Journal of Machine Learning Research 21 (2020) 1-54 Submitted 1/20; Revised 6/20; Published 6/20

用统一的文本到文本Transformer探索迁移学习的局限性

Colin Raffel* CRAFFEL@GMAIL.COM
Noam Shazeer* NOAM@GOOGLE.COM
Adam Roberts* ADAROB@GOOGLE.COM
Katherine Lee* KATHERINELEE@GOOGLE.COM
Sharan Narang SHARANNARANG@GOOGLE.COM
Michael Matena MMATENA@GOOGLE.COM
Yanqi Zhou YANQIZ@GOOGLE.COM
Wei Li MWEILI@GOOGLE.COM
Peter J. Liu PETERJLIU@GOOGLE.COM

Google, Mountain View, CA 94043, USA

Editor: Ivan Titov

Abstract

迁移学习是一种首先在数据丰富的任务上对模型进行预训练，然后在下游任务上进行微调的技术，已经成为自然语言处理(NLP)中的一种强大技术。迁移学习的有效性带来了方法、方法论和实践的多样性。本文通过引入一个统一的框架，将所有基于文本的语言问题转换为文本到文本的格式，探索了NLP的迁移学习技术的前景。在几十个语言理解任务中，对预训练目标、架构、未标记数据集、迁移方法和其他因素进行了比较。通过将我们探索的见解与规模和新的"巨大干净爬取的语料库"相结合，我们在许多基准上取得了最先进的结果，包括摘要，问答，文本分类等。为了促进NLP迁移学习的未来工作，发布了数据集、预训练模型和代码。[1]

Keywords: transfer learning, natural language processing, multi-task learning, attention-based models, deep learning

1. 简介

训练一个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。这可以被松散地视为开发通用知识，允许模型"理解"文本。这些知识可以是低级的(例如单词的拼写或含义)，也可以是高级的(例如大号太大了，装不下大多数背包)。在现代机器学习实践中，很少明确地提供这些知识;相反，它通常作为辅助任务的一部分来学习。例如，历史上常见的方法是使用词向量(Mikolov et al., 2013b,a; Pennington et al., 2014)将单词标识映射到连续表示，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，鼓励将共现词放置在连续空间的附近(Mikolov et al., 2013b)。

*. 同等贡献。每个作者的贡献描述可以在Appendix A上找到。来信craffel@gmail.com。
1. https://github.com/google-research/text-to-text-transfer-transformer

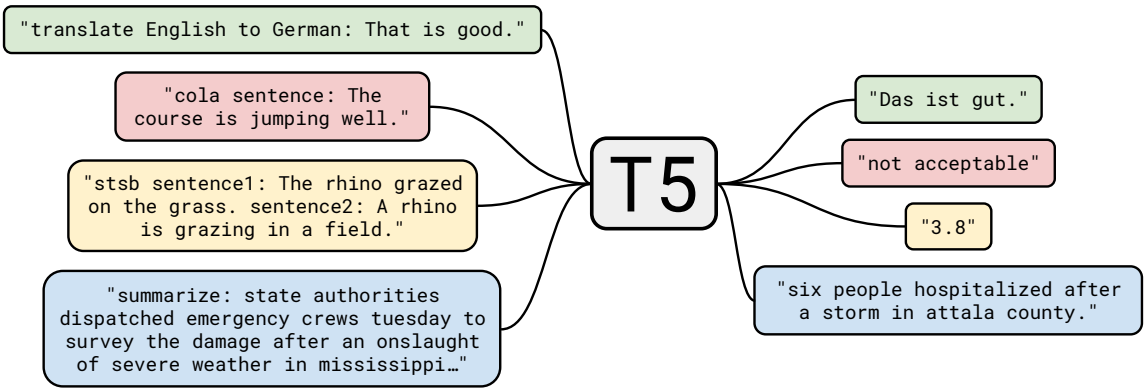

Figure 1: 文本到文本框架的图表。我们考虑的每个任务——包括翻译、问答和分类——都被视为输入我们的模型文本，并训练它生成一些目标文本。这允许我们在不同的任务集上使用相同的模型、损失函数、超参数等。它还为我们的实证调查中包括的方法提供了一个标准的测试平台。'T5 '指的是我们的模型，我们称之为' **文本 - to - 文本转移变压器** '。

最近，在数据丰富的任务上预训练整个模型变得越来越常见。理想情况下，这种预训练使模型开发通用能力和知识，然后可以转移到下游任务。在迁移学习到计算机视觉的应用中(Oquab et al., 2014; Jia et al., 2014; Huh et al., 2016; Yosinski et al., 2014)，预训练通常是通过在大型标记数据集(如ImageNet (Russakovsky et al., 2015; Deng et al., 2009))上进行监督学习来完成的。相比之下，NLP中用于迁移学习的现代技术通常使用无监督学习对无标记数据进行预训练。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 et al., 2018; Yang et al., 2019; Dong et al., 2019; Liu et al., 2019c; Lan et al., 2019)。除了经验上的优势之外，NLP的无监督预训练特别有吸引力，因为无标签的文本数据可以大量获得，这要归功于互联网——例如，Common Crawl项目[2]每月从网页中提取约20TB的文本数据。这对于神经网络来说是很自然的选择，因为神经网络已经被证明具有显著的可扩展性，即通常可以通过简单地在更大的数据集上训练更大的模型来实现更好的性能(Hestness et al., 2017; Shazeer et al., 2017; Jozefowicz et al., 2016; Mahajan et al., 2018; Radford et al., 2019; Shazeer et al., 2018; Huang et al., 2018b; Keskar et al., 2019a)。

这种协同作用导致了最近大量开发NLP迁移学习方法的工作，这产生了广泛的预训练目标(Howard and Ruder, 2018; Devlin et al., 2018; Yang et al., 2019; Dong et al., 2019)，未标记数据集(Yang et al., 2019; Liu et al., 2019c; Zellers et al., 2019)，基准(Wang et al., 2019b, 2018; Conneau and Kiela, 2018)，微调方法(Howard and Ruder, 2018; Houlsby et al., 2019; Peters et al., 2019)等。在这个新兴领域中，技术的快速进步和多样性，使得很难比较不同的算法，梳理新贡献的影响，以及理解现有迁移学习方法的空间。在对更严格理解的需求的推动下，利用统一的方法进行迁移学习，使我们能够系统地研究不同的方法，并推动该领域的当前极限。

我们工作的基本思想是将每个文本处理问题视为"文本到文本"问题，即将文本作为输入并产生新的文本作为输出。这种方法的灵感来自于之前的NLP任务统一框架，包括将所有文本问题转换为问答(McCann et al., 2018)，语言建模(Radford et al., 2019)或span提

2. http://commoncrawl.org

取Keskar et al. (2019b)任务。关键是，文本到文本框架允许我们直接将相同的模型、目标、训练程序和解码过程应用于我们考虑的每个任务。通过评估各种基于英语的NLP问题的性能，包括问答、文档摘要和情感分类，来利用这种灵活性。通过这种统一的方法，我们可以比较不同迁移学习目标、未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，超越以前的考虑，探索NLP迁移学习的局限性。

本文强调，我们的目标不是提出新的方法，而是为该领域所处的位置提供一个全面的视角。本文工作主要包括对现有技术的调查、探索和实证比较。通过扩大系统研究(训练模型到11亿参数)的见解，探索了当前方法的局限性，以在所考虑的许多任务中获得最先进的结果。为了在这种规模上进行实验，引入了"庞大的干净爬取语料库" (C4)，一个由数百gb从网络上抓取的干净英语文本组成的数据集。认识到迁移学习的主要效用是在数据稀缺的情况下利用预训练模型的可能性，本文发布了代码、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，我们将讨论基本模型及其实现，将每个文本处理问题制定为文本到文本任务的过程，以及我们考虑的一套任务。在Section 3中，我们提供了一套探索NLP迁移学习领域的大型实验。在本节的最后(Section 3.7)，结合了系统研究的见解，以在各种基准上获得最先进的结果。最后，我们对我们的研究结果进行了总结，并在Section 4上对未来进行了展望。

2. 设置

在介绍大规模实证研究的结果之前，回顾了理解结果所需的必要背景主题，包括Transformer模型架构和所评估的下游任务。本文还介绍了将每个问题视为文本到文本任务的方法，并描述了我们的“巨大的干净爬取语料库”(C4)，这是我们创建的常见的基于爬取的数据集，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**转换器**”(T5)。

2.1 模型

关于NLP迁移学习的早期结果利用了循环神经网络(Peters et al., 2018; Howard and Ruder, 2018)，但最近更常见的是使用基于“Transformer”架构的模型(Vaswani et al., 2017)。Transformer最初被证明对机器翻译是有效的，但它随后被用于各种NLP设置(Radford et al., 2018; Devlin et al., 2018; McCann et al., 2018; Yu et al., 2018)。由于Transformer的日益普及，本文研究的所有模型都基于Transformer架构。除了下面提到的细节和我们在Section 3.2中探索的变体之外，我们没有明显偏离最初提出的这种架构。我们没有提供该模型的全面定义，感兴趣的读者可以参考原始论文(Vaswani et al., 2017)或后续教程[3,4]以获得更详细的介绍。

Transformer的主要组成部分是self-attention (Cheng et al., 2016)。Self-attention是attention的一种变体(Graves, 2013; Bahdanau et al., 2015)，它通过将每个元素替换为序列其余部分的加权平均值来处理序列。最初的Transformer由编码器-解码器架构组成，用于序列到序列(Sutskever et al., 2014; Kalchbrenner et al., 2014)任务。最近，使用由单个Transformer层堆栈组成的模型也变得很常见，其中使用各种形式的自注意力来生成适合语言建模(Radford et al., 2018; Al-Rfou et al., 2019)或分类和span预测任务(Devlin et al., 2018; Yang et al., 2019)的架构。我们在Section 3.2中以经验探索了这些架构变体。

总的来说，我们的编码器-解码器Transformer实现紧跟其最初提出的形式(Vaswani et al., 2017)。首先，将输入标记序列映射到嵌入序列，然后将其传递到编码器。编码器由一

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

堆“块“组成，每个块包含两个子组件:self-attention层，后面是一个小的前馈网络。层归一化(Ba et al., 2016)应用于每个子组件的输入。我们使用层归一化的简化版本，其中激活仅被重新缩放，而不应用加性偏差。在层归一化之后，残差跳过连接(He et al., 2016)将每个子组件的输入添加到其输出。Dropout (Srivastava et al., 2014)应用于前馈网络内部，在跳跃连接上，在注意力权重上，以及在整个堆栈的输入和输出上。解码器在结构上类似于编码器，除了它在每个自注意力层之后包含一个标准的注意力机制，该机制关注编码器的输出。解码器中的自注意力机制还使用了一种形式的自回归或因果自注意力，这只允许模型关注过去的输出。最终解码器块的输出被馈送到具有softmax输出的密集层，其权重与输入嵌入矩阵共享。Transformer中的所有注意力机制都被分割为独立的“头”，其输出在进一步处理之前被连接起来。

由于自注意力是与顺序无关的(即它是对集合的操作)，通常为Transformer提供明确的位置信号。虽然原始的Transformer使用了正弦位置信号或学习位置嵌入，但最近使用相对位置嵌入变得更加常见(Shaw et al., 2018; Huang et al., 2018a)。相对位置嵌入不是对每个位置使用固定的嵌入，而是根据自注意力机制中比较的“key ‘和“query ‘之间的偏移量产生不同的学习嵌入。我们使用一种简化形式的位置嵌入，其中每个“嵌入”只是一个标量，添加到用于计算注意力权重的相应logit中。为了效率，我们还在模型的所有层中共享位置嵌入参数，尽管在给定的层中每个注意力头使用不同的学习位置嵌入。通常，学习固定数量的嵌入，每个嵌入对应于可能的键查询偏移范围。本文对所有模型使用32嵌入，其范围以对数方式增加到偏移量128，超过该偏移量，将所有相对位置分配给相同的嵌入。请注意，给定的层对128标记以外的相对位置不敏感，但后续层可以通过结合以前层的局部信息来构建对更大偏移量的敏感性。总而言之，我们的模型大致等同于Vaswani et al. (2017)提出的原始Transformer，除了删除层范数偏差，将层归一化放置在残差路径之外，并使用不同的位置嵌入方案。由于这些架构变化与迁移学习实证调查中考虑的实验因素正交，将其影响的消融留给未来的工作。

作为我们研究的一部分，我们实验了这些模型的可扩展性，即当它们具有更多参数或层时，它们的性能如何变化。训练大型模型并非易事，因为它们可能不适用于单个机器，并需要大量的计算。因此，我们使用模型和数据并行的组合，并在云TPU Pods的“切片”上训练模型。[5] TPU pods是多机架ML超级计算机，包含1,024 TPU v3芯片，通过高速2D mesh互连与支持的CPU主机连接。我们利用Mesh TensorFlow库(Shazeer et al., 2018)简化模型并行性和数据并行性的实现(Krizhevsky, 2014)。

2.2 巨大干净的爬行语料库

之前关于NLP迁移学习的大部分工作都利用大量无标签数据集进行无监督学习。本文对测量这种无标签数据的质量、特征和大小的影响感兴趣。为了生成满足需求的数据集，我们利用Common Crawl作为从网络上抓取的文本来源。Common Crawl之前被用作NLP的文本数据来源，例如训练n元语言模型(Buck et al., 2014)，作为常识推理的训练数据(Trinh and Le, 2018)，为机器翻译挖掘平行文本(Smith et al., 2013)，作为预训练数据集(Grave et al., 2018; Zellers et al., 2019; Liu et al., 2019c)，甚至简单地用作测试优化器的巨型文本语料库(Anil et al., 2019)。

Common Crawl是一个公开可用的web存档，它通过从抓取的HTML文件中删除标记和其他非文本内容来提供“web提取文本”。这个过程每个月产生大约20TB的抓取文本数据。不幸的是，大部分生成的文本不是自然语言。相反，它主要是由杂乱无章的或样板式的文本组成，如菜单、错误消息或重复的文本。此外，大量被抓取的文本包含的内容对我们考

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

虑的任何任务都不太可能有帮助(攻击性语言、占位符文本、源代码等)。为了解决这些问题，我们使用以下启发式方法来清理常见的爬虫网页提取的文本:

- 我们只保留了以标点符号结尾的行(例如，句号、感叹号、问号或结束引号)。
- 我们丢弃了所有少于5个句子的页面，只保留了至少包含3个单词的行。
- 我们删除了包含“肮脏、顽皮、淫秽或其他不良词汇”列表中的任何单词的页面。[6]
- 许多被抓取的页面包含Javascript应该被启用的警告，因此我们删除了所有包含Javascript单词的行。
- 有些页面有占位符“lorem ipsum”文本;我们删除了所有出现“lorem ipsum”字样的页面。
- 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中(例如Javascript，在web上广泛使用)，但没有出现在自然文本中，因此我们删除了所有包含大括号的页面。
- 为了消除数据集的重复，我们丢弃了数据集中出现不止一次的所有三句话，只留下一个。

此外，由于我们的大多数下游任务都集中在英语文本上，我们使用`langdetect`[7]来过滤任何非英语页面(概率至少为0.99)。我们的启发式方法受到过去使用Common Crawl作为NLP数据来源的工作的启发: 例如，Grave et al. (2018)也使用自动语言检测器过滤文本并丢弃短行，Smith et al. (2013); Grave et al. (2018)都执行行级重复删除。然而，我们选择创建一个新的数据集，因为之前的数据集使用了一组更有限的过滤启发式方法，不公开可用，和/或在范围上不同(例如限于新闻数据(Zellers et al., 2019; Liu et al., 2019c)，仅包含知识共享内容(Habernal et al., 2016)，或专注于机器翻译的并行训练数据(Smith et al., 2013))。

为了组装我们的基础数据集，我们下载了2019年4月的网络提取文本，并应用了上述过滤。这产生了一个文本集合，不仅比用于预训练的大多数数据集(约750 GB)大几个数量级，而且包含相当干净和自然的英文文本。我们将此数据集命名为“**巨大的清洁爬取语料库**”(简称C4)，并将其作为TensorFlow数据集的一部分发布。[8] 我们在Section 3.4中考虑了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的语言学习能力。本文研究了不同基准集上的下游性能，包括机器翻译、问答、抽象摘要和文本分类。测量了GLUE和SuperGLUE文本分类元基准上的性能;CNN/Daily Mail摘要;小队问答;并将英语翻译成德语、法语和罗马尼亚语。所有数据都来自TensorFlow数据集。[9]

GLUE (Wang et al., 2018)和SuperGLUE (Wang et al., 2019b)都包含一系列文本分类任务，旨在测试一般的语言理解能力:

6. https://github.com/LDNOOBW/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words
7. https://pypi.org/project/langdetect/
8. https://www.tensorflow.org/datasets/catalog/c4
9. https://www.tensorflow.org/datasets

- 句子接受度判断(CoLA (Warstadt et al., 2018))
- 情感分析(SST-2 (Socher et al., 2013))
- 复述/句子相似度(MRPC (Dolan and Brockett, 2005), STS-B (Cer et al., 2017), QQP (Iyer et al., 2017))
- 自然语言推理(MNLI (Williams et al., 2017), QNLI (Rajpurkar et al., 2016), RTE (Dagan et al., 2005), CB (De Marneff et al., 2019))
- 共指消解(WNLI和WSC (Levesque et al., 2012))
- 句子补全(COPA (Roemmele et al., 2011))
- 词义消歧(WIC (Pilehvar and Camacho-Collados, 2018))
- 问答(MultiRC (Khashabi et al., 2018), ReCoRD (Zhang et al., 2018), BoolQ (Clark et al., 2019))

我们使用GLUE和SuperGLUE基准分布的数据集。为简单起见，在进行微调时，我们通过连接所有组成数据集将GLUE基准中的所有任务(SuperGLUE也类似)视为单个任务。正如Kocijan et al. (2019)所建议的，我们还在组合的SuperGLUE任务中包括了确定代词归结(DPR)数据集(Rahman and Ng, 2012)。

CNN/Daily Mail (Hermann et al., 2015)数据集是作为问答任务引入的，但Nallapati et al. (2016)将其改编为文本摘要;我们使用来自See et al. (2017)的非匿名版本作为抽象的摘要任务。SQuAD (Rajpurkar et al., 2016)是一个常见的问答基准。在实验中，模型被灌输问题及其上下文，并被要求逐个词地生成答案。对于WMT英语到德语，我们使用与(Vaswani et al., 2017)(即新闻评论v13, Common Crawl, Europarl v7)和`newstest2013`相同的训练数据作为验证集(Bojar et al., 2014)。对于英语到法语，我们使用2015年和`newstest2014`的标准训练数据作为验证集(Bojar et al., 2015)。对于英语到罗马尼亚语(这是一个标准的低资源机器翻译基准)，我们使用来自WMT 2016 (Bojar et al., 2016)的训练和验证集。请注意，我们只对英语数据进行预训练，因此为了学习翻译给定的模型，需要学习用新语言生成文本。

2.4 输入输出格式

为了在上述不同的任务集上训练单个模型，我们将考虑的所有任务转换为“文本到文本”格式——也就是说，在任务中，模型被提供一些文本作为上下文或条件，然后被要求产生一些输出文本。该框架为预训练和微调提供了一致的训练目标。具体来说，无论任务如何，该模型都以最大似然目标(使用“教师强迫”(Williams and Zipser, 1989))进行训练。为了指定模型应该执行的任务，我们在将原始输入序列提供给模型之前，在原始输入序列中添加一个特定于任务的(文本)前缀。

例如，让模型翻译句子’ ’那就好。“从英语到德语，模型将被输入序列‘将英语翻译成德语:这很好。“并将被训练输出‘Das ist gut’。” 对于文本分类任务，该模型只是预测与目标标签相对应的单个单词。例如，在MNLI基准(Williams et al., 2017)上，目标是预测前提是否意味着(“蕴含”)、矛盾(“矛盾”)或两者都不是(“中性”)假设。通过我们的预处理，输入序列变成了“mnli前提:我讨厌鸽子。假设:我对鸽子充满敌意。“与对应的目标词‘蕴涵’”。请注意，如果我们的模型在文本分类任务上输出的文本与任何可能的标签都不对应，

则会出现一个问题(例如，如果模型输出‘汉堡包‘，而任务的唯一可能标签是‘蕴含‘、‘中性‘或‘矛盾‘)。在这种情况下，我们总是认为模型的输出是错误的，尽管我们从未在任何训练过的模型中观察到这种行为。注意，为给定任务选择的文本前缀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广泛的实验。Figure 1中显示了我们的文本到文本框架的图表，其中包含一些输入/输出示例。我们为我们在Appendix D中研究的每个任务提供了预处理输入的完整示例。

我们的文本到文本框架遵循之前的工作，将多个NLP任务转换为通用格式：McCann et al. (2018)提出“自然语言十项全能”，这是一个基准，为一套十个NLP任务使用一致的问答格式。自然语言Decathlon还规定所有模型必须是多任务的，即能够同时处理所有任务。允许在每个单独的任务上单独微调模型，并使用简短的任务前缀，而不是显式的问答格式。Radford et al. (2019)通过将一些输入作为前缀输入到模型，然后对输出进行自回归采样，评估语言模型的零样本学习能力。例如，自动摘要是通过输入一个文档，然后输入文本” TL;DR: " (” too long, didn’t read’的缩写，一个常见的缩写)，然后通过自回归解码预测摘要。主要考虑在用单独的解码器生成输出之前用编码器显式处理输入的模型，并专注于迁移学习而不是零样本学习。最后，Keskar et al. (2019b)将许多NLP任务统一为“跨度提取”，将可能的输出选择对应的文本附加到输入中，并训练模型以提取正确选择对应的输入跨度。相比之下，该框架还允许机器翻译和抽象摘要等生成任务，其中不可能列举所有可能的输出选择。

除了STS-B之外，我们能够直接将我们考虑的所有任务转换为文本到文本的格式，STS-B是一个回归任务，其目标是预测1和5之间的相似性得分。我们发现这些分数中的most以0.2的增量进行注释，因此我们简单地将任何分数舍入到最接近的增量0.2，并将结果转换为数字的字面量字符串表示(例如，浮点值2.57将映射到字符串” 2.6 ")。在测试时，如果模型输出一个与1和5之间的数字对应的字符串，我们将其转换为浮点值;否则，我们将模型的预测视为不正确。这有效地将STS-B回归问题重新转换为一个21类分类问题。

此外，还将Winograd任务(来自GLUE的WNLI，来自SuperGLUE的WSC，以及添加到SuperGLUE的DPR数据集)转换为更简单的格式，更适合文本到文本框架。Winograd任务中的例子包含一段文本，其中包含一个模棱两可的代词，可以指代文章中的多个名词短语。例如，该通道可能是“市议员拒绝向示威者发放许可证，因为他们害怕暴力。，其中包含了模棱两可的代词“他们”，可能指“市议员”或“示威者”。通过突出文本段落中模棱两可的代词，并要求模型预测其所指的名词，将WNLI、WSC和DPR任务转换为文本到文本问题。上面提到的例子将被转换为输入“市议员拒绝示威者许可证，因为*他们*害怕暴力。”“并且该模型将被训练以预测目标文本‘市议员’”。

对于WSC，示例包含文章、歧义代词、候选名词和反映候选词是否匹配代词的真/假标签(忽略任何冠词)。我们只在具有“True”标签的示例上进行训练，因为我们不知道具有“False”标签的示例的正确名词目标。为了进行评估，如果模型输出中的单词是候选名词短语中的单词的子集，我们分配一个‘True ‘标签(反之亦然)，否则分配一个‘False ‘标签。这删除了大约一半的WSC训练集，但DPR数据集增加了关于1,000代词解析的示例。来自DPR的示例使用了正确的指代名词进行了注释，从而可以轻松地以上述格式使用此数据集。

WNLI训练集和验证集与WSC训练集有显著的重叠。为了避免将验证示例泄漏到我们的训练数据中(Section 3.5.2的多任务实验中的一个特殊问题)，因此我们从不在WNLI上进行训练，也从不报告WNLI验证集的结果。忽略WNLI验证集上的结果是标准做法(Devlin et al., 2018)，因为它相对于训练集是“对抗性”的，即验证示例都是具有相反标签的训练示例的轻微扰动版本。因此，每当我们报告验证集时，我们都不将WNLI包括在平均胶水分

数中(除Section 3.7以外的所有部分，其结果在测试集上显示)。将WNLI示例转换为上述的“参照名词预测”变体稍微复杂一些;我们在Appendix B中描述了这个过程。

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、未标记的数据集等。在本节中，我们对这些技术进行实证调查，希望梳理它们的贡献和意义。结合所获得的见解，在所考虑的许多任务中达到最先进的水平。由于NLP的迁移学习是一个快速发展的研究领域，在我们的实证研究中覆盖所有可能的技术或想法是不可行的。对于更广泛的文献回顾，我们推荐Ruder et al. (2019)的最新调查。

我们通过采取合理的基线(在Section 3.1中描述)并每次改变设置的一个方面来系统地研究这些贡献。例如，在Section 3.3中，我们测量了不同的无监督目标的性能，同时保持实验管道的其余部分不变。这种“坐标上升”方法可能会错过二阶效应(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好)，但对我们研究中的所有因素进行组合探索将是非常昂贵的。在未来的工作中，我们希望能够更深入地考虑所研究方法的组合。

我们的目标是在不同的任务集上比较各种不同的方法，同时保持尽可能多的因素固定。为了满足这一目标，在某些情况下，我们并不完全复制现有的方法。例如，像BERT (Devlin et al., 2018)这样的“仅编码器”模型被设计为对每个输入标记产生单个预测或对整个输入序列产生单个预测。这使得它们适用于分类或跨度预测任务，但不适用于翻译或抽象摘要等生成任务。因此，我们考虑的模型架构中没有一个与BERT相同，或者只包含编码器结构。相反，我们测试了在精神上类似的方法——例如，我们考虑了一个与Section 3.3中BERT的“掩码语言建模”目标类似的目标，我们考虑了一个与Section 3.2中BERT在文本分类任务上的行为类似的模型架构。

在下一小节中概述了我们的基线实验设置后，对模型架构(Section 3.2)、无监督目标(Section 3.3)、预训练数据集(Section 3.4)、迁移方法(Section 3.5)和扩展(Section 3.6)进行了实证比较。在本节的高潮，将研究的见解与规模相结合，在我们考虑的许多任务中获得最先进的结果(Section 3.7)。

3.1 基线

我们的基线目标是反映典型的现代实践。使用简单的去噪目标预训练一个标准Transformer(在Section 2.1中描述)，然后分别对每个下游任务进行微调。我们将在接下来的小节中描述这个实验设置的细节。

3.1.1 模型

对于我们的模型，我们使用由Vaswani et al. (2017)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使用仅由单个“栈”组成的Transformer架构(例如，用于语言建模(Radford et al., 2018; Dong et al., 2019)或分类和跨度预测(Devlin et al., 2018; Yang et al., 2019))，但我们发现使用标准的编码器-解码器结构在生成和分类任务上都取得了良好的结果。我们在Section 3.2中探索了不同模型架构的性能。

我们的基线模型被设计成编码器和解码器的大小和配置都类似于“BERT $_{\text{BASE}}$ ” (Devlin et al., 2018)堆栈。具体来说，编码器和解码器都由12块组成(每个块包括自注意力，可选的编码器-解码器注意力和前馈网络)。每个块中的前馈网络由输出维度为$d_{\text{ff}} = 3072$的密集层、ReLU非线性和另一个密集层组成。所有注意力机制的“键”和“值”矩阵的内部维度

为$d_{\text{kv}} = 64$，所有注意力机制都有12头。所有其他子层和嵌入的维度都是$d_{\text{model}} = 768$。总的来说，这导致了一个大约有220百万参数的模型。这大约是BERT $_{\text{BASE}}$参数数量的两倍，因为我们的基线模型包含两层堆栈而不是一层。对于正则化，我们在模型中应用dropout的地方使用dropout概率0.1。

3.1.2 培训

如Section 2.4所述，所有任务都被定义为文本到文本任务。这允许我们总是使用标准的最大似然进行训练，即使用teacher强迫(Williams and Zipser, 1989)和交叉熵损失。为了优化，我们使用AdaFactor (Shazeer and Stern, 2018)。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$2^{19} = 524{,}288$步骤的预训练。我们使用的最大序列长度为512，批量大小为128序列。只要有可能，我们将多个序列“打包”到批处理的每个条目[10]中，以便我们的批处理大约包含$2^{16} = 65{,}536$标记。总的来说，此批大小和步骤数对应于$2^{35} \approx 34$B token的预训练。这比BERT (Devlin et al., 2018)和RoBERTa (Liu et al., 2019c)要少得多，前者使用的token大致为137B，后者使用的token大致为2.2T。仅使用2^{35}令牌可以产生合理的计算预算，同时仍然提供足够数量的预训练以获得可接受的性能。我们考虑了预训练对Sections 3.6 and 3.7中更多步骤的影响。请注意，2^{35} token只覆盖整个C4数据集的一部分，因此我们在预训练期间从不重复任何数据。

在预训练期间，我们使用一个“逆平方根”学习率计划:$1/\sqrt{\max(n, k)}$其中n是当前训练迭代，k是热身步骤的数量(在我们所有的实验中设置为10^4)。这为第一个10^4步骤设置了一个恒定的学习率0.01，然后以指数方式衰减学习率，直到预训练结束。我们还尝试使用三角形学习率(Howard and Ruder, 2018)，它产生了稍好的结果，但需要提前知道训练步骤的总数量。由于我们将在一些实验中改变训练步骤的数量，因此我们选择更通用的逆平方根调度。

该模型对所有任务的$2^{18} = 262{,}144$步骤进行了微调。选择这个值是为了权衡高资源任务(即具有大数据集的任务)和低资源任务(较小的数据集)之间的关系，前者受益于额外的微调，后者会很快过拟合。在微调期间，我们继续使用128长度- 512序列的批次(即每个批次2^{16} token)。我们在微调时使用0.001的恒定学习率。我们在每个5,000步骤中保存一个检查点，并报告与最高验证性能对应的模型检查点的结果。对于在多个任务上进行微调的模型，为每个任务独立选择最佳检查点。对于除Section 3.7中的实验之外的所有实验，我们报告了验证集的结果，以避免在测试集上执行模型选择。

3.1.3 词汇

我们使用SentencePiece (Kudo and Richardson, 2018)将文本编码为WordPiece token (Sennrich et al., 2015; Kudo, 2018)。对于所有实验，我们使用32,000 wordpieces词汇表。由于我们最终对英语到德语、法语和罗马尼亚语的翻译进行了微调，我们还要求我们的词汇涵盖这些非英语语言。为了解决这个问题，我们从C4中使用的常见爬取页面中将其分类为德语、法语和罗马尼亚语。然后，我们在10部分英语C4数据和1部分数据分类为德语，法语或罗马尼亚语的混合上训练我们的句子模型。这个词汇表在我们模型的输入和输出中共享。请注意，我们的词汇表使我们的模型只能处理一组预定的、固定的语言。

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

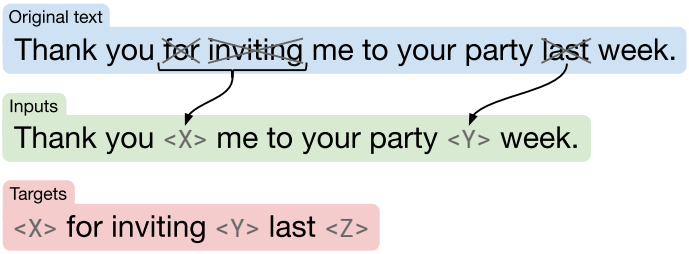

Figure 2: 我们在基线模型中使用的目标示意图。在这个例子中，我们处理了句子“Thank you for invite me to your party last week”。“for”、“invite”和“last”(标记为×)是随机选择用于表示腐败的单词。每个连续的损坏标记区间都被替换为一个哨兵标记(如<X><X><Y>和<Y>所示)，该标记在本例中是唯一的。由于“for "和“ invite "连续发生，它们被单个哨兵<X>< X>替换。然后，输出序列由删除的span组成，由用于在输入中替换它们的哨兵标记加上最后的哨兵标记<Z><Z>分隔。

3.1.4 无监督目标

利用未标记的数据对模型进行预训练，需要一个不需要标记的目标，但(松散地说)教给模型可泛化的知识，这些知识在下游任务中很有用。将预训练和微调所有模型参数的迁移学习范式应用于NLP问题的初步工作，使用了预训练的因果语言建模目标(Dai and Le, 2015; Peters et al., 2018; Radford et al., 2018; Howard and Ruder, 2018)。然而，最近的研究表明，“去噪”目标(Devlin et al., 2018; Taylor, 1953)(也称为“掩码语言建模”)产生了更好的性能，因此它们很快成为标准。在去噪目标中，模型被训练成预测输入中缺失或损坏的标记。受BERT的“掩码语言建模”目标和“单词dropout”正则化技术(Bowman et al., 2015)的启发，我们设计了一个目标，随机采样然后在输入序列中删除15%个标记。所有连续的被丢弃的标记片段都被一个哨兵标记所取代。每个哨兵令牌都分配了一个令牌ID，该ID与序列唯一。哨兵id是添加到我们词汇表中的特殊标记，不对应于任何单词。然后，目标对应于所有删除的标记范围，由输入序列中使用的相同的哨兵标记加上标记目标序列结束的最后一个哨兵标记分隔。选择屏蔽连续的token范围，只预测退出的token，以减少预训练的计算成本。对Section 3.3中的预训练目标进行了彻底的调查。Figure 2中显示了应用此目标所产生的转换示例。我们根据经验将此目标与Section 3.3中的许多其他变体进行了比较。

3.1.5 基准性能

在本节中，我们将展示使用上述基线实验过程的结果，以了解在我们的下游任务套件上预期的性能。理想情况下，我们将研究中的每个实验重复多次，以获得结果的置信区间。不幸的是，由于我们运行大量的实验，这将是非常昂贵的。作为一个更便宜的替代方案，我们从头开始训练我们的基线模型10次(即使用不同的随机初始化和数据集洗选)，并假设基础模型的这些运行的方差也适用于每个实验变体。我们不期望我们所做的大多数更改会对运行间方差产生显著影响，因此这应该可以合理地表明不同更改的重要性。另外，我们还测量了在没有预训练的情况下，在所有下游任务上为2^{18}步骤(与我们用于微调的数量相同)训练模型的性能。这让我们了解在基线设置中预训练对我们的模型有多大好处。

当在正文中报告结果时，我们只报告所有基准的分数子集，以节省空间并便于解释。对于GLUE和SuperGLUE，我们在‘GLUE ‘和‘SGLUE ‘的标题下报告所有子任

	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline average	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Baseline standard deviation	0.235	0.065	0.343	0.416	0.112	0.090	0.108
No pre-training	66.22	17.60	50.31	53.04	25.86	**39.77**	24.04

Table 1: 我们的基线模型和训练程序实现的分数的平均和标准偏差。为了进行比较，还报告了在每个任务上进行从头训练(即没有任何预训练)时的性能，用于微调基线模型的步数相同。该表中的所有分数(以及本文中的每个表，Table 14除外)都报告了每个数据集的验证集。

务的平均分数(根据官方基准规定)。对于所有翻译任务，我们报告了带有"exp"平滑和"intl"标记化的BLEU分数(Papineni et al., 2002)，该分数由SacreBLEU v1.3.0 (Post, 2018)提供。我们将WMT英语到德语、英语到法语和英语到罗马尼亚语的分数分别称为EnDe、EnFr和EnRo。对于CNN/Daily Mail，我们发现模型在ROUGE-1-F、ROUGE-2-F和ROUGE-L-F指标(Lin, 2004)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。类似地，对于SQuAD，我们发现"精确匹配"和"F1"分数的表现高度相关，因此我们仅报告"精确匹配"分数。我们在Table 16, Appendix E中提供了所有实验中每个任务的得分。

我们的结果表都进行了格式化，以便每行对应于一个特定的实验配置，其中列给出了每个基准的分数。我们将在大多数表中包含基准配置的平均性能。在基线配置出现的地方，我们将用★标记它(如Table 1的第一行所示)。我们还将**加粗**给定实验中与最大值(最佳)的两个标准差内的任何分数。

我们的基线结果显示在Table 1。总的来说，我们的结果与现有的类似规模的模型相当。例如，BERT $_{\text{BASE}}$在SQuAD上的精确分数为80.8，在MNLI-matched上的精确分数为84.4，而我们分别实现了80.88和84.24(参见Table 16)。请注意，我们不能直接将我们的基线与BERT $_{\text{BASE}}$进行比较，因为我们的模型是一个编码器-解码器模型，并且预训练了大约¼的步骤。意料之中的是，预训练在几乎所有基准上都提供了显著的增益。唯一的例外是WMT英语到法语，这是一个足够大的数据集，预训练的收益往往是边际的。在实验中加入了这项任务，以测试高资源情况下迁移学习的行为。由于我们通过选择性能最好的检查点来进行早期停止，我们的基线和"没有预训练"之间的巨大差异强调了预训练在有限数据任务上的性能提高程度。虽然本文没有明确衡量数据效率的提高，但强调这是迁移学习范式的主要好处之一。

关于运行间方差，我们发现对于大多数任务，运行间的标准偏差小于任务的基线分数的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上，我们的基线模型的平均F1分数为91.22，标准差为3.237(参见Table 16)，这可能部分是因为CB的验证集只包含56示例。请注意，GLUE和SuperGLUE的得分是计算每个基准任务得分的平均值。因此，我们需要提醒，CoLA、CB和COPA的高运行间方差可能会使仅使用GLUE和SuperGLUE分数来比较模型变得更加困难。

3.2 架构

虽然Transformer最初是通过编码器-解码器架构引入的，但许多针对NLP的迁移学习的现代工作都使用了替代架构。在本节中，我们将回顾并比较这些架构变体。

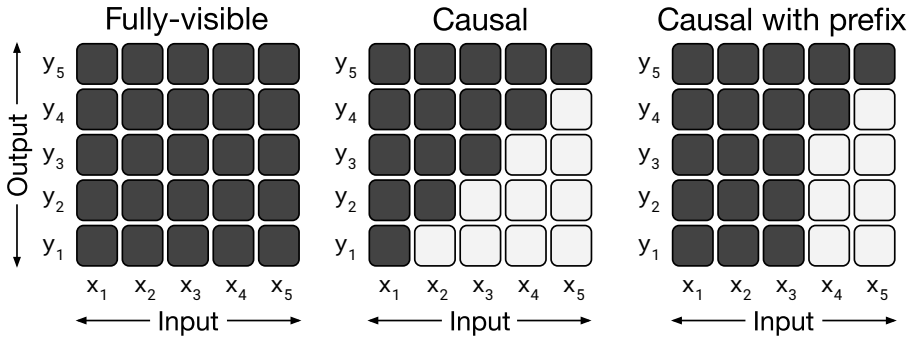

Figure 3: 代表不同注意力掩模模式的矩阵。自注意力机制的输入和输出分别表示为x和y。行i和列j的暗单元格表示自注意力机制允许在输出时间步i关注输入元素j。光电池表明不允许自注意力机制关注相应的i和j组合。左:全可见的掩码允许自注意力机制在每个输出时间步关注完整的输入。中间:因果掩码防止i th输出元素依赖于来自“未来”的任何输入元素。右:带前缀的因果掩码允许自注意力机制对输入序列的一部分使用全可见掩码。

3.2.1 模型结构

不同架构的一个主要区别因素是模型中不同注意力机制使用的“掩码”。回想一下，Transformer中的自注意力操作将一个序列作为输入，并输出一个相同长度的新序列。输出序列的每个元素都是通过计算输入序列元素的加权平均值产生的。具体来说，让y_i表示输出序列的i th个元素，让x_j表示输入序列的j th个条目。y_i用$\sum_j w_{i,j}x_j$计算，其中$w_{i,j}$是由自注意力机制产生的标量权重，是x_i和x_j的函数。然后，注意力掩码用于将某些权重归零，以约束在给定输出时间步长可以关注输入的哪些条目。我们将考虑的掩模的图表显示在Figure 3中。例如，如果$j > i$，因果掩码(Figure 3，中间)将任何$w_{i,j}$设置为零。

我们考虑的第一个模型结构是一个编码器-解码器Transformer，由两层堆栈组成:编码器，提供一个输入序列，解码器产生一个新的输出序列。Figure 4的左侧面板显示了这种架构变体的示意图。

编码器使用“全可见”注意力掩模。全可见屏蔽允许自注意力机制在产生其输出的每个条目时关注输入的任何条目。我们在Figure 3(左)中可视化这个掩码模式。这种形式的掩码适用于处理“前缀‘，即为模型提供的一些上下文，稍后在进行预测时使用。BERT (Devlin et al., 2018)还使用了完全可见的掩码模式，并向输入添加了一个特殊的“分类”标记。然后，使用BERT在分类标记对应的时间步长的输出对输入序列进行分类预测。

Transformer解码器中的自注意力操作使用"因果"掩蔽模式。当产生输出序列的i第th个条目时，因果掩盖阻止模型关注输入序列$j > i$的j第th个条目。这在训练过程中使用，以便模型在产生输出时无法“看到未来”。这个掩码模式的注意力矩阵显示在Figure 3中。

编码器-解码器Transformer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采样一个token，并将样本反馈到模型中以产生对下一个输出时间步的预测，以此类推。因此，Transformer解码器(没有编码器)可以用作语言模型(LM)，即仅为下一步预测训练的模型(Liu et al., 2018; Radford et al., 2018; Al-Rfou et al., 2019)。这构成了我们考虑的第二个模型结构。这个架构的示意图显示在Figure 4，

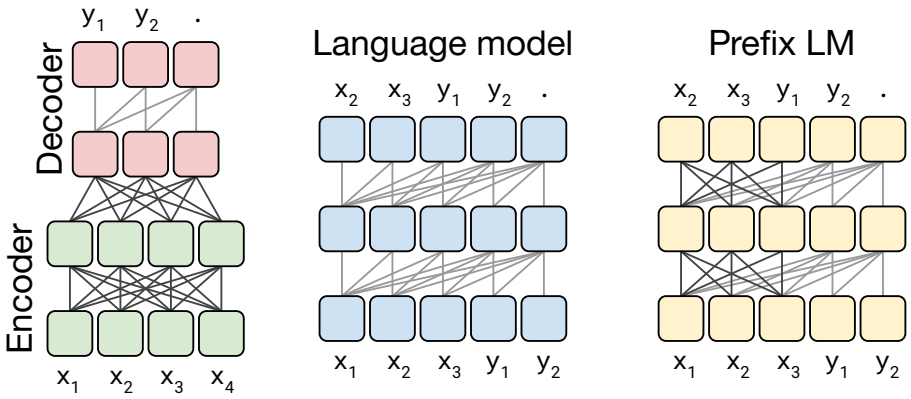

Figure 4: 我们考虑的Transformer架构变体示意图。在这个图中，块代表序列的元素，线代表注意力可见性。不同颜色的块组表示不同的Transformer层堆栈。深灰色线对应全可见掩蔽，浅灰色线对应因果掩蔽。我们用“。表示一个特殊的序列结束标记，表示预测的结束。输入和输出序列分别表示为x和y。左:标准的编码器-解码器架构在编码器和编码器-解码器注意力中使用全可见掩码，在解码器中使用因果掩码。中间:语言模型由单个Transformer层堆栈组成，并被输入和目标的连接所提供，自始至终使用因果掩码。右:为语言模型添加前缀对应于允许对输入进行完全可见的屏蔽。

中间。事实上，NLP迁移学习的早期工作使用这种带有语言建模目标的架构作为预训练方法(Radford et al., 2018)。

语言模型通常用于压缩或序列生成(Graves, 2013)。不过，也可以在文本到文本框架中使用它们，只需将输入和目标连接起来即可。举个例子，考虑英语到德语翻译的情况:如果我们有一个输入句子’ ’的训练数据点，那就好。“并瞄准‘Das ist gut’。，我们将简单地对连接的输入序列进行下一步预测训练模型。目标:Das ist gut。” 如果我们想获得此示例的模型预测，则将向模型提供前缀“将英语翻译为德语:很好。目标:”，并将被要求自动回归生成序列的剩余部分。通过这种方式，模型可以在给定输入的情况下预测出一个输出序列，满足文本到文本任务的需要。这种方法最近被用于表明语言模型可以在没有监督的情况下学习执行一些文本到文本的任务(Radford et al., 2019)。

在文本到文本的环境中使用语言模型的一个基本的和经常被引用的缺点是，因果掩盖迫使模型对输入序列的i第th个条目的表示仅依赖于条目，直到i。要了解为什么这可能是不利的，请考虑文本到文本框架，其中在要求模型进行预测之前为模型提供前缀/上下文(例如，前缀是一个英语句子，并要求模型预测德语翻译)。在完全因果掩盖的情况下，模型对前缀状态的表示只能依赖于前缀的先验项。因此，当预测输出的一个条目时，模型将关注不必要限制的前缀表示。类似的论点也反对在序列到序列模型中使用单向循环神经网络编码器(Bahdanau et al., 2015)。

在基于transformer的语言模型中，只需更改掩码模式就可以避免此问题。我们在序列的前缀部分使用全可见掩码，而不是使用因果掩码。这种掩码模式和由此产生的”前缀LM”(我们考虑的第三种模型结构)的原理图分别在Figures 3 and 4的最右边面板中加以说明。在上面提到的英德翻译示例中，将对前缀’ ’进行完全可见的屏蔽:将英语翻译为德语:这很好。目标:“因果掩盖将在训练期间用于预测目标的‘Das ist肠道’。” 在文本到文本框架中

使用前缀LM最初是由Liu et al. (2018)提出的。最近，Dong et al. (2019)表明这种架构在各种文本到文本任务上都是有效的。这种架构类似于编码器-解码器模型，在编码器和解码器中共享参数，并将编码器-解码器的注意力替换为输入和目标序列的全部注意力。

我们注意到，在遵循我们的文本到文本框架时，前缀LM架构与分类任务的BERT (Devlin et al., 2018)非常相似。要知道为什么，考虑MNLI基准的一个例子，前提是“我讨厌鸽子”。，假设是‘我对鸽子的感情充满仇恨。“正确的标签是‘蕴含’。”为了将此示例输入到语言模型中，我们将其转换为序列“mnli前提:我讨厌鸽子。假设:我对鸽子充满敌意。目标:蕴涵”。在这种情况下，全可见前缀将对应于整个输入序列，直到单词“target: “，可以将其视为类似于BERT中使用的“classification ‘ token。因此，我们的模型将对整个输入具有完全的可视性，然后将任务是通过输出单词“‘蕴涵”进行分类。给定任务前缀(在本例中为” mnli”)，模型很容易学习输出一个有效的类标签。因此，前缀LM和BERT架构之间的主要区别是，在前缀LM中，分类器简单地集成到Transformer解码器的输出层。

3.2.2 比较不同的模型结构

为了对这些架构变体进行实验比较，我们希望我们认为的每个模型在某种有意义的方式上是等效的。如果两个模型有相同数量的参数，或者它们需要大致相同的计算量来处理给定的(输入序列，目标序列)对，我们可以说它们是等效的。不幸的是，无法同时根据这两个标准将编码器-解码器模型与语言模型架构(包括单个Transformer堆栈)进行比较。要了解原因，首先请注意编码器中有L层，解码器中有L层的编码器-解码器模型与具有$2L$层的语言模型的参数数量大致相同。然而，相同的$L + L$编码器-解码器模型将具有与只有L层的语言模型大致相同的计算成本。这是因为语言模型中的L层必须同时应用于输入序列和输出序列，而编码器仅应用于输入序列，解码器仅应用于输出序列。请注意，这些等价是近似的——由于编码器-解码器的注意力，在解码器中有一些额外的参数，并且在注意力层中也有一些计算成本，这些计算成本在序列长度上是二次的。然而，在实践中，我们观察到L层语言模型与$L + L$层编码器-解码器模型几乎相同的步骤时间，表明其计算成本大致相同。此外，对于我们考虑的模型大小，编码器-解码器注意力层中的参数数量约为总参数数量的10%，因此我们做了简化的假设，即$L + L$层的编码器-解码器模型与$2L$层语言模型具有相同的参数数量。

为了提供一个合理的比较手段，我们考虑了编码器-解码器模型的多种配置。我们将把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表示$L + L$层编码器-解码器模型或L层仅解码器模型处理给定输入-目标对所需的FLOPs数量。总的来说，我们将比较:

- 编码器-解码器模型，编码器中有L层，解码器中有L层。该模型的参数为$2P$，计算开销为M FLOPs。
- 一个等效的模型，但在编码器和解码器之间共享参数，导致P参数和M -FLOP计算成本。
- 编码器-解码器模型在编码器和解码器中各有$L/2$层，给出P参数和$M/2$ -FLOP成本。
- 一个只有解码器的语言模型，具有L层和P参数，由此产生的计算成本为M FLOPs。
- 一个仅解码器前缀LM，具有相同的架构(因此参数和计算成本相同)，但对输入具有完全可见的自注意力。

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39
Encoder-decoder	LM	$2P$	M	79.56	18.59	76.02	64.29	26.27	39.17	26.86
Enc-dec, shared	LM	P	M	79.60	18.13	76.35	63.50	26.62	39.17	27.05
Enc-dec, 6 layers	LM	P	$M/2$	78.67	18.26	75.32	64.06	26.13	38.42	26.89
Language model	LM	P	M	73.78	17.54	53.81	56.51	25.23	34.31	25.38
Prefix LM	LM	P	M	79.68	17.84	76.87	64.86	26.28	37.51	26.76

Table 2: Section 3.2.2中描述的不同体系结构变体的性能。我们使用P表示12层基本Transformer层堆栈中的参数数量，M表示使用编码器-解码器模型处理序列所需的FLOPs。我们使用去噪目标(在Section 3.1.4中描述)和自回归目标(通常用于训练语言模型)来评估每个架构变体。

3.2.3 目标

作为一个无监督目标，我们将考虑一个基本的语言建模目标以及Section 3.1.4中描述的基线去噪目标。我们包括了语言建模目标，因为它曾经作为预训练目标(Dai and Le, 2015; Ramachandran et al., 2016; Howard and Ruder, 2018; Radford et al., 2018; Peters et al., 2018)，以及它自然适合我们考虑的语言模型架构。对于在进行预测之前采集前缀的模型(编码器-解码器模型和前缀LM)，我们从未标记的数据集中对一段文本进行采样，并选择一个随机点将其分为前缀和目标部分。对于标准语言模型，我们训练模型来预测从头到尾的整个跨度。无监督去噪目标是为文本到文本模型设计的;为了使其与语言模型一起使用，我们将输入和目标连接起来，如Section 3.2.1所述。

3.2.4 结果

我们比较的每种架构所取得的分数显示在Table 2中。对于所有任务，具有去噪目标的编码器-解码器架构表现最佳。这种变体具有最高的参数数量($2P$)，但与P -参数解码器模型相同的计算成本。令人惊讶的是，我们发现在编码器和解码器中共享参数的性能几乎一样好。相比之下，将编码器和解码器堆栈的层数减半会显著降低性能。并发工作(Lan et al., 2019)还发现，在Transformer块之间共享参数可以有效地降低总参数数量，而不会牺牲太多性能。XLNet也与具有去噪目标(Yang et al., 2019)的共享编码器-解码器方法有一些相似之处。共享参数编码器-解码器的性能优于仅解码器的前缀LM，表明显式编码器-解码器注意力的添加是有益的。证实了一个被广泛接受的概念，即与语言建模目标相比，使用去噪目标总是会导致更好的下游任务性能。此前，Devlin et al. (2018)、Voita et al. (2019)和Lample and Conneau (2019)等人也观察到了这一点。在下一节中，我们将对无监督目标进行更详细的探索。

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，模型可以获得应用于下游任务的通用知识。这导致了各种预训练目标的发展(Dai and Le, 2015; Ramachandran

Objective	Inputs	Targets
Prefix language modeling	Thank you for inviting	me to your party last week .
BERT-style Devlin et al. (2018)	Thank you <M> <M> me to your party apple week .	*(original text)*
Deshuffling	party me for your to . last fun you inviting week Thank	*(original text)*
MASS-style Song et al. (2019)	Thank you <M> <M> me to your party <M> week .	*(original text)*
I.i.d. noise, replace spans	Thank you <X> me to your party <Y> week .	<X> for inviting <Y> last <Z>
I.i.d. noise, drop tokens	Thank you me to your party week .	for inviting last
Random spans	Thank you <X> to <Y> week .	<X> for inviting me <Y> your party last <Z>

Table 3: 我们考虑的一些无监督目标产生的输入和目标的例子，应用于输入文本“感谢您邀请我参加您上周的派对”。请注意，我们所有的目标都处理标记化文本。对于这个特定的句子，所有的单词都被我们的词汇表映射到一个标记。我们将*(原文)*作为目标，以表示模型的任务是重建整个输入文本。< m >表示共享掩码令牌，< x >、< y >和< z >表示哨兵令牌，这些令牌分配了唯一的令牌id。bert风格的目标(第二行)包含一个损坏，其中一些标记被随机标记ID替换;我们通过灰色的单词apple来展示这一点。

et al., 2016; Radford et al., 2018; Devlin et al., 2018; Yang et al., 2019; Liu et al., 2019b; Wang et al., 2019a; Song et al., 2019; Dong et al., 2019; Joshi et al., 2019)。在本节中，我们对无监督目标空间进行程序性探索。在许多情况下，我们不会完全复制现有的目标-一些将被修改以适应我们的文本到文本编码器-解码器框架，在其他情况下，我们将使用结合多种常见方法的概念的目标。

总的来说，我们所有的目标都是从未标记的文本数据集中采集与分词后的文本跨度相对应的token id序列。对标记序列进行处理以产生(损坏的)输入序列和相应的目标。然后，像往常一样以最大似然训练模型来预测目标序列。我们提供了在Table 3中考虑的许多目标的说明性例子。

3.3.1 不同的高级方法

首先，比较了三种受常用目标启发，但在方法上有显著差异的技术。首先，我们包括一个基本的“前缀语言建模”目标，正如在Section 3.2.3中使用的那样。这种技术将一段文本分成两个部分，一个用作编码器的输入，另一个用作解码器预测的目标序列。其次，本文考虑一个受BERT中使用的"掩码语言建模" (MLM)目标启发的目标(Devlin et al., 2018)。MLM采取一段文本，并破坏15%的token。损坏令牌的90%被替换为特殊的掩码令牌，10%被替换为随机令牌。由于BERT是一个纯编码器模型，因此在预训练期间，其目标是在编码器输出时重建被掩码的标记。在编码器-解码器的情况下，我们简单地使用整个未损坏的序列作为目标。注意，这与我们的基线目标不同，基线目标只使用损坏的标记作为目标;我们在Section 3.3.2中比较了这两种方法。最后，我们还考虑了一个基本的去混杂目标，如在(Liu et al., 2019a)中，它被应用于去噪顺序自编码器。这种方法获取一个标记序列，对其进行打乱，然后使用原始打乱的序列作为目标。我们在Table 3的前三行中提供了这三个方法的输入和目标的示例。

这三个目标的性能在Table 4中显示。总的来说，我们发现bert风格的目标表现最好，尽管前缀语言建模目标在翻译任务上取得了类似的性能。事实上，BERT目标的动机是超越基于语言模型的预训练。deshuffling目标的表现比前缀语言建模和bert风格的目标都差得多。

Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
Prefix language modeling	80.69	18.94	77.99	65.27	**26.86**	39.73	**27.49**
BERT-style (Devlin et al., 2018)	**82.96**	**19.17**	**80.65**	**69.85**	**26.78**	**40.03**	**27.41**
Deshuffling	73.17	18.59	67.61	58.47	26.11	39.30	25.62

Table 4: Section 3.3.1中描述的三个不同的预训练目标的表现。

Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
BERT-style (Devlin et al., 2018)	82.96	19.17	**80.65**	69.85	26.78	**40.03**	27.41
MASS-style (Song et al., 2019)	82.32	19.16	80.10	69.28	26.79	**39.89**	27.55
★ Replace corrupted spans	83.28	**19.24**	**80.88**	**71.36**	**26.98**	39.82	**27.65**
Drop corrupted tokens	**84.44**	**19.31**	**80.52**	68.67	**27.07**	39.76	**27.82**

Table 5: bert式预训练目标变体的比较。在前两种变体中，模型被训练以重建原始的未损坏的文本段。在后两种情况下，该模型仅预测错误标记序列。

3.3.2 简化BERT目标

基于上一节的结果，我们现在将重点探索对bert式去噪目标的修改。这个目标最初是作为用于分类和跨度预测的纯编码器模型的预训练技术提出的。因此，可以对其进行修改，使其在我们的编码器-解码器文本到文本的设置中表现更好或更高效。

首先，我们考虑bert式目标的一个简单变体，其中不包括随机token交换步骤。由此产生的目标只是将输入中标记的15%替换为掩码标记，并训练模型以重建原始的未损坏序列。一个类似的掩蔽目标被Song et al. (2019)使用，它被称为“质量”，所以我们称这种变体为“质量风格”的目标。其次，我们感兴趣的是是否有可能避免预测整个未损坏的文本跨度，因为这需要在解码器中的长序列上进行自注意力。我们考虑两种策略来实现这一点:首先，不是将每个损坏的令牌替换为掩码令牌，而是将每个连续跨度的损坏令牌全部替换为唯一的掩码令牌。然后，目标序列成为‘corrupted ‘ span的拼接，每个span的前缀是用于替换输入中的掩码令牌。这是我们在基线中使用的预训练目标，在Section 3.1.4中描述。其次，考虑一种变体，简单地从输入序列中完全删除损坏的标记，并让模型按顺序重建丢失的标记。Table 3的第五行和第六行显示了这些方法的示例。

原始bert风格的目标与这三种替代方案的实证比较见Table 5。我们发现，在我们的设置中，所有这些变体的表现都类似。唯一的例外是完全丢弃损坏的标记产生了胶水分数的小改进，这要归功于CoLA的显著更高的分数(60.04，与我们的基线平均值53.84相比，请参见Table 16)。这可能是因为CoLA涉及对给定句子是否在语法和语法上可接受进行分类，并且能够确定标记何时缺失与检测可接受性密切相关。然而，完全丢弃标记比用SuperGLUE上的哨兵标记替换它们的性能更差。两种不需要预测完整原始序列的变体(“替换损坏的片段‘和“删除损坏的片段‘)都具有潜在的吸引力，因为它们使目标序列更短，从而使训练更快。展望未来，我们将探索用哨兵标记替换损坏的span并仅预测损坏的标记的变体(如我们的基线目标)。

3.3.3 改变腐败率

到目前为止，我们已经损坏了15%的token，即BERT中使用的值(Devlin et al., 2018)。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出错率是否对我

Corruption rate	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
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**

Table 6: 不同腐败率下i.i.d.腐败目标的表现。

Span length	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline (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**

Table 7: 不同平均跨度长度下的跨度破坏目标(受Joshi et al. (2019)启发)的性能。在所有情况下，我们破坏了15%的原始文本序列。

们更好。我们在Table 6中比较了10%、15%、25%和50%的腐败率。总的来说，我们发现腐败率对模型的性能有有限的影响。唯一的例外是，我们考虑的最大腐败率(50%)导致胶水和SQuAD的性能显著下降。使用较大的腐败率还会导致目标较长，这可能会减慢训练速度。基于这些结果和BERT设定的历史先例，我们将使用15%的腐败率继续前进。

3.3.4 腐蚀的跨度

现在我们转向通过预测更短的目标来加快训练的目标。到目前为止，我们使用的方法对每个输入令牌进行i.i.d.决策，以确定是否破坏它。当多个连续的令牌被损坏时，它们被视为一个“span ‘，并使用一个唯一的掩码令牌替换整个span。用单个标记替换整个span会导致未标记的文本数据被处理为更短的序列。由于我们使用的是i.i.d.腐败策略，因此大量腐败的令牌并不总是连续出现。因此，我们可以通过专门破坏token范围而不是以i.i.d.方式破坏单个token来获得额外的速度。破坏span之前也被认为是BERT的预训练目标，它被发现可以提高性能(Joshi et al., 2019)。

为测试这一想法，本文考虑一个目标，专门破坏连续的、随机间隔的标记范围。这个目标可以通过被损坏的标记的比例和损坏的跨度的总数来参数化。然后随机选择跨度长度以满足这些指定的参数。例如，如果我们正在处理一个500标记序列，并且我们已经指定标记的15%应该损坏，并且应该有25总跨度，那么损坏的标记的总数将是$500 \times 0.15 = 75$，平均跨度长度将是$75/25 = 3$。请注意，给定原始序列长度和损坏率，我们可以通过平均跨度长度或跨度总数量等效地将此目标参数化。

我们将span-corruption目标与Table 7中的i.i.d-corruption目标进行了比较。我们在所有情况下使用15%的腐败率，并使用2, 3, 5和10的平均跨度长度进行比较。同样，我们发现这些目标之间的差异有限，尽管平均跨度长度为10的版本在某些情况下略逊于其他值。我们还特别发现，在大多数非翻译基准上，使用3的平均跨度长度略(但显著)优于i.i.d.目标。幸运的是，与i.i.d.噪声方法相比，跨度破坏目标还在训练过程中提供了一些加速，因为平均而言，跨度破坏产生的序列更短。

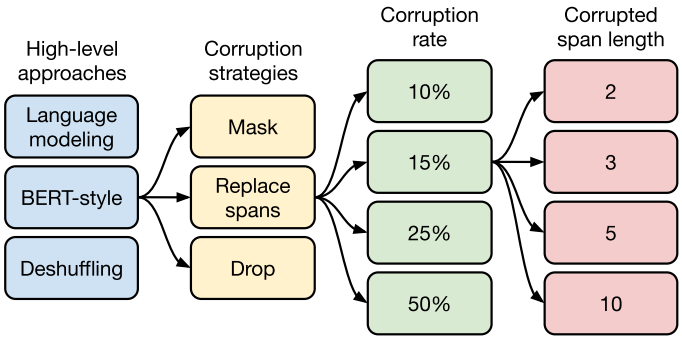

Figure 5: 探索无监督目标的流程图。本文首先考虑了Section 3.3.1中的几种不同方法，发现bert风格的去噪目标表现最好。然后，考虑各种方法来简化BERT目标，使其在Section 3.3.2中产生更短的目标序列。鉴于用哨兵标记替换掉的span表现良好，并导致短目标序列，在Section 3.3.3中，我们实验了不同的损坏率。评估了一个有意破坏Section 3.3.4中连续标记范围的目标。

3.3.5 讨论

Figure 5 展示了我们在探索无监督目标过程中所做选择的流程图。总的来说，我们观察到的性能最显著的差异是，去噪目标的表现优于语言建模和预训练的解混。我们没有观察到所探索的去噪目标的许多变体之间的显著差异。然而，不同的目标(或目标的参数化)可能导致不同的序列长度，从而不同的训练速度。这意味着，我们在这里考虑的去噪目标中，应该主要根据它们的计算成本进行选择。结果还表明，对与本文考虑的目标类似的目标的进一步探索，可能不会为所考虑的任务和模型带来显著的收益。相反，探索利用未标记数据的完全不同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，预训练数据集本身是迁移学习管道的重要组成部分。然而，与目标和基准不同，新的预训练数据集通常不被视为本身的重大贡献，通常不与预训练模型和代码一起发布。相反，它们通常是在介绍新方法或模型的过程中引入的。因此，不同的预训练数据集之间的比较相对较少，并且缺乏用于预训练的“标准”数据集。最近一些值得注意的例外情况(Baevski et al., 2019; Liu et al., 2019c; Yang et al., 2019)将在一个新的大型数据集(通常是常见的爬虫来源)上进行预训练与使用一个较小的预先存在的数据集(通常是维基百科)进行比较。为了更深入地探讨预训练数据集对性能的影响，在本节中，我们比较C4数据集的变体和其他潜在的预训练数据来源。我们发布了所有我们认为是TensorFlow数据集一部分的C4数据集变体。[11]

3.4.1 未标记数据集

在创建C4时，我们开发了各种启发式方法来从普通爬行器中过滤web提取的文本(有关描述，请参阅Section 2.2)。除了将其与其他过滤方法和常见的预训练数据集进行比较之外，

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

我们还有兴趣测量这种过滤是否会提高下游任务的性能。为此，我们在以下数据集上预训练后，比较了基线模型的性能:

C4 作为基线，我们首先考虑对建议的未标记数据集进行预训练，如Section 2.2所述。

Unfiltered C4 为了衡量我们在创建C4时使用的启发式过滤的效果(重复数据删除、删除不好的单词、只保留句子等)，我们还生成了一个放弃这种过滤的C4的替代版本。注意，我们仍然使用`langdetect`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`有时会给非自然英语文本分配低概率。

RealNews-like 最近的工作使用了从新闻网站(Zellers et al., 2019; Baevski et al., 2019)提取的文本数据。为了与这种方法进行比较，我们通过额外过滤C4以仅包括"RealNews"数据集(Zellers et al., 2019)中使用的其中一个域的内容来生成另一个未标记的数据集。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;唯一的区别是，我们表面上省略了任何非新闻内容。

WebText-like 类似地，WebText数据集(Radford et al., 2019)仅使用提交到内容聚合网站Reddit并获得至少3分的网页的内容。提交到Reddit的网页得分是根据支持(upvote)或反对(downvote)网页的用户比例计算的。使用Reddit评分作为质量信号背后的想法是，该网站的用户只会给高质量的文本内容投票。为了生成一个可比较的数据集，我们首先尝试删除C4中所有不是来自OpenWebText工作准备的列表中出现的URL的内容。[12] 然而，这导致了相对较少的内容——只有2 GB——因为大多数页面从未出现在Reddit上。回想一下，C4是基于一个月的常见爬行数据创建的。为了避免使用过于小的数据集，我们从Common Crawl上下载了2018年8月至2019年7月12个月的数据，对C4应用了我们的启发式过滤，然后应用了Reddit过滤器。这产生了一个17 GB的类似WebText的数据集，与原始40GB的WebText数据集(Radford et al., 2019)的大小相当。

Wikipedia 维基百科网站由数百万百科全书文章组成。本网站的内容受到严格的质量指导方针，因此已被用作干净和自然的文本的可靠来源。我们使用来自TensorFlow数据集的英文维基百科文本数据[13]，它省略了文章中的任何标记或参考部分。

Wikipedia + Toronto Books Corpus 使用来自维基百科的预训练数据的缺点是它只能表示自然文本(百科文章)的一个可能领域。为了缓解这个问题，BERT (Devlin et al., 2018)将来自维基百科的数据与多伦多图书语料库(TBC) (Zhu et al., 2015)相结合。TBC包含从电子书中提取的文本，它代表了自然语言的不同领域。BERT的流行导致了Wikipedia + TBC的组合被用于许多后续工作。

在每个数据集上进行预训练后取得的结果显示在Table 8中。第一个明显的结论是，从C4中删除启发式过滤会均匀地降低性能，并使未过滤的变体在每个任务中表现最差。除此之外，我们发现在某些情况下，具有更受限域的预训练数据集优于多样化的C4数据集。例如，使用Wikipedia + TBC语料库产生了73.24的SuperGLUE分数，超过了我们的基线分数(使用C4) 71.36。这几乎完全归因于性能的提高，从25.78(基线，C4)到50.93(维基百科+ TBC)对MultiRC的精确匹配分数(见Table 16)。MultiRC是一个阅读理解数据集，其最大的数据来源是小说，这正是TBC所涵盖的领域。类似地，使用RealNews-like数据集进

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

Data set	Size	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ C4	745GB	83.28	**19.24**	80.88	71.36	**26.98**	**39.82**	**27.65**
C4, unfiltered	6.1TB	81.46	19.14	78.78	68.04	26.55	39.34	27.21
RealNews-like	35GB	**83.83**	**19.23**	80.39	72.38	**26.75**	**39.90**	**27.48**
WebText-like	17GB	**84.03**	**19.31**	**81.42**	71.40	**26.80**	**39.74**	**27.59**
Wikipedia	16GB	81.85	**19.31**	81.29	68.01	**26.94**	39.69	**27.67**
Wikipedia + TBC	20GB	83.65	**19.28**	**82.08**	**73.24**	**26.77**	39.63	**27.57**

Table 8: 在不同数据集上进行预训练的性能。前四种变体基于我们新的C4数据集。

行预训练，使ReCoRD(衡量新闻文章阅读理解能力的数据集)的精确匹配分数从68.16增加到73.72。最后一个例子是，使用来自维基百科的数据在SQuAD上取得了显著(但不那么显著)的效果，SQuAD是一个问答数据集，其中的段落来自维基百科。在之前的工作中也进行了类似的观察，例如Beltagy et al. (2019)发现，对研究论文的文本进行预训练BERT可以提高其在科学任务上的性能。这些发现背后的主要教训是，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可以快速适应任意领域语言任务的模型，这并不奇怪，但也不令人满意。Liu et al. (2019c)还观察到，在更多样化的数据集上进行预训练，可以改善下游任务。这一观察也推动了自然语言处理领域适应的平行研究。有关该领域的调查请参见Ruder (2019); Li (2012)。

只在单个域上进行预训练的缺点是得到的数据集通常要小得多。类似地，虽然类似webtext的变体在我们的基线设置中表现与C4数据集一样好或更好，但基于reddit的过滤产生的数据集比C4小40×，尽管它是基于来自Common Crawl的12×更多数据。然而，请注意，在我们的基线设置中，我们只对$2^{35} \approx 34$B token进行预训练，这只比我们考虑的最小预训练数据集大大约8倍。我们将在下一节中研究在什么时候使用较小的预训练数据集提出一个问题。

3.4.2 预训练数据集大小

我们用于创建C4的管道被设计为能够创建非常大的预训练数据集。访问如此多的数据使我们能够在不重复示例的情况下对模型进行预训练。目前尚不清楚在预训练期间重复示例对下游性能是有益还是有害，因为我们的预训练目标本身是随机的，并且可以帮助防止模型多次看到相同的精确数据。

为了测试有限的未标记数据集大小的影响，我们在人为截断的C4版本上预训练了我们的基线模型。回想一下，我们在$2^{35} \approx 34$B token(占C4总大小的一小部分)上预训练了基线模型。考虑对由2^{29}, 2^{27}, 2^{25}和2^{23} token组成的C4的截断变体进行训练。这些大小对应于在预训练过程中分别重复数据集64、256、1,024和4,096次。

由此产生的下游性能如Table 9所示。正如预期的那样，随着数据集的缩小，性能会下降。我们怀疑这可能是因为模型开始记忆预训练数据集。为了衡量这是否正确，我们在Figure 6中绘制每种数据集大小的训练损失。实际上，随着预训练数据集的缩小，该模型获得了明显更小的训练损失，这表明可能的记忆。Baevski et al. (2019)类似地观察到，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，当预训练数据集只重复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是无害的。然而，考虑到额外的预训练可能是有益的(我们将在Section 3.6中显示)，并且获得额外的未标记数据廉价且容易，我们建议尽可能使用大型预训练数据集。我们还注意到，这种影响对于较大的模型尺寸可能更明显，即较大的模型可能更容易对较小的预训练数据集进行过拟合。

Number of tokens	Repeats	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Full data set	0	**83.28**	**19.24**	**80.88**	**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

Table 9: 在预训练期间测量重复数据的效果。在这些实验中，我们只使用来自C4的第一个N标记(第一列中显示了N的不同值)，但仍然在2^{35}标记上进行预训练。这导致数据集在预训练过程中不断重复(每个实验的重复次数如第二列所示)，这可能会导致记忆(见Figure 6)。

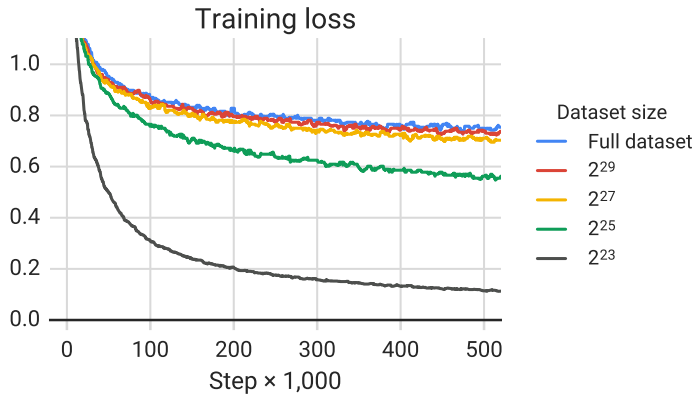

Figure 6: 我们原始C4数据集的预训练损失以及4人为截断版本。列出的长度指的是每个数据集中的标记数量。考虑的四种大小对应于在预训练过程中在64和4,096之间重复数据集。使用较小的数据集会导致较小的训练损失值，这可能意味着对未标记数据集的一些记忆。

3.5 培训策略

到目前为止，我们考虑的是模型的所有参数在对个别监督任务进行微调之前在无监督任务上进行预训练的设置。虽然这种方法很简单，但已经提出了各种用于在下游/监督任务上训练模型的替代方法。在本节中，我们比较了微调模型的不同方案，以及在多个任务上同时训练模型的方法。

3.5.1 微调方法

有人认为，对所有模型参数进行微调可能会导致次优结果，特别是在低资源任务上(Peters et al., 2019)。关于文本分类任务的迁移学习的早期结果主张只对一个小型分类器的参数进行微调，该分类器是由固定的预训练模型产生的句子嵌入(Subramanian et al., 2018; Kiros et al., 2015; Logeswaran and Lee, 2018; Hill et al., 2016; Conneau et al., 2017)。这种方法不太适用于我们的编码器-解码器模型，因为必须训练整个解码器以输出给定任务的目标序列。本文专注于两种可选的微调方法，这些方法只更新编码器-解码器模型的一部分参数。

Fine-tuning method	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ All parameters	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Adapter layers, $d = 32$	80.52	15.08	79.32	60.40	13.84	17.88	15.54
Adapter layers, $d = 128$	81.51	16.62	79.47	63.03	19.83	27.50	22.63
Adapter layers, $d = 512$	81.54	17.78	79.18	64.30	23.45	33.98	25.81
Adapter layers, $d = 2048$	81.51	16.62	79.47	63.03	19.83	27.50	22.63
Gradual unfreezing	82.50	18.95	79.17	**70.79**	26.71	39.02	26.93

Table 10: 只更新模型参数子集的不同备选微调方法的比较。对于适配器层，d指的是适配器的内部维度。

第一个是“适配器层”(Houlsby et al., 2019; Bapna et al., 2019)，其动机是在微调时保持原始模型的大部分固定。适配器层是额外的dense-ReLU-dense块，在Transformer的每个块中的每个预先存在的前馈网络之后添加。这些新的前馈网络的设计使其输出维度与输入相匹配。这使得它们可以插入到网络中，而无需对结构或参数进行额外的更改。微调时，只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部维数d，它改变了添加到模型中的新参数的数量。我们对d使用不同的值进行实验。

我们考虑的第二种替代微调方法是“逐步解冻”(Howard and Ruder, 2018)。在逐步解冻过程中，随着时间的推移，越来越多的模型参数被微调。逐渐解冻最初应用于由单个层栈组成的语言模型架构。在这种设置中，在微调开始时，只更新最后一层的参数，然后在训练一定数量的更新后，也包括倒数第二层的参数，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，我们并行地逐渐解冻编码器和解码器中的层，在这两种情况下都从顶部开始。由于输入嵌入矩阵和输出分类矩阵的参数是共享的，我们通过微调来更新它们。回想一下，我们的基线模型由12层组成，每个层在编码器和解码器中，并针对2^{18}步骤进行了微调。因此，我们将微调过程细分为$2^{18}/12$步骤的12 episode，每个步骤并在n th episode中从$12 - n$层训练到12。我们注意到Howard and Ruder (2018)建议在每个训练周期后微调一个额外的层。然而，由于我们的监督数据集在大小上差异很大，并且由于我们的一些下游任务实际上是许多任务(GLUE和SuperGLUE)的混合物，因此我们相反采取更简单的策略，在每个$2^{18}/12$步骤后微调一个额外的层。

这些微调方法的性能比较见Table 10。对于适配器层，我们使用32, 128, 512, 2048的内部维度d报告性能。根据过去的结果(Houlsby et al., 2019; Bapna et al., 2019)我们发现，像SQuAD这样的低资源任务在d的小值下工作得很好，而较高资源任务需要很大的维度才能实现合理的性能。这表明，只要将维度适当缩放到任务大小，适配器层可能是一种很有前途的技术，可以对更少的参数进行微调。请注意，在我们的例子中，我们通过连接它们的组成数据集将GLUE和SuperGLUE都视为单个的“任务”，因此尽管它们包含一些低资源的数据集，但组合的数据集足够大，需要很大的值d。我们发现逐渐解冻会导致所有任务的性能略有下降，尽管它在微调期间确实提供了一些加速。更仔细地调整解冻时间表可能会取得更好的效果。

3.5.2 多任务学习

到目前为止，我们一直在单个无监督学习任务上预训练模型，然后在每个下游任务上分别进行微调。另一种方法，称为“多任务学习”(Ruder, 2017; Caruana, 1997)，是一次在多个任务上训练模型。这种方法通常具有训练一个可以同时执行许多任务的单个模型的目标，即模型及其大多数参数在所有任务中共享。本文稍微放松了这一目标，转而研究同时在多个

任务上进行训练的方法，以最终产生在每个单独任务上表现良好的单独参数设置。例如，我们可能在多个任务上训练一个模型，但在报告性能时，我们可以为每个任务选择不同的检查点。这放松了多任务学习框架，与我们迄今为止考虑的先训练再微调的方法相比，使其更加均衡。在统一的文本到文本框架中，“多任务学习”只是对应于将数据集混合在一起。因此，在使用多任务学习时，我们仍然可以在未标记的数据上进行训练，将无监督任务视为混合在一起的任务之一。相比之下，大多数多任务学习的NLP应用都添加了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 et al., 2019b)。

正如Arivazhagan et al. (2019)指出的那样，在多任务学习中，一个极其重要的因素是每个任务中应该训练模型的数据量。我们的目标是不欠训练或过度训练模型——也就是说，我们希望模型从给定的任务中看到足够的数据，使它能够很好地执行任务，但不希望看到太多的数据，使它能够记住训练集。如何准确地设置来自每个任务的数据比例可能取决于各种因素，包括数据集大小、学习任务的“难度”(即模型在能够有效执行任务之前必须看到多少数据)、正则化等。另一个问题是潜在的“任务干扰”或“负迁移”，在一个任务上取得良好的性能可能会阻碍另一个任务的性能。鉴于这些问题，本文首先探索了设置每个任务的数据比例的各种策略。Wang et al. (2019a)也进行了类似的探索。

Examples-proportional mixing 影响模型对给定任务过拟合速度的一个主要因素是任务的数据集大小。因此，设置混合比例的一种自然方法是根据每个任务的数据集大小进行抽样。这相当于连接所有任务的数据集，并从合并后的数据集中随机抽样。然而，请注意，我们包括了无监督去噪任务，它使用的数据集比其他任务大几个数量级。由此可见，如果我们简单地根据每个数据集的大小进行抽样，模型看到的绝大多数数据将是无标签的，并且它将在所有监督任务上进行欠训练。即使没有无监督任务，一些任务(例如WMT英语到法语)非常大，它们同样会挤出大多数批次。为了解决这个问题，我们在计算比例之前人为地设置了数据集大小的“限制”。具体来说，如果我们每个N任务的数据集中的示例数量是$e_n, n \in \{1, \ldots, N\}$，那么我们将在训练期间从$m$ th任务中采样示例的概率设置为$r_m = \min(e_m, K)/\sum \min(e_n, K)$，其中$K$是人工数据集大小限制。

Temperature-scaled mixing 缓解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”。多语言BERT使用这种方法，以确保模型在低资源语言上得到充分的训练。[14] 为了实现温度缩放T，我们提高每个任务的混合率r_m的$1/T$次方，并重新归一化速率，使它们的总和为1。当$T = 1$时，这种方法相当于示例的比例混合，随着T的增加，比例变得更接近于平均混合。我们保留数据集大小限制K(用于在温度缩放之前获得r_m)，但将其设置为一个较大的值$K = 2^{21}$。我们使用一个较大的值K，因为升高温度会降低最大数据集的混合率。

Equal mixing 在这种情况下，我们以相同的概率从每个任务中采样。具体来说，每个批次中的每个示例都是从我们训练的一个数据集中均匀随机抽样的。这很可能是一种次优的策略，因为模型在低资源任务上很快会过拟合，而在高资源任务上则会过拟合。我们主要将其作为一个参考点，以说明当比例设置为次优时可能出现的问题。

为了将这些混合策略与预训练-然后微调的基线结果进行平等的比较，本文以相同的总步骤数训练多任务模型:$2^{19} + 2^{18} = 786{,}432$。结果显示在Table 11。

多任务训练的表现不如预训练，然后对大多数任务进行微调。“平等”混合策略尤其会导致性能显著下降，这可能是因为低资源任务存在过拟合，高资源任务没有看到足够的数

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

Mixing strategy	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline (pre-train/fine-tune)	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Equal	76.13	19.02	76.51	63.37	23.89	34.31	26.78
Examples-proportional, $K = 2^{16}$	80.45	19.04	77.25	69.95	24.35	34.99	27.10
Examples-proportional, $K = 2^{17}$	81.56	19.12	77.00	67.91	24.36	35.00	27.25
Examples-proportional, $K = 2^{18}$	81.67	19.07	78.17	67.94	24.57	35.19	27.39
Examples-proportional, $K = 2^{19}$	81.42	**19.24**	79.78	67.30	25.21	36.30	**27.76**
Examples-proportional, $K = 2^{20}$	80.80	**19.24**	**80.36**	67.38	25.66	36.93	**27.68**
Examples-proportional, $K = 2^{21}$	79.83	18.79	79.50	65.10	25.82	37.22	27.13
Temperature-scaled, $T = 2$	81.90	**19.28**	79.42	69.92	25.42	36.72	27.20
Temperature-scaled, $T = 4$	80.56	**19.22**	77.99	69.54	25.04	35.82	27.45
Temperature-scaled, $T = 8$	77.21	19.10	77.14	66.07	24.55	35.35	27.17

Table 11: 使用不同混合策略的多任务训练比较。样本比例混合是指根据每个数据集的总大小从每个数据集中采样样本，并对最大数据集大小设置人为限制(K)。温度比例混合通过温度重新缩放采样率T。对于温度比例混合，我们使用人工数据集的大小限制$K = 2^{21}$。

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。例如，比例混合，我们发现对于大多数任务K都有一个“甜蜜点”，在这个点上模型获得了最佳性能，K的较大或较小的值往往会导致性能较差。例外(对于我们考虑的K值范围)是WMT英语到法语的翻译，这是一个高资源的任务，它总是从更高的混合比例中受益。最后，我们注意到，温度比例混合还提供了一种从大多数任务中获得合理性能的方法，$T = 2$在大多数情况下表现最好。之前已经观察到，在每个单独任务上训练的单独模型优于多任务模型，例如Arivazhagan et al. (2019)和McCann et al. (2018)，尽管已表明多任务设置可以在非常相似的任务上带来好处Liu et al. (2019b); Ratner et al. (2018)。在下一节中，我们将探讨如何缩小多任务训练和“先训练后微调”方法之间的差距。

3.5.3 将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放松版本，在其中，我们在混合任务上训练单个模型，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，考虑模型一次性对所有任务进行预训练，然后对单个监督任务进行微调的情况。这是‘MT-DNN‘ (Liu et al., 2015, 2019b)使用的方法，当它被引入时，它在GLUE和其他基准上取得了最先进的性能。我们考虑这种方法的三种变体: 首先，我们简单地在样本比例混合上预训练模型，人工数据集大小限制为$K = 2^{19}$，然后在每个单独的下游任务上对其进行微调。这有助于我们衡量在预训练期间将监督任务与无监督目标结合在一起是否会使模型对下游任务有一些有益的早期接触。我们还可能希望，混合许多监督来源，可以帮助预训练模型在适应单个任务之前获得一组更通用的“技能”(松散地说)。为了直接测量这一点，我们考虑第二种变体，在相同的样本上预训练模型-比例混合($K = 2^{19}$)，只是我们省略了预训练混合中的一个下游任务。然后，对预训练期间遗漏的任务进行微调。我们对考虑的每个下游任务重复此操作。我们将这种方法称为“留一”多任务训练。这模拟了现实世界的环境，其中预训练模型对预训练期间未见过的任务进行微调。请注意，多任务预训练提供了监督任务的不同混合。由于其他领域(例如计算机视觉(Oquab et al., 2014; Jia et al., 2014; Huh et al., 2016; Yosinski et al., 2014))使用监督数据集进行预训练，我们有兴趣看看从多任务预训练混合中省略无监督任务是否仍然产生良好的结果。因此，对于第三种变体，我们

Training strategy	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Unsupervised pre-training + fine-tuning	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	39.82	27.65
Multi-task training	81.42	**19.24**	79.78	67.30	25.21	36.30	27.76
Multi-task pre-training + fine-tuning	**83.11**	**19.12**	**80.26**	**71.03**	**27.08**	39.80	**28.07**
Leave-one-out multi-task training	81.98	19.05	79.97	**71.68**	**26.93**	39.79	**27.87**
Supervised multi-task pre-training	79.93	18.96	77.38	65.36	26.81	**40.13**	**28.04**

Table 12: 无监督预训练、多任务学习和各种形式的多任务预训练的比较。

在$K = 2^{19}$考虑的所有监督任务的样本比例混合上进行预训练。在所有这些变体中，我们遵循对2^{19}步骤进行预训练的标准程序，然后对2^{18}步骤进行微调。

我们在Table 12上比较了这些方法的结果。为了进行比较，还包括基线(预训练然后微调)和标准多任务学习(没有微调)的结果，示例-比例混合$K = 2^{19}$。在多任务预训练后进行微调，其性能与基线相当。这表明，在多任务学习之后使用微调可以帮助缓解Section 3.5.2中描述的不同混合率之间的一些权衡。有趣的是，“留一”训练的表现仅略差，这表明在各种任务上训练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显著的任务干扰)。最后，监督式多任务预训练在除翻译任务外的所有情况下的性能显著下降。这可能表明翻译任务从(英语)预训练中获益较少，而无监督预训练是其他任务中的一个重要因素。

3.6 缩放

机器学习研究的“惨痛教训”认为，可以利用额外计算的通用方法最终战胜依赖人类专业知识的方法(Sutton, 2019; Hestness et al., 2017; Shazeer et al., 2017; Jozefowicz et al., 2016; Mahajan et al., 2018; Shazeer et al., 2018, 2017; Huang et al., 2018b; Keskar et al., 2019a)。最近的结果表明，这可能适用于NLP中的迁移学习(Liu et al., 2019c; Radford et al., 2019; Yang et al., 2019; Lan et al., 2019)，即反复表明，与更精心设计的方法相比，扩大规模可以产生更好的性能。然而，有多种可能的扩展方法，包括使用更大的模型，训练模型以进行更多步骤，以及集成。在本节中，我们通过解决以下前提来比较这些不同的方法:“您只是获得了4×更多的计算。你应该如何使用它?”

我们从基线模型开始，该模型具有220M参数，并分别针对2^{19}和2^{18}步骤进行了预训练和微调。编码器和解码器的大小都类似于“BERT $_{\text{BASE}}$ “。为了实验增加的模型大小，我们遵循‘BERT $_{\text{LARGE}}$ " Devlin et al. (2018)的指导方针，并使用$d_{\text{ff}} = 4096$, $d_{\text{model}} = 1024$, $d_{\text{kv}} = 64$和16 -头部注意力机制。然后，我们在编码器和解码器中分别生成两个变体16和32层，生成带有2×和4×的模型，其参数与原始模型一样多。这两种变体的计算成本大致为2×和4×。使用我们的基线和这两个更大的模型，我们考虑使用4×作为计算量的三种方法:为4×进行训练，为2×使用2×更大的模型进行训练，为4×更大的模型进行“”基线“训练步数。当我们增加训练步骤时，为了简单起见，我们同时扩展了预训练和微调步骤。请注意，当增加预训练步骤的数量时，我们有效地包括更多的预训练数据，因为C4是如此大，即使在为2^{23}步骤进行训练时，我们也没有完成一次数据传递。

让模型看到4×那么多数据的另一种方法是将批处理大小增加4的倍数。由于更有效的并行化，这可能会导致更快的训练。然而，使用4×更大的批量大小进行训练，可能会产生与使用4×(因为步骤多(Shallue et al., 2018))进行训练不同的结果。我们还包含了一个额外的实验，在该实验中，我们使用4×更大的批量大小来训练我们的基线模型，以比较这两种情况。

Scaling strategy	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline	83.28	19.24	80.88	71.36	26.98	39.82	27.65
1× size, 4× training steps	85.33	19.33	82.45	74.72	27.08	40.66	27.93
1× size, 4× batch size	84.60	19.42	82.52	74.64	27.07	40.60	27.84
2× size, 2× training steps	**86.18**	19.66	**84.18**	77.18	27.52	**41.03**	28.19
4× size, 1× training steps	**85.91**	19.73	**83.86**	**78.04**	27.47	40.71	28.10
4× ensembled	84.77	**20.10**	83.09	71.74	**28.05**	40.53	**28.57**
4× ensembled, fine-tune only	84.05	19.57	82.36	71.55	27.55	40.22	28.09

Table 13: 扩展基线模型的不同方法的比较。除了集成微调模型外，所有方法都使用4×计算作为基线。“Size ‘指的是模型中的参数数量，“training time ‘指的是用于预训练和微调的步数。

在我们考虑的许多基准上，通过使用一组模型进行训练和评估来获得额外的性能是常见的做法。这提供了一种使用额外计算的正交方法。为了将其他缩放方法与集成进行比较，本文还测量了4分别预训练和微调模型的集成的性能。在将它们输入到输出softmax非线性之前，我们对集合进行对数平均，以获得聚合预测。与预训练4单独的模型相比，更便宜的替代方法是采用单个预训练模型并生成4单独的微调版本。虽然这没有使用我们的整个4×计算预算，但我们还包括这个方法，以查看它是否能产生与其他扩展方法相竞争的性能。

应用这些不同的缩放方法后所取得的性能可以在Table 13中看到。不出所料，增加训练时间和/或模型大小会不断提高基线。在4×进行多步骤训练和使用4×更大的批量大小之间没有明显的赢家，尽管两者都是有益的。一般来说，与单纯增加训练时间或批量大小相比，增加模型大小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× as long的2×更大模型与训练4×更大模型之间的巨大差异。这表明，增加训练时间和增加模型大小可以是提高性能的互补手段。结果还表明，集成提供了一种正交的、有效的方法，通过尺度来提高性能。在一些任务中(CNN/DM, WMT英语到德语，WMT英语到罗马尼亚语)，4完全单独训练的模型集成明显优于其他所有扩展方法。联合预训练但单独微调的集成模型也比基线有了显著的性能提升，这表明这是一种更廉价的提高性能的方法。唯一的例外是SuperGLUE，其中两种集成方法都没有明显改善基线。

我们注意到，不同的扩展方法有不同的权衡，这与它们的性能是分开的。例如，使用更大的模型会使下游的微调和推理更昂贵。相比之下，如果将小模型预训练较长时间的成本应用于许多下游任务，则可以有效地分摊成本。另外，我们注意到，集成N单独的模型与使用具有N×更高计算成本的模型具有类似的成本。因此，在选择缩放方法时，对模型的最终使用进行一些考虑很重要。

3.7 把它们放在一起

现在，我们利用系统研究的见解来确定在流行的NLP基准上可以将性能提高到什么程度。我们还对通过在大量数据上训练更大的模型来探索NLP迁移学习目前的局限性感兴趣。我们从我们的基线训练方法开始，并进行以下更改:

Objective 我们将基线中的i.i.d.去噪目标替换为Section 3.3.4中描述的span-corruption目标，该目标大致受到SpanBERT (Joshi et al., 2019)的启发。具体来说，我们使用

原始序列的平均跨度长度3和腐败15%。我们发现，该目标产生了略微更好的性能(Table 7)，同时由于目标序列长度较短，计算效率略高。

Longer training 我们的基线模型使用了相对少量的预训练(¼和BERT (Devlin et al., 2018)一样多，1⁄16和XLNet (Yang et al., 2019)一样多，1⁄64和RoBERTa (Liu et al., 2019c)一样多，等等)。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时间(这可能是有害的，如Section 3.4.2所示)。我们在Section 3.6中发现额外的预训练确实是有帮助的，增加批量大小和增加训练步骤的数量都可以带来这种好处。因此，在长度为512的批次大小2^{11}序列上对模型进行1百万步的预训练，对应于总共约1万亿预训练token(约为我们的基线的32×多)。在Section 3.4.1中，我们证明了在RealNews-like、WebText-like和Wikipedia + TBC数据集上的预训练在一些下游任务上优于C4的预训练。然而，这些数据集变体足够小，在对1万亿token的预训练过程中会重复数百次。由于我们在Section 3.4.2中表明这种重复可能是有害的，因此我们选择继续使用C4数据集。

Model sizes 在Section 3.6中，我们还展示了如何扩大基线模型大小来提高性能。然而，在可用于微调或推理的计算资源有限的情况下，使用较小的模型可能是有帮助的。基于这些因素，我们训练了各种规模的模型:

- **基地。**这是我们的基线模型，其超参数在Section 3.1.1中描述。它大约有220万个参数。
- **很小。**我们考虑一个更小的模型，它通过使用$d_{\text{model}} = 512$, $d_{\text{ff}} = 2{,}048$, 8 -头部注意力，和编码器和解码器中每个仅6层来缩小基线。这个变体大约有60百万参数。
- **大号。**由于我们的基线使用BERT $_{\text{BASE}}$大小的编码器和解码器，我们还考虑编码器和解码器在大小和结构上都类似于BERT $_{\text{LARGE}}$的变体。具体来说，该变体在编码器和解码器中分别使用$d_{\text{model}} = 1{,}024$、$d_{\text{ff}} = 4{,}096$、$d_{\text{kv}} = 64$、16 -头部注意力和24层，从而产生大约770万个参数。
- **3B和11B。**为了进一步探索使用更大的模型时可能的性能，我们考虑两个额外的变体。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为首的注意力，这导致大约2.8亿个参数;对于'11B '，我们使用$d_{\text{ff}} = 65{,}536$和128为首的注意力产生一个大约11亿参数的模型。我们选择扩大d_{ff}是因为现代加速器(如我们训练模型的tpu)对于大型密集矩阵乘法(如Transformer的前馈网络中的矩阵乘法)是最有效的。

Multi-task pre-training 在Section 3.5.3中，我们展示了在微调之前对无监督和有监督任务的多任务混合进行预训练，以及单独对无监督任务进行预训练。这就是"MT-DNN"(Liu et al., 2015, 2019b)提倡的方法。它还有一个实际的好处，即能够在整个训练期间监控"下游"的性能，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。本文假设训练时间较长的较大模型可能从较大比例的未标记数据中受益，因为它们更有可能过拟合较小的训练数据集。然而，我们也注意到Section 3.5.3的结果表明，在多任务预训练后进行微调，可以缓解因选择次优比例的未标记数据而可能出现的一些问题。基于这些想法，我们在使用标准示例-比例混合(在Section 3.5.2中描述)之前，为我们的未标记数据替换以下人工数据集大小:710,000表示小数据，2,620,000表示基本数据，8,660,000表示大数

据，33,500,000表示3B, 133,000,000表示11B。对于所有模型变体，我们还在预训练期间限制了WMT英语到法语数据集和WMT英语到德语数据集到1M示例的有效数据集大小。

Fine-tuning on individual GLUE and SuperGLUE tasks 到目前为止，在对GLUE和SuperGLUE进行微调时，我们已经将每个基准中的所有数据集连接起来，因此我们只对GLUE和SuperGLUE分别进行一次微调。这种方法使我们的研究在逻辑上更简单，但我们发现，与单独对任务进行微调相比，这牺牲了一些任务的性能。对单个任务进行微调的一个潜在问题是，我们可能会对低资源任务快速过拟合，否则将通过一次性对所有任务进行训练来缓解这个问题。例如，对于许多低资源GLUE和SuperGLUE任务，我们的大批量2^{11}长度- 512序列将导致整个数据集在每个批次中出现多次。因此，在对每个GLUE和SuperGLUE任务进行微调期间，我们使用8长度- 512序列的较小批量大小。我们还在每个1,000步骤而不是每个5,000步骤中保存检查点，以确保我们在模型过拟合之前能够访问模型的参数。

Beam search 我们之前的所有结果都是使用贪婪解码报告的。对于具有长输出序列的任务，我们发现使用束搜索(Sutskever et al., 2014)提高了性能。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$\alpha = 0.6$(Wu et al., 2016)的长度惩罚。

Test set 由于这是我们的最终实验集，我们报告测试集上的结果，而不是验证集。对于CNN/Daily Mail，我们使用与数据集一起分布的标准测试集。对于WMT任务，这对应于英语-德语使用`newstest2014`，英语-法语使用`newstest2015`，英语-罗马尼亚语使用`newstest2016`。对于GLUE和SuperGLUE，我们使用基准评估服务器来计算官方测试集的分数。[15,16] 对于SQuAD来说，在测试集上进行评估需要在基准服务器上运行推理。不幸的是，这个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。因此，我们继续报告在阵容验证集上的表现。幸运的是，在SQuAD测试集上表现最好的模型也报告了在验证集上的结果，因此我们仍然可以与表面上最先进的模型进行比较。

除了上述变化之外，我们使用相同的训练过程和超参数作为基线(adfactor优化器、用于预训练的逆平方根学习率调度、用于微调的恒定学习率、dropout正则化、词汇表等)。这些细节描述在Section 2中，以供参考。

最后一组实验的结果显示在Table 14。总的来说，在我们考虑的24任务中，我们在18上实现了最先进的性能。正如预期的那样，我们最大的(11十亿参数)模型在所有任务的模型大小变体中表现最好。我们的T5-3B模型变体在一些任务中确实击败了之前的技术水平，但将模型大小扩展到11亿参数是实现最佳性能的最重要因素。现在我们来分析每个基准测试的结果。

我们达到了最先进的平均胶水分数90.3。值得注意的是，在自然语言推理任务MNLI、RTE和WNLI方面，性能大大优于之前的最先进水平。RTE和WNLI是机器性能历史上落后于人类性能的两个任务，分别是93.6和95.9(Wang et al., 2018)。在参数数量方面，我们的11B模型变体是提交到GLUE基准的最大模型。然而，大多数得分最高的提交都使用了大量的集成和计算来产生预测。例如，性能最好的ALBERT (Lan et al., 2019)版本使用的模型大小和架构与我们的3B版本相似(尽管由于巧妙的参数共享，它的参数大大减少)。为了在胶水上产生令

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

Model	GLUE Average	CoLA Matthew's	SST-2 Accuracy	MRPC F1	MRPC Accuracy	STS-B Pearson	STS-B Spearman
Previous best	89.4[a]	69.2[b]	97.1[a]	**93.6**[b]	**91.5**[b]	92.7[b]	92.3[b]
T5-Small	77.4	41.0	91.8	89.7	86.6	85.6	85.0
T5-Base	82.7	51.1	95.2	90.7	87.5	89.4	88.6
T5-Large	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**

Model	QQP F1	QQP Accuracy	MNLI-m Accuracy	MNLI-mm Accuracy	QNLI Accuracy	RTE Accuracy	WNLI Accuracy
Previous best	74.8[c]	**90.7**[b]	91.3[a]	91.0[a]	**99.2**[a]	89.2[a]	91.8[a]
T5-Small	70.0	88.0	82.4	82.3	90.3	69.9	69.2
T5-Base	72.6	89.4	87.1	86.2	93.7	80.1	78.8
T5-Large	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**

Model	SQuAD EM	SQuAD F1	SuperGLUE Average	BoolQ Accuracy	CB F1	CB Accuracy	COPA Accuracy
Previous best	90.1[a]	95.5[a]	84.6[d]	87.1[d]	90.5[d]	95.2[d]	90.6[d]
T5-Small	79.10	87.24	63.3	76.4	56.9	81.6	46.0
T5-Base	85.44	92.08	76.2	81.4	86.2	94.0	71.2
T5-Large	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**

Model	MultiRC F1a	MultiRC EM	ReCoRD F1	ReCoRD Accuracy	RTE Accuracy	WiC Accuracy	WSC Accuracy
Previous best	84.4[d]	52.5[d]	90.6[d]	90.0[d]	88.2[d]	69.9[d]	89.0[d]
T5-Small	69.3	26.3	56.3	55.4	73.3	66.9	70.5
T5-Base	79.7	43.1	75.0	74.2	81.5	68.3	80.8
T5-Large	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**

Model	WMT EnDe BLEU	WMT EnFr BLEU	WMT EnRo BLEU	CNN/DM ROUGE-1	CNN/DM ROUGE-2	CNN/DM ROUGE-L
Previous best	**33.8**[e]	**43.8**[e]	**38.5**[f]	43.47[g]	20.30[g]	40.63[g]
T5-Small	26.7	36.0	26.8	41.12	19.56	38.35
T5-Base	30.9	41.2	28.0	42.05	20.34	39.40
T5-Large	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**

Table 14: 我们的T5变体在我们研究的每个任务上的表现。Small、Base、Large、3B和11B指的是模型配置，参数分别为60 million、220 million、770 million、3 billion和11 billion。在每个表的第一行，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，并在本文标题末尾列出了参考文献。除了使用验证集的SQuAD之外，所有结果都报告在测试集上。[a](Lan et al., 2019) [b](Wang et al., 2019c) [c](Zhu et al., 2019) [d](Liu et al., 2019c) [e](Edunov et al., 2018) [f](Lample and Conneau, 2019) [g](Dong et al., 2019)

人印象深刻的性能，ALBERT作者根据任务将6到17英寸的模型组合在一起。这可能导致用ALBERT集合进行预测的计算成本比用T5-11B更高。

在阵容方面，我们在精确的比赛得分上比之前的最先进(阿尔伯特(Lan et al., 2019))高出一分以上。SQuAD是一个创建于三年前的长期基准，最近的大多数改进只是将最先进的性能提高了一个百分点的零头。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(例如TriviaQA (Joshi et al., 2017)或NewsQA (Trischler et al., 2016))来增强小型小队训练集。人类在阵容上的表现分别在82.30和91.22上估算精确匹配和F1指标(Rajpurkar et al., 2016)，因此目前尚不清楚在这个基准上的进一步改进是否有意义。

对于SuperGLUE，我们在最先进的基础上进行了很大的改进(从平均分数84.6(Liu et al., 2019c)到88.9)。SuperGLUE设计的任务“超出了目前最先进系统的范围，但大多数受过大学教育的英语使用者都可以解决”(Wang et al., 2019b)。我们几乎匹配人类的表现89.8(Wang et al., 2019b)。有趣的是，在阅读理解任务(MultiRC和ReCoRD)上，我们大大超过了人类的表现，这表明用于这些任务的评估指标可能偏向于机器制造的预测。另一方面，人类在COPA和WSC上都达到了100%的精度，明显优于我们的模型的性能。这表明，仍然存在我们的模型难以完善的语言任务，特别是在低资源环境下。

我们没有在任何WMT翻译任务上取得最先进的性能。这可能部分是因为我们使用的是只有英语的未标记数据集。我们还注意到，这些任务上的大多数最佳结果使用反向翻译(Edunov et al., 2018; Lample and Conneau, 2019)，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还使用了其他形式的跨语言无监督训练(Lample and Conneau, 2019)。我们的结果表明，规模和英语预训练可能不足以匹配这些更复杂的方法的性能。更具体地说，`newstest2014`上的最佳结果使用的是来自WMT 2018 (Edunov et al., 2018)的更大的训练集，这使得很难直接与我们的结果进行比较。

最后，在CNN/Daily Mail上，我们取得了最先进的性能，尽管只在ROUGE-2-F分数上有很大的提高。已经表明，对ROUGE分数的改进不一定对应于更连贯的摘要(Paulus et al., 2017)。此外，虽然CNN/Daily Mail被作为一个抽象的摘要基准，但纯提取方法已被证明是有效的(Liu, 2019)。也有人认为，用最大似然训练的生成模型容易产生重复的摘要(See et al., 2017)。尽管存在这些潜在问题，但发现所提出模型确实生成了连贯且基本正确的摘要。我们在Appendix C中提供了一些非精选的验证集示例。

为了取得强大的结果，T5将我们实验研究的见解与前所未有的规模相结合。请注意，在Section 3.6中，我们发现扩大基线模型的预训练量或大小会产生实质性的收益。鉴于此，我们有兴趣测量我们引入到T5中的“非扩展”更改对其强大性能的贡献。因此，我们进行了最后一个实验，比较了以下三种配置: 首先，标准基线模型，在$2^{35} \approx 34$B token上进行预训练; 其次，基线训练了大约1万亿代币(即用于T5的相同数量的预训练)，我们称之为“基线- 1t”; 第三，T5-Base。请注意，baseline-1T和T5- base之间的差异包括我们在设计T5时所做的“非扩展”更改。因此，比较这两个模型的性能，可以具体衡量系统研究的见解的影响。

这三种模型配置的性能在Table 15中显示。与Section 3.6中的发现一致，我们发现额外的预训练比基线提高了性能。然而，T5-Base在所有下游任务上都大大优于基线- 1t。这表明规模并不是T5成功的唯一因素。本文假设较大的模型不仅受益于其规模的增加，还受益于这些非缩放因子。

Model	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline	83.28	19.24	80.88	71.36	26.98	39.82	27.65
Baseline-1T	84.80	19.62	83.01	73.90	27.46	40.30	28.34
T5-Base	**85.97**	**20.90**	**85.44**	**75.64**	**28.37**	**41.37**	**28.98**

Table 15: T5-Base与本文其余部分使用的基线实验设置的性能比较。结果报告在验证集上。“ baseline - 1t ‘是指通过在1万亿token(与T5模型变体使用的相同数字)上预训练基线模型而不是$2^{35} \approx 34$B token(与基线使用的相同)来实现的性能。

4. 反射

在完成我们的系统研究后，我们首先总结一些最重要的发现。我们的结果提供了一些高层次的视角，说明哪些研究途径可能更有希望或不太有希望。最后，概述了一些我们认为可能为该领域的进一步发展提供有效方法的主题。

4.1 外卖

Text-to-text 所提出的文本到文本框架提供了一种简单的方法，使用相同的损失函数和解码过程在各种文本任务上训练单个模型。展示了这种方法如何成功应用于生成式任务，如抽象摘要，分类任务，如自然语言推理，甚至是回归任务，如STS-B。尽管简单，但我们发现文本到文本框架获得了与特定任务架构相当的性能，并在结合scale时最终产生了最先进的结果。

Architectures 虽然一些关于NLP迁移学习的工作考虑了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到文本框架中效果最好。尽管编码器-解码器模型使用的参数是“仅编码器”(例如BERT)或“仅解码器”(语言模型)架构的两倍，但它具有相似的计算成本。我们还发现，在编码器和解码器中共享参数并没有导致参数总数减半的情况下性能的显著下降。

Unsupervised objectives 总的来说，我们发现大多数"去噪"目标(训练模型以重建随机损坏的文本)在文本到文本的设置中表现类似。本文建议使用产生短目标序列的目标，以使无监督预训练在计算上更有效。

Data sets 我们介绍了“巨大的干净爬取语料库”(C4)，它包括从常见爬取web转储中启发式清理的文本。当将C4与使用额外过滤的数据集进行比较时，我们发现对域内未标记数据进行训练可以提高一些下游任务的性能。然而，限制在单一域通常会导致较小的数据集。我们分别表明，当未标记的数据集足够小，以至于在预训练过程中多次重复时，性能会下降。这激发了对C4这样的大型多样化数据集的使用，以完成通用语言理解任务。

Training strategies 我们发现，在微调期间更新所有预训练模型参数的基本方法，性能优于旨在更新较少参数的方法，尽管更新所有参数是最昂贵的。我们还尝试了各种方法来同时在多个任务上训练模型，在我们的文本到文本设置中，这只是对应于在构建批次时混合来自不同数据集的示例。在多任务学习中，主要关注的是设置每个任务的训练比例。最终，我们没有找到一种设置混合比例的策略，与无监督预训练然后有监

督微调的基本方法的性能相匹配。然而，我们发现，在对混合任务进行预训练后进行微调，产生了与无监督预训练相当的性能。

Scaling 我们比较了利用额外计算的各种策略，包括在更多的数据上训练模型、训练更大的模型和使用集成模型。我们发现每种方法都显著提高了性能，尽管在更多的数据上训练一个较小的模型往往比用更少的步骤训练一个较大的模型要好。模型的集合可以提供比单个模型更好的结果，单个模型提供了利用额外计算的正交方法。使用相同的基本预训练模型进行微调的集成模型，其性能比完全单独预训练和微调所有模型的性能要差，尽管仅进行微调的集成仍然大大优于单个模型。

Pushing the limits 我们结合上述见解并训练了更大的模型(多达11亿参数)，以在我们考虑的许多基准上实现最先进的结果。对于无监督训练，我们从C4数据集中提取文本，并应用了一个破坏连续标记区间的去噪目标。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。总的来说，我们的模型是在超过1万亿token上训练的。为了促进结果的复制、扩展和应用，发布了代码、C4数据集和每个T5变体的预训练模型权重。[1]

4.2 展望

The inconvenience of large models 我们研究的一个不奇怪但很重要的结果是，越大的模型往往表现得越好。用于运行这些模型的硬件不断变得更便宜和更强大的事实表明，扩大规模可能仍然是实现更好性能的一种有希望的方式(Sutton, 2019)。然而，在某些应用程序和场景中，使用更小或更便宜的模型是有帮助的，例如在执行客户端推理或联邦学习(Konečnỳ et al., 2015, 2016)时。与此相关，迁移学习的一个有益用途是有可能在低资源任务上取得良好的性能。根据定义，低资源任务经常发生在缺乏标记更多数据的资源的环境中。因此，低资源应用程序通常也限制了对计算资源的访问，这可能会导致额外的成本。因此，我们主张研究用更便宜的模型实现更强性能的方法，以便将迁移学习应用于其影响最大的地方。目前在这方面的一些工作包括蒸馏(Hinton et al., 2015; Sanh et al., 2019; Jiao et al., 2019)、参数共享(Lan et al., 2019)和条件计算(Shazeer et al., 2017)。

More efficient knowledge extraction 回想一下，预训练的目标之一(松散地说)是为模型提供通用的“知识”，以提高其在下游任务上的性能。我们在这项工作中使用的方法，也是目前常见的做法，是训练模型来对损坏的文本范围进行去噪。我们怀疑这种简单的技术可能不是教授模型通用知识的非常有效的方法。更具体地说，能够获得良好的微调性能，而不需要首先在1万亿标记的文本上训练我们的模型，这将是有用的。遵循这些思路的一些并发工作通过预训练模型来区分真实文本和机器生成文本(Clark et al., 2020)来提高效率。

Formalizing the similarity between tasks 我们观察到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(Section 3.4)。这一发现主要依赖于基本的观察，例如SQuAD是使用维基百科的数据创建的。在预训练和下游任务之间制定一个更严格的“相似性”概念将是有用的，这样我们就可以对使用什么未标记数据来源做出更有原则的选择。在计算机视觉领域中有一些早期的经验工作(Huh et al., 2016; Kornblith et al., 2018; He et al., 2018)。更好地理解任务的相关性也可以帮助选择有监督的预训练任务，这已被证明对GLUE基准是有帮助的(Phang et al., 2018)。

Language-agnostic models 我们失望地发现，纯英语预训练并没有在我们研究的翻译任务上取得最先进的结果。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与语言无关的模型，即可以在给定的NLP任务中，无论文本是什么语言，都有良好性能的模型。鉴于英语不是世界上大多数人的母语，这是一个特别相关的问题。

本文的动机是最近关于NLP迁移学习的一系列工作。在我们开始这项工作之前，这些进展已经在基于学习的方法尚未被证明有效的环境中取得了突破。我们很高兴能够继续这种趋势，例如，通过在SuperGLUE基准上接近人类水平的性能，该任务专为现代迁移学习管道的困难而设计。该结果源于一个直接和统一的文本到文本框架、新的C4数据集和系统研究的见解的结合。此外，我们提供了该领域的经验概述和对其现状的看法。我们很高兴看到继续使用迁移学习实现通用语言理解的目标。

Acknowledgments

We thank Grady Simon, Noah Fiedel, Samuel R. Bowman, Augustus Odena, Daphne Ippolito, Noah Constant, Orhan Firat, Ankur Bapna, and Sebastian Ruder for their comments on this manuscript; Zak Stone and the TFRC team for their support; Austin Tarango for his guidance on data set creation; Melvin Johnson, Dima Lepikhin, Katrin Tomanek, Jeff Klingner, and Naveen Arivazhagan for insight into multi-task machine translation; Neil Houlsby for comments on adapter layers; Olga Wichowska, Ola Spyra, Michael Banfield, Yi Lin, and Frank Chen for assistance with infrastructure; Etienne Pot, Ryan Sepassi, and Pierre Ruyssen for collaboration on TensorFlow Datasets; Rohan Anil for help with our download pipeline for Common Crawl; Robby Neale and Taku Kudo for their work on SentencePiece; and many other members of the Google Brain team for their discussion and insight.

Appendix A. 贡献

Colin设计了这个项目的范围并写了这篇论文，在Sections 3.1 to 3.6上运行了所有的实验，并贡献了我们的大部分代码。Noam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;并在Section 3.7上进行了实验。Adam监督了这个项目的所有工程方面，创建了C4数据集，实现了我们的数据集管道，并添加了各种基准数据集。Katherine协调实验，编写和更新文档，运行实验来帮助设计基线，并为我们的代码库的许多部分做出了贡献。Sharan贡献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，还共同领导了我们的代码库的开源。Michael拥有Winograd数据集的所有方面，采集了我们使用的许多数据集，对我们的基础设施做出了各种改进和修复，并运行了一些初步实验。Yanqi进行了实验并实现了一些方法，以帮助确定一个合理的基线，并帮助在Section 3.7中对模型进行最终的微调。Wei还帮助我们对一些预处理器进行了最后的微调和改进。Peter制作了一个预训练数据集的早期版本，并解决了与球队和CNN/DM任务有关的问题。所有作者都帮助确定了我们在这项工作中遵循的范围和研究方向。

Appendix B. 将WNLI转换为我们的文本到文本格式

请注意，正如在Section 2.4中讨论的那样，我们不使用任何来自WNLI的数据进行训练。相反，当对WNLI测试集进行评估时(结果见Section 3.7)，我们将WNLI测试集转换为“参照名词预测”文本到文本的格式，以便我们可以使用在WSC和DPR上训练的模型进行评估。我们的WNLI预处理器的灵感来自He et al. (2019)提出的预处理器。回想一下，WNLI中的例子由一个前提、一个假设和一个标签组成，标签表示假设是真还是假。使用Section 2.4的例子，假设是“市议员拒绝向示威者发放许可证，因为他们害怕暴力。“前提是‘示威者害怕暴力。标签是假的。我们首先找到前提中所有代词的位置(在我们的例子中是“他们”)。然后，我们找到假设(在我们的例子中是“害怕暴力”)中每个代词之前或之后的最大单词数，忽略大小写和标点符号。当前提包含多个候选代词时，我们选择假设的最大子串之前或之后的代词。然后，我们用星号把前提中的代词标出来。对于候选名词(与我们的模型预测获得的真或假标签进行比较)，我们从假设中删除匹配的子字符串，并有选择地使其非所有格(导致” the demonstrative ”)。

Appendix C. CNN/每日邮报的预测示例

为了表明我们的模型正在生成流畅的摘要，我们在验证集上包括了一些来自最佳模型(T5-11B)的示例解码，以及基本事实摘要。这些例子是随机选择的，不是精心挑选的。

1. **真实情况:**南非克鲁格国家公园里，美洲豹在一种尖锐的生物拒绝让步后放弃了战斗。41岁的lisl moolman是野生动物爱好者，她带着相机在路上拍下了这场奇怪的打斗。这只豹子和豪猪从灌木丛中滚了出来，开始在路边打斗，莫尔曼女士就在一旁看着。

   预测:在南非克鲁格国家公园，一只豹子试图用一只勇敢的豪猪做午餐。但在这只尖刺动物拒绝在战斗中后退后，捕食者被牢牢地放在了它的位置上。41岁的lisl moolman是野生动物爱好者，她带着相机在路上拍下了这场奇怪的打斗。

2. **基本事实:**研究人员说，住宅是未发现物种的温床。对北卡罗来纳州40个家庭的研究发现了8000种细菌和古细菌。在加州的11栋房子里，我们发现了各种各样的真菌。

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，几乎相当于德克萨斯州的面积。一项对北卡罗来纳州40座房屋的研究使用基因组技术记录了超过8000种细菌和古细菌分类。另一项对同一州50座房屋的研究描述了超过750种节肢动物，包括昆虫和蜘蛛。对加州11所房子的检查发现，我们中间有各种各样的真菌。

3. **事实真相:**周一晚上，在注射死刑的药物出现问题后，该州暂停执行死刑。凯利·吉森达纳第二次推迟了她的死亡时间，因为她发现致命的戊巴比妥剂量出现了混浊。州政府官员对这个决定犹豫不决，三次改变主意，最后决定不冒这个行不通的风险。乔治亚州司法部长奥伦斯(Sam olens)说，执行死刑必须符合宪法。

   预测:乔治亚州死囚区唯一的女性凯利·吉森达纳(kelly gissendaner)周一第二次被推迟执行死刑。执行小组发现，致命的戊巴比妥剂量出现浑浊。这种混浊的药物支持了反对死刑的人，在该国其他地区发生了三起拙劣的死刑处决后，他们大声疾呼反对死刑。

4. **事实:**丹尼·阿尔维斯没有入选巴西队对阵法国和智利的大名单。巴塞罗那后卫被拍到周六给人们提供热狗。本周，阿尔维斯与前队友何塞·平托发行了一张慈善单曲。阿尔维斯似乎准备在今年夏天自由转会离开巴塞罗那。西班牙媒体报道称他将签约巴黎圣日耳曼。

   预测:丹尼·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和何塞·平托一起发行了一张慈善单曲。周六，阿尔维斯被拍到在一家热狗店的柜台后端着零食。阿尔维斯还没有在诺坎普得到一份新合同。点击这里获取所有最新的巴塞罗那新闻。

Appendix D. 预处理示例

在本节中，我们将举例说明如何对所考虑的每个数据集进行预处理。

D.1 可乐

Original input: Sentence: 约翰使比尔成为主人。

Processed input: 约翰使比尔成为自己的主人。

Original target: 1

Processed target: 可接受

D.2 rte

Original input: Sentence 1: 南斯拉夫的意大利人在斯洛文尼亚定居的比例较小(在1991年的全国人口普查中，大约3000斯洛文尼亚居民宣布自己是意大利人)。

Sentence 2: 斯洛文尼亚有3000名居民。

Processed input: 一小部分南斯拉夫的意大利人定居在斯洛文尼亚(在1991年的全国人口普查中，大约3000斯洛文尼亚居民宣布自己是意大利人)。句子2:斯洛文尼亚有3000名居民。

Original target: 1

Processed target: 非蕴含

D.3 mnli

Original input: Hypothesis: 圣路易斯红雀队总是赢。

Premise: 是的，失败是，我的意思是，我来自圣路易斯和圣路易斯红雀队当时他们是一支失败的球队，但是

Processed input: mnli假设:圣路易斯红雀队总是获胜。前提:是的，失败是，我的意思是，我来自圣路易斯和圣路易红雀队当时他们基本上是一支失败的球队，但是

Original target: 2

Processed target: 矛盾

D.4 MRPC

Original input: Sentence 1: 我们之所以采取行动，是因为我们通过9月11日的经历，以新的眼光看待现有的证据。"

Sentence 2: 相反，美国之所以采取行动，是因为政府"通过我们在9 / 11事件中的经历，以新的视角看待现有证据"。

Processed input: 我们之所以采取行动，是因为我们通过9月11日的经验，从新的角度看到了现有的证据。"句子2:相反，美国采取行动是因为政府"通过我们在9·11事件中的经历，以新的视角看待现有证据"。

Original target: 1

Processed target: 等效的

D.5 qnli

Original input: Question: 杰比死在哪里?

Sentence: 成吉思汗很快将苏布台召回蒙古，杰别死在了回撒马尔罕的路上。

Processed input: Jebe死在哪里?句子:成吉思汗不久将苏布台召回蒙古，杰别死在回撒马尔罕的路上。

Original target: 0

Processed target: 蕴涵

D.6 QQP

Original input: Question 1: 在古罗马，什么样的品质会让你备受青睐?

Question 2: 我是如何以一名新生的身份进入IT公司的?

Processed input: 问题1:在古罗马，什么样的品质会让你很受欢迎?问题2:作为一名新生，我如何获得进入IT公司的机会?

Original target: 0

Processed target: not__duplicate

D.7 sst2

Original input: Sentence: 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。

Processed input: 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。

Original target: 1

Processed target: 积极的

D.8 STSB

Original input: Sentence 1: 记者周三未能立即联系到Puretunes的代表置评。

Sentence 2: 记者周四无法找到Puretunes的代表就此事发表评论。

Processed input: 周三，记者未能立即联系到纯粹音乐的代表就此置评。周四，记者无法找到Puretunes的代表就该诉讼发表评论。

Original target: 3.25

Processed target: 3.2

D.9 cb

Original input: Hypothesis: Valence很有帮助

Premise: 对空虚的大脑，对善良的男仆。为什么费格就不能选择他自己的解剖部位呢?他觉得自己在帮忙吗?

Processed input: cb假说:效价是帮助前提:效价大脑空虚，效价贤惠的男仆。为什么费格就不能选择他自己的解剖部位呢?他觉得自己在帮忙吗?

Original target: 1

Processed target: 矛盾

D.10 copa

Original input: Question: 效应

Premise: 这个国家发生了政治暴力事件。

Choice 1: 许多市民搬到了国会大厦。

Choice 2: 许多公民在其他地区避难。

Processed input: 选择一:许多市民搬到了国会大厦。选择2:许多公民在其他地区避难。前提:政治暴力在这个国家爆发。问题:效果

Original target: 1

Processed target: 正确

D.11 多rc

Original input: Answer: 只有馅饼可以吃，而不是传统的早餐食品

Paragraph: 很久以前，有一只松鼠叫Joey。例如:乔伊喜欢出去和他的表兄吉米玩。例如:乔伊和吉米在一起做愚蠢的游戏，总是笑个不停。有一天，乔伊和吉米一起去朱莉阿姨的池塘游泳。在他们离开之前，乔伊一大早就起来吃了点东西。送了6 :他除了派找不到任何吃的!通常，乔伊早餐会吃麦片、水果(梨)或燕麦片。他吃完饭后，他和Jimmy去了池塘。在他们去那里的路上，他们看到了他们的朋友兔子杰克。他们跳入水中，游了几个小时。送11:太阳出来了，但风很冷。例如:乔伊和吉米从水里出来，开始往家走。他们的皮毛是湿的，微风使他们感到寒冷。当他们回到家，把衣服擦干后，吉米穿上了他最喜欢的紫色衬衫。送 15:Joey穿上了一件红绿圆点的蓝色衬衫。送16:两只松鼠吃了乔伊妈妈茉莉做的食物，然后去睡觉了

Question: 为什么乔伊早上醒来吃早餐时很惊讶?

Processed input: 乔伊早上起床吃早餐时为什么会感到惊讶?答案:早餐只有馅饼吃，而不是传统的早餐食物从前，有一只叫乔伊的松鼠。例如:乔伊喜欢出去和他的表兄吉米玩。例如:乔伊和吉米在一起做愚蠢的游戏，总是笑个不停。有一天，乔伊和吉米一起去朱莉阿姨的池塘游泳。在他们离开之前，乔伊一大早就起来吃了点东西。送了6 :他除了派找不到任何吃的!通常，乔伊早餐会吃麦片、水果(梨)或燕麦片。他吃完饭后，他和Jimmy去了池塘。在他们去那里的路上，他们看到了他们的朋友兔子杰克。他们跳入水中，游了几个小时。送11:太阳出来了，但风很冷。例如:乔伊和吉米从水里出来，开始往家走。他们的皮毛是湿的，微风使他们感到寒冷。当他们回到家，把衣服擦干后，吉米穿上了他最喜欢的紫色衬衫。送 15:Joey穿上了一件红绿圆点的蓝色衬衫。送16:两只松鼠吃了乔伊妈妈茉莉做的食物，然后去睡觉了

Original target: 1

Processed target: 正确

D.12 WiC

Original input: POS: n

Sentence 1: 他的行为是故意的，这是对他的侮辱。

Sentence 2: 陪审团的审议

Word: 深思熟虑

Processed input: 他的行为是故意的，这是侮辱。句子2:陪审团的审议。单词:审议

Original target: 0

Processed target: 假的

D.13 WSC和DPR

Original input: Span 2 text: 它

Span 1 text: 稳定的

Span 2 index: 20.

Span 1 index: 1

Text: 马厩很宽敞，有四个不错的马厩;一扇大窗户朝院子里开，使院子通风宜人。

Processed input: 马厩很宽敞，有四个不错的隔间;一扇大窗户向院子敞开，使院子通风宜人。

Original target: 1

Processed target: 稳定的

D.14 CNN/每日邮报

Original input: 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，而且是最好的伙伴。在周三对阵纽卡斯尔的比赛之前，这对曼联和比利时双雄都在周一晚上晒出了自己在一家餐馆就餐的照片。贾努扎伊在费莱尼和一个朋友中间摆姿势，看起来像是没能收到关于这是杰克逊五兄弟主题之夜的通知。英超双雄阿德南·贾努扎伊和费莱尼在舞池上与朋友合影。曼联和比利时双雄费莱尼和贾努扎伊在球场内外都是好朋友。曼联王牌费莱尼跑到替补席上，和他的朋友贾努扎伊一起庆祝他在对阵QPR比赛中的进球。背景中的迪斯科效果也印证了这一理论，但贾努扎伊似乎并不介意，因为他们后来与其他朋友在舞池中摆姿势。曼联本赛季没有太多的理由来唱歌和跳舞，所以他们可能会去迪斯科舞厅作为另一种形式的释放。然而，周三对纽卡斯尔的胜利至少会让主教练范加尔感到高兴，他们将继续为本赛季的欧冠席位而战。贾努扎伊和罗宾·范佩西和费莱尼一起在西布朗的曼联球迷面前庆祝。贾努扎伊从曼联的荷兰籍主帅范加尔那里得到了一些智慧之言。贾努扎伊和费莱尼在一些朋友的陪伴下来到舞池，迎接纽卡斯尔的比赛。

Processed input: 总结:费莱尼和贾努扎伊继续向世界展示他们不仅是队友，也是最好的伙伴。在周三对阵纽卡斯尔的比赛之前，这对曼联和比利时双雄都在周一晚上晒出了自己在一家餐馆就餐的照片。贾努扎伊在费莱尼和一个朋友中间摆姿势，看起来像是没能收到关于这是杰克逊五兄弟主题之夜的通知。英超双雄阿德南·贾努扎伊和费莱尼在舞池上与朋友合影。曼联和比利时双雄费莱尼和贾努扎伊在球场内外都是好朋友。曼联王牌费莱尼跑到替补席上，和他的朋友贾努扎伊一起庆祝他在对阵QPR比赛中的进球。背景中的迪斯科效果也印证了这一理论，但贾努扎伊似乎并不介意，因为他们后来与其他朋友在舞池中摆姿势。曼联本赛季没有太多的理由来唱歌和跳舞，所以他们可能会去迪斯科舞厅作为另一种形式的释放。然而，周三对纽卡斯尔的胜利至少会让主教练范加尔感到高兴，他们将继续为本赛季的欧冠席位而战。贾努扎伊和罗宾·范佩西和费莱尼一

起在西布朗的曼联球迷面前庆祝。贾努扎伊从曼联的荷兰籍主帅范加尔那里得到了一些智慧之言。贾努扎伊和费莱尼在一些朋友的陪伴下来到舞池，迎接纽卡斯尔的比赛。

Original target: 周一晚上，这对比利时双人组合和一些朋友一起来到舞池。曼联将在周三对阵纽卡斯尔。红魔期待的是7年来的第2场联赛客场胜利。范加尔的球队目前领先利物浦2分，排名第四。

Processed target: 周一晚上，这对比利时双人组合和一些朋友一起来到舞池。曼联将在周三对阵纽卡斯尔。红魔期待的是7年来的第2场联赛客场胜利。范加尔的球队目前领先利物浦2分，排名第四。

D.15 小队

Original input: Question: 患者肺部氧浓度升高会导致什么移位?

Context: 高压氧药物使用特殊的氧气室来增加0 2 .在病人周围，必要时，医护人员。一氧化碳中毒，气体坏疽和减压病("减压病")有时使用这些设备治疗。0增加肺中的浓度有助于将一氧化碳从血红蛋白中的血红素组中转移出来。氧气对引起气性坏疽的厌氧细菌是有毒的，所以增加氧气的分压有助于杀死它们。减压病发生在潜水后过快减压的潜水员身上，导致血液中形成惰性气体气泡，主要是氮气和氦气。增加0的压强2 .尽快是治疗的一部分。

Processed input: 问题:患者肺部氧浓度升高会导致什么移位?背景:高压氧医学使用特殊的氧气室来增加0 2 .在病人周围，必要时，医护人员。一氧化碳中毒，气体坏疽和减压病("减压病")有时使用这些设备治疗。0增加肺中的浓度有助于将一氧化碳从血红蛋白中的血红素组中转移出来。氧气对引起气性坏疽的厌氧细菌是有毒的，所以增加氧气的分压有助于杀死它们。减压病发生在潜水后过快减压的潜水员身上，导致血液中形成惰性气体气泡，主要是氮气和氦气。增加0的压强2 .尽快是治疗的一部分。

Original target: 一氧化碳

Processed target: 一氧化碳

D.16 从英语到德语

Original input: "路易吉经常对我说，他不希望这对兄弟上法庭，"她写道。

Processed input: 她写道:"路易吉经常对我说，他不希望这对兄弟上法庭。"

Original target: "Luigi sagte oft zu mir, dass er nie wollte, dass die Brüder vor Gericht landen", schrieb sie。

Processed target: "Luigi sagte oft zu mir, dass er nie wollte, dass die Brüder vor Gericht landen", schrieb sie。

D.17 英语转法语

Original input: 斯皮策望远镜的红外记录图像显示了无数代恒星的"全家福":最古老的恒星被视为蓝色的点，而更难识别的是恒星产房里粉红色的"新生儿"。

Processed input: 这张来自斯皮策望远镜红外记录的图像显示了无数代恒星的"全家福":最古老的恒星被视为蓝色的点，而更难识别的是恒星产房里粉红色的"新生儿"。

Original target: Ce détail d'une摄影基础设施奖télescope斯皮策·蒙特雷拍摄的《无名者的肖像》générations d'étoiles: les + vieilles étoiles蓝与les点玫瑰，加难辨词à标识符，sont les "nouveau-nés" dans la salle d'accouchement de l'univers。

Processed target: Ce détail d'une摄影基础设施奖télescope斯皮策·蒙特雷拍摄的《无名者的肖像》générations d'étoiles: les + vieilles étoiles蓝与les点玫瑰，加难辨词à标识符，sont les "nouveau-nés" dans la salle d'accouchement de l'univers。

D.18 WMT英语到罗马尼亚语

Original input: 塔可钟表示，计划到2022年在美国增加2000家门店。

Processed input: 塔可钟表示，计划到2022年在美国增加2000家门店。

Original target: 塔可钟a afirmat că, până în 2022, intenționează să deschidă 2000 de restaurante în SUA。

Processed target: 塔可钟a afirmat că, până în 2022, intenționează să deschidă 2000 de restaurante în SUA。

Appendix E. 所有实验中每个任务的分数

下表列出了Sections 3.2 to 3.6中描述的实验中每个任务的得分。

		GLUE													CNN/DM			SQuAD		SuperGLUE											WMT		
Table	Experiment	Score Average	CoLA MCC	SST-2 Acc	MRPC F1	MRPC Acc	STSB PCC	STSB SCC	QQP F1	QQP Acc	MNLI_m Acc	MNLI_mm Acc	QNLI Acc	RTE Acc	R-1-F	R-2-F	R-L-F	EM	F1	Score Average	BoolQ Acc	CB F1	CB Acc	COPA Acc	MultiRC F1	MultiRC EM	ReCoRD F1	ReCoRD EM	RTE Acc	WiC Acc	WSC Acc	EnDe BLEU	EnFr BLEU	EnRo BLEU
1	★Baseline average	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	Baseline standard deviation	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	No pre-training	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	★Enc/dec, denoising	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	Enc/dec, shared, denoising	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	Enc/dec, 6 layers, denoising	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	Language model, denoising	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	Prefix LM, denoising	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	Enc/dec, shared, 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	Enc/dec, 6 layers, 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	Language model, 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	Prefix 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	Language modeling with prefix	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-style (Devlin et al., 2018)	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	Deshuffling	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-style (Devlin et al., 2018)	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 et al., 2019)	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	★Replace corrupted spans	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	Drop corrupted tokens	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	Corruption rate = 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	★Corruption rate = 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	Corruption rate = 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	Corruption rate = 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	★Baseline (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	Average span length = 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	Average span length = 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	Average span length = 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	Average span length = 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, unfiltered	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	RealNews-like	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	WebText-like	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	Wikipedia	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	Wikipedia + TBC	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	★Full data set	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 repeats)	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 repeats)	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} (1,024 repeats)	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 repeats)	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	★All parameters	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	Adapter layers, $d = 32$	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	Adapter layers, $d = 128$	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	Adapter layers, $d = 512$	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	Adapter layers, $d = 2048$	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	Gradual Unfreezing	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	★Baseline (pre-train/fine-tune)	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	Equal	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	Examples-proportional, $K = 2^{16}$	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	Examples-proportional, $K = 2^{17}$	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	Examples-proportional, $K = 2^{18}$	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	Examples-proportional, $K = 2^{19}$	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	Examples-proportional, $K = 2^{20}$	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	Examples-proportional, $K = 2^{21}$	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	Temperature-scaled, $T = 2$	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	Temperature-scaled, $T = 4$	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	Temperature-scaled, $T = 8$	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	★Unsupervised pre-training + fine-tuning	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	Multi-task training	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	Multi-task pre-training + fine-tuning	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	Leave-one-out multi-task training	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	Supervised multi-task pre-training	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	★Baseline	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× size, 4× training steps	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× size, 4× batch size	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× size, 2× training steps	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× size, 1× training steps	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× ensembled	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× ensembled, fine-tune only	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

Table 16: 在本文的所有实验中，我们考虑的每个任务的得分。在第一列中，我们列出了一个表，其中给出了给定实验的浓缩结果。与正文一样，标记为★的行表示我们的基线模型(在Section 3.1中描述)。

References

Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. Character-level language modeling with deeper self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

Rohan Anil, Vineet Gupta, Tomer Koren, and Yoram Singer. Memory-efficient adaptive optimization for large-scale learning. *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, et al. Massively multilingual neural machine translation in the wild: Findings and challenges. *arXiv preprint arXiv:1907.05019*, 2019.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *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, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Third International Conference on Learning Representations*, 2015.

Ankur Bapna, Naveen Arivazhagan, and Orhan Firat. Simple, scalable adaptation for neural machine translation. *arXiv preprint arXiv:1909.08478*, 2019.

Iz Beltagy, Kyle Lo, and Arman Cohan. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, et al. Findings of the 2014 workshop on statistical machine translation. 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, et al. Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, 2015.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, et al. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, 2016.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

Christian Buck, Kenneth Heafield, and Bas Van Ooyen. N-gram counts and language models from the common crawl. In *LREC*, 2014.

Rich Caruana. Multitask learning. *Machine learning*, 28(1), 1997.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and 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 and 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. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*, 2017.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, 2005.

Andrew M. Dai and Quoc V. Le. Semi-supervised sequence learning. In *Advances in neural information processing systems*, 2015.

Marie-Catherine De Marneff, Mandy Simons, and Judith Tonhauser. The CommitmentBank: Investigating projection in naturally occurring discourse. In *Sinn und Bedeutung 23*, 2019.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-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, and David Grangier. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*, 2018.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. Learning word vectors for 157 languages. *arXiv preprint arXiv:1802.06893*, 2018.

Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Ivan Habernal, Omnia Zayed, and Iryna Gurevych. C4Corpus: Multilingual web-size corpus with free license. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 914–922, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking ImageNet pre-training. *arXiv preprint arXiv:1811.08883*, 2018.

Pengcheng He, Xiaodong Liu, Weizhu Chen, and Jianfeng Gao. A hybrid neural network model for commonsense reasoning. *arXiv preprint arXiv:1907.11983*, 2019.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, 2015.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. Learning distributed representations of sentences from unlabelled data. *arXiv preprint arXiv:1602.03483*, 2016.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. *arXiv preprint arXiv:1902.00751*, 2019.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *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, and Douglas Eck. Music transformer: Generating music with long-term structure. In *Seventh International Conference on Learning Representations*, 2018a.

Yanping Huang, Yonglong Cheng, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, and Zhifeng Chen. GPipe: Efficient training of giant neural networks using pipeline parallelism. *arXiv preprint arXiv:1811.06965*, 2018b.

Minyoung Huh, Pulkit Agrawal, and Alexei A. Efros. What makes ImageNet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.

Shankar Iyer, Nikhil Dandekar, and Kornel Csernai. First Quora dataset release: Question pairs. https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs, 2017.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, 2014.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and 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. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. CTRL: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019a.

Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. Unifying question answering and text classification via span extraction. *arXiv preprint arXiv:1904.09286*, 2019b.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.

Ryan Kiros, Yukun Zhu, Ruslan R. Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, 2015.

Vid Kocijan, Ana-Maria Cretu, Oana-Maria Camburu, Yordan Yordanov, and Thomas Lukasiewicz. A surprisingly robust trick for Winograd schema challenge. *arXiv preprint arXiv:1905.06290*, 2019.

Jakub Konečnỳ, Brendan McMahan, and Daniel Ramage. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*, 2015.

Jakub Konečnỳ, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do better ImageNet models transfer better? *arXiv preprint arXiv:1805.08974*, 2018.

Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*, 2018.

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*, 2019.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

Hector Levesque, Ernest Davis, and Leora Morgenstern. The Winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2012.

Qi Li. Literature survey: domain adaptation algorithms for natural language processing. 2012.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text summarization branches out*, 2004.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating Wikipedia by summarizing long sequences. *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. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*, 2019b.

Yang Liu. Fine-tune BERT for extractive summarization. *arXiv preprint arXiv:1903.10318*, 2019.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019c.

Lajanugen Logeswaran and Honglak Lee. An efficient framework for learning sentence representations. *arXiv preprint arXiv:1803.02893*, 2018.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*, 2018.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 2013b.

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. *arXiv preprint arXiv:1602.06023*, 2016.

Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002.

Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014.

Matthew Peters, Sebastian Ruder, and Noah A. Smith. To tune or not to tune? adapting pretrained representations to diverse tasks. *arXiv preprint arXiv:1903.05987*, 2019.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

Jason Phang, Thibault Févry, and Samuel R. Bowman. Sentence encoders on STILTs: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*, 2018.

Mohammad Taher Pilehvar and Jose Camacho-Collados. WIC: 10,000 example pairs for evaluating context-sensitive representations. *arXiv preprint arXiv:1808.09121*, 2018.

Matt Post. A call for clarity in reporting BLEU scores. *arXiv preprint arXiv:1804.08771*, 2018.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019.

Altaf Rahman and Vincent Ng. Resolving complex cases of definite pronouns: the Winograd schema challenge. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, 2012.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Prajit Ramachandran, Peter J. Liu, and Quoc V. Le. Unsupervised pretraining for sequence to sequence learning. *arXiv preprint arXiv:1611.02683*, 2016.

Alex Ratner, Braden Hancock, Jared Dunnmon, Roger Goldman, and Christopher Ré. Snorkel MeTaL: Weak supervision for multi-task learning. In *Proceedings of the Second Workshop on Data Management for End-To-End Machine Learning*, 2018.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI Spring Symposium Series*, 2011.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Sebastian Ruder. *Neural transfer learning for natural language processing*. PhD thesis, NUI Galway, 2019.

Sebastian Ruder, Matthew E. Peters, Swabha Swayamdipta, and Thomas Wolf. Transfer learning in natural language processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 15–18, 2019.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *International journal of computer vision*, 2015.

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

Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*, 2017.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

Christopher J Shallue, Jaehoon Lee, Joe Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E. Dahl. Measuring the effects of data parallelism on neural network training. *arXiv preprint arXiv:1811.03600*, 2018.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. *arXiv preprint arXiv:1804.04235*, 2018.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyoukJoong Lee, Mingsheng Hong, Cliff Young, Ryan Sepassi, and Blake Hechtman. Mesh-tensorflow: Deep learning for supercomputers. In *Advances in Neural Information Processing Systems*, 2018.

Jason R. Smith, Herve Saint-Amand, Magdalena Plamada, Philipp Koehn, Chris Callison-Burch, and Adam Lopez. Dirt cheap web-scale parallel text from the common crawl. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013.

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, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 2014.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J. Pal. Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079*, 2018.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 2014.

Richard S. Sutton. The bitter lesson. http://www.incompleteideas.net/IncIdeas/BitterLesson.html, 2019.

Wilson L. Taylor. "Cloze procedure": A new tool for measuring readability. *Journalism Bulletin*, 1953.

Trieu H. Trinh and Quoc V. Le. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*, 2018.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 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, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, 2017.

Elena Voita, Rico Sennrich, and Ivan Titov. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. *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. Can you tell me how to get past Sesame Street? Sentence-level pretraining beyond language modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 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.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Liwei Peng, and Luo Si. StructBERT: Incorporating language structures into pre-training for deep language understanding. *arXiv preprint arXiv:1908.04577*, 2019c.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*, 2018.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1989.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, 2014.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. QAnet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*, 2018.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. Defending against neural fake news. *arXiv preprint 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, and Jingjing Liu. Freelb: Enhanced adversarial training for language understanding. *arXiv preprint arXiv:1909.11764*, 2019.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, 2015.