00	59.39	17.94	56.94	56.04	64.98	65.20	73.08	26.38	39.56	26.80
9	2^{23} （4 096 次重复）	76.34	32.68	89.45	89.84	86.03	83.49	83.42	87.18	90.61	77.80	78.69	85.47	64.62	40.16	18.33	37.66	70.92	80.20	59.29	69.85	73.48	73.21	56.00	57.66	14.38	46.69	45.79	59.57	65.05	68.27	26.37	38.84	25.81
10	⋆ 所有参数	83.28	53.84	92.68	92.07	88.92	88.02	87.94	88.67	91.56	84.24	84.57	90.48	76.28	41.33	19.24	38.77	80.88	88.81	71.36	76.62	91.22	91.96	66.20	66.13	25.78	69.05	68.16	75.34	68.04	78.56	26.98	39.82	27.65
10	适配器层，$d = 32$	80.52	45.33	91.63	90.59	86.76	88.38	88.06	86.99	90.26	83.63	83.94	90.72	67.15	34.50	15.08	32.15	79.32	87.70	60.40	65.32	50.87	73.21	52.00	58.61	19.41	65.50	64.58	62.09	64.58	73.08	13.84	17.88	15.54
10	适配器层，$d = 128$	81.51	45.35	92.89	91.49	88.24	87.73	87.65	87.73	90.93	83.64	84.09	90.52	72.56	36.71	16.62	34.37	79.47	87.61	63.03	69.20	52.21	75.00	56.00	61.08	18.05	67.94	66.97	68.59	66.77	73.08	19.83	27.50	22.63
10	适配器层，$d = 512$	81.54	44.25	93.35	91.00	87.25	88.74	88.44	88.02	91.15	83.08	83.80	89.62	74.37	38.63	17.78	36.25	79.18	87.32	64.30	73.18	59.86	71.43	56.00	62.94	18.57	66.56	65.74	70.76	67.87	74.04	23.45	33.98	25.81
10	适配器层，$d = 2048$	82.62	49.86	92.55	91.30	87.99	88.46	88.35	88.36	91.40	83.63	83.18	90.66	76.53	39.44	18.30	37.06	79.40	87.36	68.61	74.53	88.00	91.07	58.00	61.10	18.89	66.73	66.06	73.29	71.16	75.96	25.64	36.92	26.93
10	逐渐解冻	82.50	51.74	91.97	92.61	89.71	87.27	86.90	88.26	91.35	83.42	83.49	89.71	75.09	40.88	18.95	38.40	79.17	87.30	70.79	75.51	93.09	94.64	70.00	62.03	21.51	65.69	64.79	72.92	69.12	77.89	26.71	39.02	26.93
11	⋆ 基准线（预培训/微调）	83.28	53.84	92.68	92.07	88.92	88.02	87.94	88.67	91.56	84.24	84.57	90.48	76.28	41.33	19.24	38.77	80.88	88.81	71.36	76.62	91.22	91.96	66.20	66.13	25.78	69.05	68.16	75.34	68.04	78.56	26.98	39.82	27.65
11	平等	76.13	39.47	90.94	82.90	75.74	78.83	78.44	86.45	89.71	82.08	82.92	90.13	59.93	40.95	19.02	38.39	76.51	85.61	63.37	73.06	82.37	83.93	65.00	60.89	17.52	60.51	59.70	61.01	60.03	65.38	23.89	34.31	26.78
11	示例-比例，$K = 2^{16}$	80.45	42.07	91.97	90.97	87.50	85.41	85.04	86.89	90.10	83.01	83.66	90.74	72.56	41.16	19.04	38.59	77.25	85.72	69.95	76.67	86.38	89.29	70.00	65.93	27.91	62.78	61.95	76.90	65.83	73.08	24.35	34.99	27.10
11	示例-比例，$K = 2^{17}$	81.56	47.35	91.40	91.55	88.24	86.15	85.93	86.94	90.06	82.76	84.12	90.79	75.09	41.06	19.12	38.47	77.00	85.87	67.91	77.89	77.54	85.71	57.00	67.78	27.07	61.51	60.54	79.06	65.20	74.04	24.36	35.00	27.25
11	示例-比例，$K = 2^{18}$	81.67	46.85	91.63	91.99	88.73	87.68	87.20	86.93	90.35	83.30	84.01	91.47	73.29	40.96	19.07	38.43	78.17	86.74	67.94	76.57	78.88	87.50	62.00	67.70	30.85	63.43	62.54	76.53	65.67	67.31	24.57	35.19	27.39
11	示例-比例，$K = 2^{19}$	81.42	45.94	91.63	92.20	89.22	88.44	88.32	86.84	90.10	83.73	84.29	91.84	70.40	41.26	19.24	38.71	79.78	88.15	67.30	75.66	75.59	87.50	59.00	68.22	30.64	65.32	64.29	73.65	65.05	69.23	25.21	36.30	27.76
11	示例-比例，$K = 2^{20}$	80.80	42.55	92.78	91.27	87.99	88.36	88.10	86.10	89.62	84.15	84.26	92.20	68.95	41.05	19.24	38.46	80.36	88.27	67.38	73.21	76.18	83.93	62.00	67.57	26.86	66.12	65.22	76.90	64.73	69.23	25.66	36.93	27.68
11	示例-比例，$K = 2^{21}$	79.83	44.45	91.28	89.00	84.31	87.54	87.40	84.93	88.53	82.54	84.16	90.85	67.87	40.51	18.79	37.92	79.50	87.48	65.10	71.16	68.88	85.71	57.00	62.75	23.40	64.50	63.65	72.92	64.11	71.15	25.82	37.22	27.13
11	温标，$T = 2$	81.90	54.00	91.74	90.56	86.76	85.11	84.60	86.40	89.74	83.47	84.15	91.51	72.56	41.09	19.28	38.54	79.42	87.77	69.92	76.73	92.37	92.86	57.00	69.80	31.90	66.65	65.74	72.92	67.08	75.96	25.42	36.72	27.20
11	温标，$T = 4$	80.56	45.38	91.97	89.68	85.78	83.13	82.76	86.39	90.00	82.78	84.19	91.16	73.65	41.09	19.22	38.51	77.99	86.81	69.54	76.76	97.36	96.43	59.00	68.10	31.48	64.26	63.27	74.73	64.26	71.15	25.04	35.82	27.45
11	温标，$T = 8$	77.21	40.07	91.06	88.11	83.33	79.20	79.06	86.60	89.90	83.05	83.56	90.21	59.93	41.01	19.10	38.40	77.14	85.99	66.07	73.94	93.70	94.64	60.00	66.36	26.86	63.46	62.60	62.09	63.32	65.38	24.55	35.35	27.17
12	⋆ 无监督预训练 + 微调	83.28	53.84	92.68	92.07	88.92	88.02	87.94	88.67	91.56	84.24	84.57	90.48	76.28	41.33	19.24	38.77	80.88	88.81	71.36	76.62	91.22	91.96	66.20	66.13	25.78	69.05	68.16	75.34	68.04	78.56	26.98	39.82	27.65
12	多任务培训	81.42	45.94	91.63	92.20	89.22	88.44	88.32	86.84	90.10	83.73	84.29	91.84	70.40	41.26	19.24	38.71	79.78	88.15	67.30	75.66	75.59	87.50	59.00	68.22	30.64	65.32	64.29	73.65	65.05	69.23	25.21	36.30	27.76
12	多任务预培训 + 微调	83.11	51.42	92.66	91.73	88.73	88.06	87.70	88.61	91.61	84.09	84.31	91.85	76.53	41.15	19.12	38.59	80.26	88.50	71.03	79.54	81.69	87.50	65.00	70.72	31.48	65.94	65.03	81.23	68.18	73.08	27.08	39.80	28.07
12	留白式多任务训练	81.98	48.00	93.23	91.72	88.24	87.76	87.32	88.61	91.44	84.00	84.11	90.79	72.20	41.34	19.05	38.77	79.97	88.10	71.68	78.35	86.76	89.29	66.00	68.09	29.49	66.23	65.27	79.06	68.65	78.85	26.93	39.79	27.87
12	有监督的多任务预培训	79.93	36.60	92.43	91.58	88.24	87.03	86.78	88.15	91.20	82.87	83.16	90.13	70.76	41.12	18.96	38.49	77.38	85.65	65.36	75.66	68.87	83.93	58.00	64.81	21.93	55.37	54.61	71.12	67.40	75.96	26.81	40.13	28.04
13	⋆ 基准线	83.28	53.84	92.68	92.07	88.92	88.02	87.94	88.67	91.56	84.24	84.57	90.48	76.28	41.33	19.24	38.77	80.88	88.81	71.36	76.62	91.22	91.96	66.20	66.13	25.78	69.05	68.16	75.34	68.04	78.56	26.98	39.82	27.65
13	*1*× 大小，*4*× 训练步骤	85.33	60.29	93.81	94.06	91.67	89.42	89.25	89.15	91.87	86.01	85.70	91.63	78.34	41.52	19.33	38.96	82.45	90.19	74.72	79.17	94.75	92.86	71.00	67.34	29.70	72.63	71.59	78.34	72.10	82.69	27.08	40.66	27.93
13	*1* 倍大小，*4* 倍批量	84.60	56.08	93.12	92.31	89.22	88.85	88.84	89.35	92.07	85.98	86.13	91.07	80.14	41.70	19.42	39.08	82.52	90.21	74.64	78.78	93.69	94.64	72.00	68.09	30.95	74.73	73.90	76.53	70.06	81.73	27.07	40.60	27.84
13	*2* 倍大小，2 *倍*训练步骤	86.18	62.04	93.69	93.36	90.69	89.18	89.23	89.35	92.05	87.23	87.05	92.68	81.95	41.74	19.66	39.14	84.18	91.29	77.18	80.98	97.36	96.43	74.00	71.34	35.68	77.11	76.34	80.51	69.28	85.58	27.52	41.03	28.19
13	*4* 倍大小，*1* 倍训练步骤	85.91	57.58	94.38	92.67	89.95	89.60	89.60	89.44	92.14	87.05	87.12	93.12	83.39	41.60	19.73	39.08	83.86	91.32	78.04	81.38	89.09	94.64	73.00	73.74	40.40	78.25	77.40	81.59	70.22	91.35	27.47	40.71	28.10
13	*4*× 组合	84.77	56.14	93.46	93.31	90.67	89.71	89.60	89.62	92.24	86.22	86.53	91.60	77.98	42.10	20.10	39.56	83.09	90.40	71.74	77.58	89.85	91.07	66.00	69.32	29.49	72.67	71.94	76.90	69.12	72.12	28.05	40.53	28.09
13	*4*× 组合，仅微调	84.05	54.78	92.78	93.15	90.44	88.34	88.12	89.27	91.97	85.33	85.88	90.98	77.62	41.66	19.57	39.12	82.36	89.86	71.56	77.43	90.07	92.86	69.00	67.31	26.34	70.47	69.64	75.45	68.18	74.04	27.55	40.22	28.09

表 16：本文所有实验中每项任务的得分。在第一列中，我们列出了某项实验的结果浓缩表。与正文中一样，标有⋆的一行表示我们的基线模型（见第 3.1 节）。

参考资料

Rami Al-Rfou、Dokook Choe、Noah Constant、Mandy Guo 和 Llion Jones。具有更深自我关注的字符级语言建模。2019年*AAAI人工智能大会论文集。*

Rohan Anil、Vineet Gupta、Tomer Koren 和 Yoram Singer.大规模学习的内存效率自适应优化。*arXiv preprint arXiv:1901.11150*, 2019.

Naveen Arivazhagan、Ankur Bapna、Orhan Firat、Dmitry Lepikhin、Melvin Johnson、Maxim Krikun、Mia Xu Chen、Yuan Cao、George Foster、Colin Cherry 等：野生大规模多语言神经机器翻译：*ArXiv preprint arXiv:1907.05019*, 2019.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton.层归一化》，*arXiv preprint arXiv:1607.06450*, 2016.

Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli.Cloze-driven pretraining of self-attention networks. *arXiv preprint arXiv:1903.07785*, 2019.

Dzmitry Bahdanau、Kyunghyun Cho 和 Yoshua Bengio。通过联合学习对齐和翻译的神经机器翻译。2015年*第三届学习表征国际会议。*

Ankur Bapna、Naveen Arivazhagan 和 Orhan Firat.神经机器翻译的简单、可扩展适应性。*arXiv preprint arXiv:1909.08478*, 2019.

Iz Beltagy、Kyle Lo 和 Arman Cohan。SciBERT：科学文本的预训练语言模型。*2019年自然语言处理实证方法会议暨第九届自然语言处理国际联合会议（EMNLP-IJCNLP）论文集*，2019年。

Ondřej Bojar、Christian Buck、Christian Federmann、Barry Haddow、Philipp Koehn、Jo- hannes Leveling、Christof Monz、Pavel Pecina、Matt Post、Herve Saint-Amand 等：2014 年统计机器翻译研讨会成果。In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, 2014.

Ondřej Bojar、Rajen Chatterjee、Christian Federmann、Barry Haddow、Matthias Huck、Chris Hokamp、Philipp Koehn、Varvara Logacheva、Christof Monz、Matteo Negri 等：2015 年统计机器翻译研讨会成果。*第十届统计机器翻译研讨会*

论文集，2015 年。

Ondřej Bojar、Rajen Chatterjee、Christian Federmann、Yvette Graham、Barry Haddow、Matthias Huck、Antonio Jimeno Yepes、Philipp Koehn、Varvara Logacheva、Christof Monz 等。 2016 年机器翻译大会的研究成果。*首届机器翻译大会论文集*，2016 年。

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio.从连续空间生成句子。*arXiv preprint arXiv:1511.06349*, 2015.

Christian Buck、Kenneth Heafield 和 Bas Van Ooyen。来自普通抓取的 N-gram 计数和语言模型。在 *LREC*，2014 年。

里奇-卡鲁阿纳多任务学习*机器学习*》，28（1），1997 年。

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia.Semeval-2017 任务1：语义文本相似性--多语言和跨语言重点评估。*arXiv预印本arXiv:1708.00055*，2017。

Jianpeng Cheng, Li Dong, and Mirella Lapata.机器阅读的长短期记忆网络》，*arXiv preprint arXiv:1601.06733*, 2016.

Christopher Clark、Kenton Lee、Ming-Wei Chang、Tom Kwiatkowski、Michael Collins 和 Kristina Toutanova.BoolQ: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning.Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

Alexis Conneau 和 Douwe Kiela.SentEval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*, 2018.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes.从自然语言推理数据超级学习通用句子表征》，*arXiv preprint arXiv:1705.02364*, 2017.

Ido Dagan、Oren Glickman 和 Bernardo Magnini。PASCAL 识别文本引申挑战。*机器学习挑战研讨会*，2005 年。

Andrew M. Dai 和 Quoc V. Le.半监督序列学习。*神经信息处理系统进展*，2015 年。

Marie-Catherine De Marneff、Mandy Simons 和 Judith Tonhauser。承诺库：调查自然发生话语中的投射。In *Sinn und Bedeutung 23*, 2019.

邓佳、董伟、理查德-索彻、李佳、李凯、李菲菲。ImageNet：大规模分层图像数据库。*2009年IEEE计算机视觉与模式识别会议*，2009年。

Jacob Devlin、Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova.BERT：用于语

言理解的深度双向变换器预训练。*arXiv preprint arXiv:1810.04805*, 2018.

William B. Dolan and Chris Brockett.自动构建句法短语语料库。*第三届仿拟国际研讨会论文集（IWP2005）*，2005 年。

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiaoo-Wuen Hon. Unified language model pre-training for natural language understanding and generation. *arXiv preprint arXiv:1905.03197*, 2019.

Sergey Edunov、Myle Ott、Michael Auli 和 David Grangier.理解大规模反向翻译》，*arXiv preprint arXiv:1808.09381*, 2018.

Edouard Grave、Piotr Bojanowski、Prakhar Gupta、Armand Joulin 和 Tomas Mikolov。

学习 157 种语言的单词向量》，*arXiv preprint arXiv:1802.06893*, 2018.

亚历克斯-格雷夫斯用递归神经网络生成序列》，*arXiv preprint arXiv:1308.0850*, 2013.

Ivan Habernal、Omnia Zayed 和 Iryna Gurevych。C4Corpus：免费许可的多语言网络语料库。*第十届语言资源与评估国际会议（LREC'16）论文集*，第 914-922 页，2016 年。

何开明、张翔宇、任少清和孙健。图像识别的深度残差学习。2016年*IEEE计算机视觉与模式识别大会论文集。*

何开明、Ross Girshick 和 Piotr Dollár.反思 ImageNet 预训练》，*arXiv preprint arXiv:1811.08883*, 2018.

何鹏程、刘晓东、陈伟柱、高剑锋。用于常识推理的混合神经网络模型。*arXiv preprint arXiv:1907.11983*, 2019.

卡尔-莫里茨-赫尔曼（Karl Moritz Hermann）、托马斯-科西斯基（Tomas Kocisky）、爱德华-格雷芬斯特（Edward Grefenstette）、拉塞-埃斯佩霍尔特（Lasse Espeholt）、威尔-凯（Will Kay）、穆斯塔法-苏莱曼（Mustafa Suleyman）和菲尔-布隆索姆（Phil Blunsom）。教机器阅读和理解。*神经信息处理系统进展*，2015。

Joel Hestness、Sharan Narang、Newsha Ardalani、Gregory Diamos、Heewoo Jun、Hassan Kianinejad、Md.Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou.深度学习的扩展是可预测的，经验上是如此。*arXiv preprint arXiv:1712.00409*, 2017.

Felix Hill, Kyunghyun Cho, and Anna Korhonen.从无标签数据中学习句子的分布式表征》，*arXiv preprint arXiv:1602.03483*, 2016.

Geoffrey Hinton、Oriol Vinyals 和 Jeff Dean。提炼神经网络中的知识。*arXiv preprint arXiv:1503.02531*, 2015.

Neil Houlsby、 Andrei Giurgiu、 Stanislaw Jastrzebski、 Bruna Morrone、 Quentin De Laroussilhe、Andrea Gesmundo、Mona Attariyan 和 Sylvain Gelly。用于 NLP 的参数高效迁移学习。*arXiv preprint arXiv:1902.00751*, 2019.

杰里米-霍华德和塞巴斯蒂安-鲁德。用于文本分类的通用语言模型微调。*ArXiv preprint arXiv:1801.06146*, 2018.

Cheng-Zhi Anna Huang、 Ashish Vaswani、 Jakob Uszkoreit、 Ian Simon、 Curtis Hawthorne、 Noam Shazeer、 Andrew M. Dai、 Matthew D. Hoffman、 Monica Dinculescu 和 Dou- glas Eck。音乐转换器：生成具有长期结构的音乐。*第七届学习表征国际会议*，2018a。

Yanping Huang、Yonglong Cheng、Dehao Chen、HyoukJoong Lee、Jiquan Ngiam、Quoc V Le 和 Zhifeng Chen。GPipe：利用流水线并行性高效训练巨型神经网络。*arXiv preprint arXiv:1811.06965*, 2018b.

Minyoung Huh, Pulkit Agrawal, and Alexei A. Efros.是什么让 ImageNet 适合迁移学习？*arXiv preprint arXiv:1608.08614*, 2016.

Shankar Iyer、Nikhil Dandekar 和 Kornel Csernai。首个 Quora 数据集发布：Question pairs. https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs, 2017.

杨庆佳、埃文-谢尔哈默、杰夫-多纳休、谢尔盖-卡拉耶夫、乔纳森-龙、罗斯-吉尔希克、塞尔吉奥-瓜达拉马和特雷弗-达雷尔。Caffe：用于快速特征嵌入的卷积架构。*第 22 届 ACM 国际多媒体会议论文集*，2014 年。

焦晓琪、尹一春、尚立峰、蒋昕、陈晓、李琳琳、王芳、刘群。TinyBERT: Distilling BERT for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

Mandar Joshi、Eunsol Choi、Daniel S. Weld 和 Luke Zettlemoyer。TriviaQA： A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy.SpanBERT: Improving pre-training by representing and predicting spans. *arXiv preprint arXiv:1907.10529*, 2019.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu.探索语言建模的极限。*arXiv preprint arXiv:1602.02410*, 2016.

Nal Kalchbrenner、Edward Grefenstette 和 Phil Blunsom。用于句子建模的卷积神经网络。*第 52 届计算语言学协会年会论文集*，2014 年。

Nitish Shirish Keskar、Bryan McCann、Lav R. Varshney、Caiming Xiong 和 Richard Socher。CTRL：可控生成的条件转换器语言模型。*arXiv 预印本 arXiv:1909.05858*, 2019a。

Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher.通过跨度提取

统一问题解答和文本分类》，*arXiv preprint arXiv:1904.09286*, 2019b.

Daniel Khashabi、Snigdha Chaturvedi、Michael Roth、Shyam Upadhyay 和 Dan Roth。透过表面看本质：多句阅读理解挑战集。*计算语言学协会北美分会（NAACL）论文集*，2018年。

Ryan Kiros、Yukun Zhu、Ruslan R. Salakhutdinov、Richard Zemel、Raquel Urtasun、Antonio Torralba 和 Sanja Fidler。跳跃思维向量。*神经信息处理系统进展》*，2015 年。

Vid Kocijan, Ana-Maria Cretu, Oana-Maria Camburu, Yordan Yordanov, and Thomas Lukasiewicz.维诺格拉德模式挑战的惊人稳健诀窍》，*arXiv预印本arXiv:1905.06290*，2019.

Jakub Konečny`、Brendan McMahan 和 Daniel Ramage。联邦优化：*ArXiv preprint arXiv:1511.03575*, 2015.

Jakub Konečny`、H. Brendan McMahan、Felix X. Yu、Peter Richtárik、Ananda Theertha Suresh 和 Dave Bacon。联合学习：*ArXiv preprint arXiv:1610.05492*, 2016.

Simon Kornblith、Jonathon Shlens 和 Quoc V. Le.更好的 ImageNet 模型能更好地传输吗？ *arXiv preprint arXiv:1805.08974*, 2018.

Alex Krizhevsky.卷积神经网络并行化的一个怪招》，*arXiv preprint arXiv:1404.5997*, 2014.

Taku Kudo.子词正则化：用多个候选子词改进神经网络翻译模型。*arXiv preprint arXiv:1804.10959*, 2018.

Taku Kudo 和 John Richardson。SentencePiece：用于神经文本处理的简单且独立于语言的子单词标记器和去标记器。*arXiv preprint arXiv:1808.06226*, 2018.

Guillaume Lample 和 Alexis Conneau.跨语言语言模型预训练》，*arXiv preprint arXiv:1901.07291*, 2019.

蓝振中、陈明达、塞巴斯蒂安-古德曼、凯文-金佩尔、皮尤什-夏尔马和拉杜-索里库特。ALBERT: A lite BERT for self-supervised learning of language representa- tions. *ArXiv preprint arXiv:1909.11942*, 2019.

赫克托-莱维斯克、欧内斯特-戴维斯和莉奥拉-莫根斯滕。维诺格拉德模式挑战。*第十三届知识表示与推理原理国际会议*，2012 年。

Qi Li.文献调查：自然语言处理的领域适应算法
2012.

Chin-Yew Lin.ROUGE：用于自动评估摘要的软件包。In *Text summarization branches out*, 2004.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer.通过总结长序列生成维基百科》，*arXiv preprint arXiv:1801.10198*, 2018.

Peter J. Liu, Yu-An Chung, and Jie Ren.SummAE: Zero-shot abstractive text summarization using length-agnostic auto-encoders. *ArXiv preprint arXiv:1910.00998*, 2019a.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang.使用多任务深度神经网络进行语义分类和信息检索的再现学习。*计算语言学协会北美分会 2015 年会议论文集：人类语言技术*，2015 年。

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao.用于自然语言理解的多任务深度神经网络》，*arXiv preprint arXiv:1901.11504*, 2019b.

杨柳.微调提取式摘要的 BERT。*arXiv preprint arXiv:1903.10318*, 2019.

刘寅涵、Myle Ott、Naman Goyal、杜京飞、Mandar Joshi、陈丹琪、Omer Levy、Mike Lewis、Luke Zettlemoyer 和 Veselin Stoyanov。RoBERTa： A robustly optimized BERT pretraining approach. *ArXiv preprint arXiv:1907.11692*, 2019c.

Lajanugen Logeswaran 和 Honglak Lee.学习句子表征的高效框架。*ArXiv preprint arXiv:1803.02893*, 2018.

Dhruv Mahajan、Ross Girshick、Vignesh Ramanathan、Kaiming He、Manohar Paluri、Yixuan Li、Ashwin Bharambe 和 Laurens van der Maaten。探索弱监督预训练的极限。*欧洲计算机视觉会议（ECCV）论文集*，2018 年。

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher.自然语言十项全能：*ArXiv preprint arXiv:1806.08730*, 2018.

Tomas Mikolov、Kai Chen、Greg Corrado 和 Jeffrey Dean.向量空间中单词表征的高效估计。*arXiv preprint arXiv:1301.3781*, 2013a。

Tomas Mikolov、Ilya Sutskever、Kai Chen、Greg S. Corrado 和 Jeff Dean。单词和短语的分布式表征及其构成性。*神经信息处理系统进展*，2013b。

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, and Bing Xiang.使用序列到序列 RNN 及其他方法进行抽象文本总结》，*arXiv 预印本 arXiv:1602.06023*, 2016.

Maxime Oquab、Leon Bottou、Ivan Laptev 和 Josef Sivic。使用卷积神经网络学习和传输中层图像表征。*IEEE 计算机视觉与模式识别会议论文集*，2014 年。

Kishore Papineni、Salim Roukos、Todd Ward 和 Wei-Jing Zhu。BLEU：机器翻译自动评估方法。*计算语言学协会第 40 届年会论文集*。计算语言学协会，2002 年。

Romain Paulus, Caiming Xiong, and Richard Socher.抽象概括的深度强化模型》，*arXiv preprint arXiv:1705.04304*, 2017.

Jeffrey Pennington、Richard Socher 和 Christopher Manning。GloVe：用于单词表示的全局向量。*2014年自然语言处理实证方法会议（EMNLP）论文集*，2014年。

Matthew Peters, Sebastian Ruder, and Noah A. Smith.调整还是不调整？适应不同任务的预训练表征》，*arXiv preprint arXiv:1903.05987*, 2019.

Matthew E. Peters、Mark Neumann、Mohit Iyyer、Matt Gardner、Christopher Clark、Kenton Lee 和 Luke Zettlemoyer。深度语境化单词表征。*arXiv preprint arXiv:1802.05365*, 2018.

Jason Phang, Thibault Févry, and Samuel R. Bowman.STILT 上的句子编码器：*ArXiv preprint arXiv:1811.01088*, 2018.

Mohammad Taher Pilehvar 和 Jose Camacho-Collados.WIC：用于评估上下文敏感表征的 10,000 对示例。*arXiv 预印本 arXiv:1808.09121*, 2018。

马特-波斯特呼吁明确报告 BLEU 分数。*ArXiv 预印本 arXiv:1804.08771*, 2018。

Alec Radford、Karthik Narasimhan、Tim Salimans 和 Ilya Sutskever。通过生成预训练提高语言理解能力，2018.

Alec Radford、Jeffrey Wu、Rewon Child、David Luan、Dario Amodei 和 Ilya Sutskever。
语言模型是无监督的多任务学习器，2019 年。

Altaf Rahman 和 Vincent Ng.解决定语代词的复杂情况：Winograd 模式挑战。*2012 年自然语言处理和计算自然语言学习经验方法联合会议论文集*》。计算语言学协会，2012 年。

Pranav Rajpurkar、Jian Zhang、Konstantin Lopyrev 和 Percy Liang。小队：用于文本机器理解的 10 万多个问题。*arXiv preprint arXiv:1606.05250*, 2016.

Prajit Ramachandran, Peter J. Liu, and Quoc V. Le.序列到序列学习的无监督预训练》，*arXiv preprint arXiv:1611.02683*, 2016.

Alex Ratner、Braden Hancock、Jared Dunnmon、Roger Goldman 和 Christopher Ré。

Snorkel MeTaL：多任务学习的弱监督。*第二届端到端机器学习数据管理研讨会论文集*》，2018 年。

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon.可信替代方案的选择：常识因果推理的评估。*2011年AAAI春季系列研讨会*，2011年。

Sebastian Ruder.深度神经网络中的多任务学习概述》，*arXiv preprint arXiv:1706.05098*, 2017.

塞巴斯蒂安-鲁德*自然语言处理的神经迁移学习*。博士论文，NUI Galway，2019 年。

Sebastian Ruder、Matthew E. Peters、Swabha Swayamdipta 和 Thomas Wolf。自然语言处理中的迁移学习。*计算语言学协会北美分会 2019 年会议论文集：教程》*，第 15-18 页，2019 年。

Olga Russakovsky、Jia Deng、Hao Su、Jonathan Krause、Sanjeev Satheesh、Sean Ma、Zhiheng Huang、Andrej Karpathy、Aditya Khosla、Michael Bernstein 等：ImageNet 大规模视觉识别挑战。*国际计算机视觉杂志*，2015 年。

Victor Sanh、Lysandre Debut、Julien Chaumond 和 Thomas Wolf.DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Abigail See、Peter J. Liu 和 Christopher D. Manning。直奔主题：用指针生成器网络进行总结。*arXiv preprint arXiv:1704.04368*, 2017.

Rico Sennrich、Barry Haddow 和 Alexandra Birch.用子词单元进行罕见词的神经机器翻译》，*arXiv preprint arXiv:1508.07909*, 2015.

Christopher J Shallue, Jaehoon Lee, Joe Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E. Dahl.测量数据并行性对神经网络训练的影响》，*arXiv preprint arXiv:1811.03600*, 2018.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani.使用相对位置表征的自我注意力》，*arXiv preprint arXiv:1803.02155*, 2018.

诺姆-沙泽尔和米切尔-斯特恩。Adafactor：具有亚线性内存成本的自适应学习率。*arXiv preprint arXiv:1804.04235*, 2018.

Noam Shazeer、Azalia Mirhoseini、Krzysztof Maziarz、Andy Davis、Quoc Le、Geoffrey Hinton 和 Jeff Dean。超大型神经网络：稀疏门控的专家混合物层。*ArXiv 预印本 arXiv:1701.06538*, 2017.

Noam Shazeer、程幼龙、Niki Parmar、Dustin Tran、Ashish Vaswani、Penporn Koanantakool、Peter Hawkins、HyoukJoong Lee、Mingsheng Hong、Cliff Young、Ryan Sepassi 和 Blake Hechtman。Mesh-tensorflow：超级计算机深度学习。*神经信息处理系统进展》*，2018 年。

Jason R. Smith、 Herve Saint-Amand、 Magdalena Plamada、 Philipp Koehn、 Chris Callison-Burch、Adam Lopez。来自普通抓取的廉价网络规模并行文本。*第 51 届计算语言学协会年会论文集*，2013 年。

Richard Socher、 Alex Perelygin、 Jean Wu、 Jason Chuang、 Christopher D. Manning、 Andrew Ng 和 Christopher Potts。情感树库语义合成的递归深度模型。*2013 年自然语言处理实证方法会议论文集*，2013 年。

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu.MASS: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*, 2019.

Nitish Srivastava、Geoffrey Hinton、Alex Krizhevsky、Ilya Sutskever 和 Ruslan Salakhutdi-nov。Dropout：防止神经网络过度拟合的简单方法。*机器学习研究》杂志*，2014 年。

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J. Pal.通过大规模多任务学习学习通用分布式句子表征》，*arXiv preprint arXiv:1804.00079*, 2018.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le.用神经网络进行序列到序列学习。*神经信息处理系统进展*，2014 年。

Richard S. Sutton. 苦涩的一课。http://www.incompleteideas.net/IncIdeas/BitterLesson.html，2019.

Wilson L. Taylor."Cloze程序"：衡量可读性的新工具。*新闻通讯》*，1953 年。

Trieu H. Trinh 和 Quoc V. Le.Trinh 和 Quoc V. Le.常识推理的一种简单方法。*arXiv 预印本 arXiv:1806.02847*, 2018.

亚当-特里施勒、王彤、袁星迪、贾斯汀-哈里斯、亚历山德罗-索多尼、菲利普-巴赫曼和卡希尔-苏莱曼。NewsQA：A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*, 2016.

Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N. Gomez、Łukasz Kaiser 和 Illia Polosukhin。注意力就是你所需要的一切。*神经信息处理系统进展》*，2017 年。

Elena Voita、Rico Sennrich 和 Ivan Titov.转换器中表征的自下而上演化：以机器翻译和语言建模为目标的研究。*arXiv preprint arXiv:1909.01380*, 2019.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman.GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Alex Wang, Jan Hula, Patrick Xia, Raghavendra Pappagari, R . Thomas McCoy, Roma Patel, Najoung Kim, Ian Tenney, Yinghui Huang, Katherin Yu, et al. 你能告诉我如何穿过芝麻街吗？超越语言建模的句子级预训练。*第 57 届计算语言学协会年会论文集*，2019a

。

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman.SuperGLUE: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint arXiv:1905.00537*, 2019b.

王伟、毕斌、严明、吴晨、鲍祖毅、彭立伟和罗思。StructBERT：将语言结构纳入深度语言理解的预训练》，*arXiv preprint arXiv:1908.04577*, 2019c.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman.神经网络可接受性判断。*arXiv preprint arXiv:1805.12471*, 2018.

Adina Williams, Nikita Nangia, and Samuel R. Bowman.通过推理理解句子的广覆盖挑战语料库》，*arXiv preprint arXiv:1704.05426*, 2017.

Ronald J. Williams 和 David Zipser.持续运行完全递归神经网络的学习算法》。*神经计算*，1989 年。

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 谷歌的神经机器翻译系统：*ArXiv preprint arXiv:1609.08144*, 2016.

杨志林、戴子航、杨一鸣、Jaime Carbonell、Ruslan Salakhutdinov 和 Quoc V. Le。XLNet：用于语言理解的广义自回归预训练。*arXiv preprint arXiv:1906.08237*, 2019.

Jason Yosinski、Jeff Clune、Yoshua Bengio 和 Hod Lipson。深度神经网络中的特征有多大转移性？*神经信息处理系统进展*》，2014 年。

Adams Wei Yu、David Dohan、Minh-Thang Luong、Rui Zhao、Kai Chen、Mohammad Norouzi 和 Quoc V. Le。QAnet：结合局部卷积与全局自我关注的阅读理解。*arXiv preprint arXiv:1804.09541*, 2018.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi.抵御神经假新闻。*arXiv 预印本 arXiv:1905.12616*, 2019.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme.ReCoRD: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint arXiv:1810.12885*, 2018.

Chen Zhu、Yu Cheng、Zhe Gan、Siqi Sun、Thomas Goldstein 和 Jingjing Liu。Freelb：En- hanced adversarial training for language understanding. *ArXiv preprint arXiv:1909.11764*, 2019.

朱玉坤、Ryan Kiros、Rich Zemel、Ruslan Salakhutdinov、Raquel Urtasun、Antonio Torralba 和 Sanja Fidler。将书籍和电影对齐：通过观看电影和阅读书籍实现类似故事的视觉解释。2015年*IEEE计算机视觉国际会议论文集。*