目 录

基于 Bert 模型的金融新闻分类模型

摘 要

参考现有金融新闻的数据集，新闻文本存在常见的长度分布不均衡问题，且由于金融行业的特殊性，与其相关的文本所含专业术语繁杂，语义晦涩难懂。想要从中挑选出对金融领域的业务开展过程中有帮助的新闻则需要对其进行分类，但使用人工进行标注的方式则需要涉及相关领域的从业人员，代价昂贵且效率低下。本文基于在自然语言处理各个任务取得突出成绩的 Bert 模型，将其改造使之能够应用于金融新闻的抽取式摘要处理任务，提取出文本中的核心内容，以减少错误以及无关信息对新闻标注工作的影响。然后本文采用了主流的无监督机器学习方法 LDA 主题模型和 K-means 聚类算法，分别来对摘要后的新闻文本做分类工作，来更好的帮助金融领域的从业工作者甄别错综复杂的新闻信息。

关键词：金融新闻；Bert 模型；抽取式摘要；机器学习

ABSTRACT

Referring to the existing dataset of financial news, news texts not only have the common problems of unbalanced length distribution and high content consistency, but also contain complex professional terms and obscure semantics due to the particularity of the financial industry, which makes it extremely difficult to obtain key information from them. If we want to select the news that is helpful to the business development in the financial field, we need to label the texts. However, it is expensive and inefficient to use manual method to label the financial corpus, which requires practitioners in related fields. Based on the Bert model which has achieved outstanding results in various tasks of natural language processing, this paper rebuilds its encoder to be able to be applied to the extraction summary processing task of financial news. Meanwhile, this paper tries different classification layers, such as linear classifier, Transformer decoder and RNN model, to extract the core content of the text in order to reduce the impact of errors and irrelevant information on the news annotation work. Then this paper uses two kinds of mainstream unsupervised machine learning method to classify the text after the summary respectively, one is the LDA topic model, which divides news text into multiple topics and marks the text according to the best theme; the other is K-means algorithm, which clusters news into different text groups, and some relevant indicators are used to evaluate the clustering results. Hope this work can help financial practitioners to screen the intricate news information.

Keyword: Financial news; Bert model; Abstract; Machine learning

第 1 章 绪论

1.1 机器学习在自然语言处理中的作用

1.1.1 自然语言处理

自然语言处理(NLP)是计算机程序在口语和书面语中理解人类语言的能力，被称为自然语言[1]。它是人工智能(AI)的组成部分。自然语言处理已经存在了 50 多年，它起源于语言学领域。它在许多领域都有各种实际应用，包括医学研究、搜索引擎和商业智能。

1.1.2 机器学习

机器学习是一种自动建立分析模型的数据分析方法。它是人工智能的一个分支，其理念是，系统可以从数据中学习，识别模式，并在最少的人工干预下做出决定。机器学习的概念几乎无处不在，如医疗保健、金融、基础设施、市场营销、自动驾驶汽车、推荐系统、聊天机器人、社交网站、游戏、网络安全等等。

1.1.3 应用

用于自然语言处理和文本分析的机器学习涉及一套统计方法，用于识别文本的词性、实体、情感等其他方面。这些技术可以表示为一个模型，然后应用于其他文本，也称为监督机器学习。当然它也可能是一套跨大数据集提取含义的算法，这被称为无监督机器学习。

在有监督的机器学习中，一批文本文档被标记或注释，而计算机根据标注进行学习，这些文档被用来“训练”一个统计模型，然后给出未标记的文本进行分析。现在最流行的有监督 NLP 机器学习算法包括贝叶斯网络，最大熵，条件随机场，神经网络/深度学习等等。而无监督机器学习包括在没有预先标记或注释的情况下训练模型，这些方法一般都通常易懂。

1.2 文本摘要与分类

1.2.1 文本摘要

摘要是对长篇作品(如论文或研究论文)的简短总结。摘要能够简明扼要报告研究的目的和结果，可以帮助读者快速了解论文或文章的主旨或精髓，以便决定是否阅读全文，同时一篇摘要可以帮助读者记住你论文中的要点。在大规模的文本数据库中，文章的摘要能够帮助过滤大量无关与错误信息，节约时间与经济成本。

1.2.2 文本分类

由于文本数据集的混乱性质，对文本数据进行分析、理解、组织和排序是困难且耗时的，很多存在于文本内部的信息未能被有效的利用起来，这就是使用机器学习进行文本分类的原因。使用文本分类器，可以以一种快速而经济的方式自动在各种文本数据库中构建相关文本类群，例如电子邮件、法律文件、社交媒体、聊天机器人、调查文档，从而节省分析文本数据的时间，自动化业务流程，并做出数据驱动的业务决策[2]。

1.3 本文的主要工作

由于金融行业的特殊性，与其相关的文本所含专业术语繁杂，语义晦涩难懂。想要从中挑选出对金融领域的业务开展过程中有帮助的新闻则需要对其进行分类，但使用人工进行标注的方式则需要涉及相关领域的从业人员，代价昂贵且效率低下。故本文采用了文本摘要加文本分类的方法来对金融数据集进行文本标注工作，均采用了机器学习的方法，前者通过有监督学习实现文本摘要，具体的方法是通过现有 Bert 预训练模型与不同的摘要分类层，提取出关键信息，降低分类成本并高提高准确率，后者则通过无监督学习来进行文本分类，并使用一定的评价指标进行文本分类结果的评估，最终达到将金融新闻数据集中的关键信息有效的利用起来的效果，以用于金融领域的业务开展工作。

1.4 论文的组织结构

本文将在第 2 章中详细探讨如何将 Bert 模型应用于文章摘要中，同时使用不同的摘要层进行评估，并采用效果最好的模型进行摘要工作；在第 3，4 章将分别采用 LDA 主题模型以及 K-means 聚类算法进行摘要的分类工作，同时采用相关的评价指标进行效果评估，并完成最终文本标注的工作；在第 5 章中将对本文所做工作进行总结，并进行一定的展望。

第 2 章 微调 Bert 模型用于抽取式摘要

2.1 单文档摘要任务

单文档摘要的任务是在保留原始文本最重要信息的同时能够在长度上有一定的压缩。这项任务由于在各种信息领域中有着广泛的应用而在自然语言处理界有着比较重要的地位(例如，新闻、社交、媒体、评论等)。而一般这个任务通常分为两种模式，分别为抽取式摘要与抽象式摘要，在抽象摘要中，生成的摘要不包含原始文本中已经存在的词句，通常要通过文本的重写操作来生成，而在抽取式摘要则通过复制和连接原始文档中最重要的词句来形成摘要，而在本文中最终采用的方法是抽取式文档摘要。

2.2 Bert 模型

2.2.1 模型简介

BERT 是一个用于自然语言处理(NLP)的开放源码机器学习框架[3]。BERT 设计用于帮助计算机理解文本中模糊语言的含义，并通过使用周围的文本建立上下文语义。BERT 框架使用了来自维基百科的文本进行了预训练，可以通过问答数据集进行微调。该模型一经推出就引在各大自然语言处理任务中取得了非常优异的成绩。

2.2.2 模型框架

BERT 利用了 Transformer[4]，这是一种学习文本中单词或子单词之间上下文关系的注意机制。在它的普通形式中，Transformer 包括两个独立的机制:读取本输入的编码器和生成任务预测的解码器。但由于 BERT 的目标是生成语言模型，因此只需要编码器机制。Transformer 是一个在自然语言领域得到广泛应用的神经网络学习框架，其模型的结构如下所示：

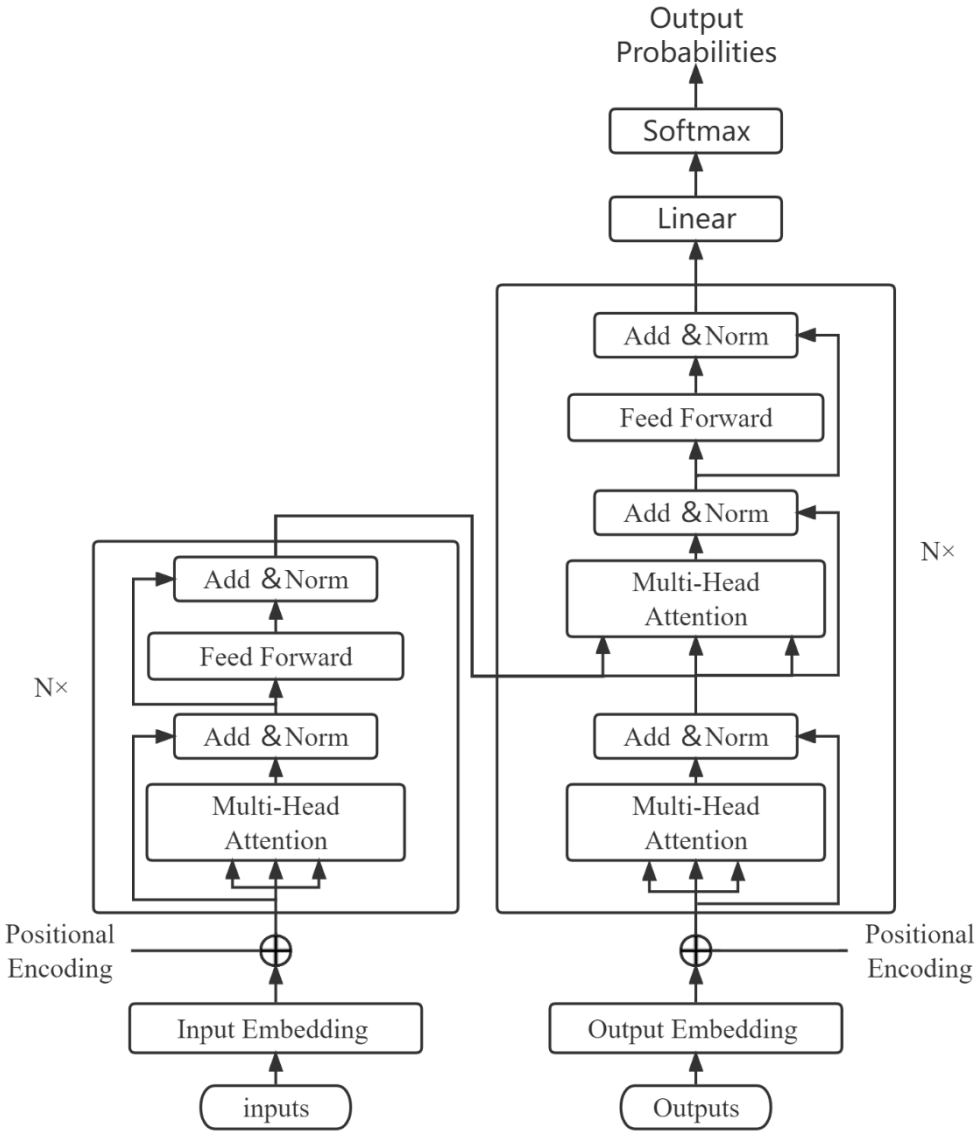

图 2-1 Transformer 编码器-解码器结构图

BERT 是用了编码器侧的网络，与按顺序(从左到右或从右到左)读取文本输入的方向模型相反，Transformer 编码器一次读取整个单词序列。因此，可以认为它是双向的，但更准确地说，它是无方向性的。而通过这个特性允许模型根据单词周围的环境(单词的左边和右边)来学习单词的上下文。Bert 整体网络结构如下图所示，其中 Trm 表示一层编码器网络。最后 BERT 的输出会基于所有层中的左右两侧语义：

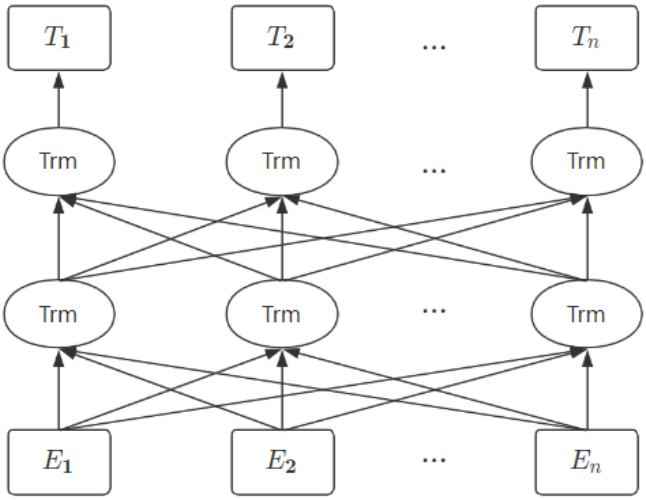

图 2-2 Bert 结构图

2.2.3 模型输入表示

BERT 输入的编码向量是由 3 个部分的嵌入和组成，下面来分别介绍其具体含义：

1.标记嵌入（Token Embedding）：采用了 WordPiece 进行分词，这种算法在基本词汇上训练语言模型，选取可能性最高的一对，将这一对添加到词汇中，在新的词汇上训练语言模型，重复上述步骤，直到达到所需的词汇量或可能性阈值。

2.位置嵌入（Position Embedding）：位置嵌入是 0 到 512 之间每个可能位置的学习向量。Tranformer 不像递归神经网络那样具有序列性质，因此需要输入的顺序信息;如果忽略这一点，输出将是排列不变的。

3.分割嵌入（Segment Embedding）：分段嵌入基本上是被编码到向量中的句子序号。模型必须知道一个特定的单词是属于 BERT 中的句子 A 还是句子 B。而分割嵌入就是通过生成不同的固定标记来实现这种区分的。在分割嵌入层中只有两个向量表示。属于输入 1 的所有标记被分配给第一个向量(索引 0)，而属于输入 2 的所有标记被分配给第二个向量(索引 1)。

图中的有两个特殊的符号[CLS]和[SEP]，前者放在句子之前，后者用于区分下一个句子，以用于断开输入中的两个句子。BERT 使用[CLS]句子 A[SEP]句子 B[SEP]格式进行预训练。在下一个句子预测任务中需要区分两个句子，

所以这两个特殊符号还是非常重要的。中间的[SEP]则帮助模型理解哪个单词属于哪个句子。在进行下游任务微调的时候，如果使用不同于训练前格式的格式，可能会混淆模型。模型不知道有两句话，就会把它看成是一句话。

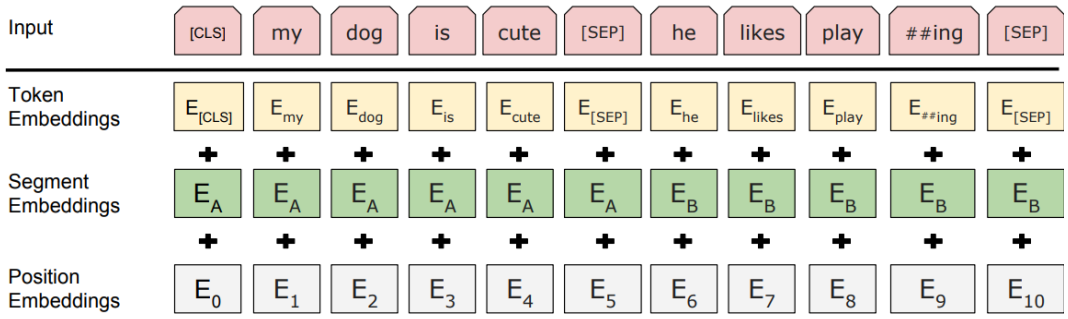

图 2-3 Bert 模型的输入向量空间

2.2.4 模型的预训练任务

BERT 在两个不同但相关的 NLP 自监督任务上被预先训练:屏蔽语言建模和下一个句子预测任务，MLM 训练的目的是在句子中隐藏一个词，然后让程序根据这个词的上下文来预测被隐藏的词。“下一个句子预测”训练的目标是让程序预测两个给定的句子是否有逻辑的、顺序的联系，或者它们的关系是否只是随机的。下面对它们的实现分别进行简单的介绍：

1.掩码预测任务(Masked Language Model)：BERT 是一个自编码语言模型，即预测时同时从两个方向阅读序列。在一个屏蔽语言建模任务中，对于给定的输入序列，随机屏蔽掉 15%的单词，通过预测这些屏蔽的单词从而来对模型进行训练。为了做到这一点，我们的模型以两个方向读入序列然后尝试预测屏蔽的单词。

2.句子预测任务(Next Sentence Prediction)：NSP 是二分类任务，在此任务中，我们输入两个句子，然后 BERT 需要判断第二个句子是否为第一个句子的下一句。通过运行 NSP 任务，我们的模型可以理解两个句子之间的关系，这会有利于很多下游任务，像问答和文本生成。

通过使用两个任务的数据集合来执行不同的预训练任务训练 Bert 模型，获得预训练好的模型后，这样对于特定的下游自然语言任务可以进行适当的微调，将其应用于自己的任务中。

2.3 微调 Bert 进行文章抽取式摘要

2.3.1 总体思路

为了能够使用 BERT 进行抽取式摘要任务，就必须要求它能够得到每个句子的向量表示，但由于 BERT 被预训练为掩码语言模型，输出的向量是基于单词而不是基于句子。同时，尽管 BERT 的输入中有分割嵌入，但是由于在原始模型中仅存在着两个标签（句子 A 和句子 B），不适用于提取摘要任务中需要识别多个句子的需求。本文将从修改输入的序列以及嵌入的角度来解决这些问题从而使得 BERT 模型能够进行文章摘要。其整体的模型结构图如下所示。

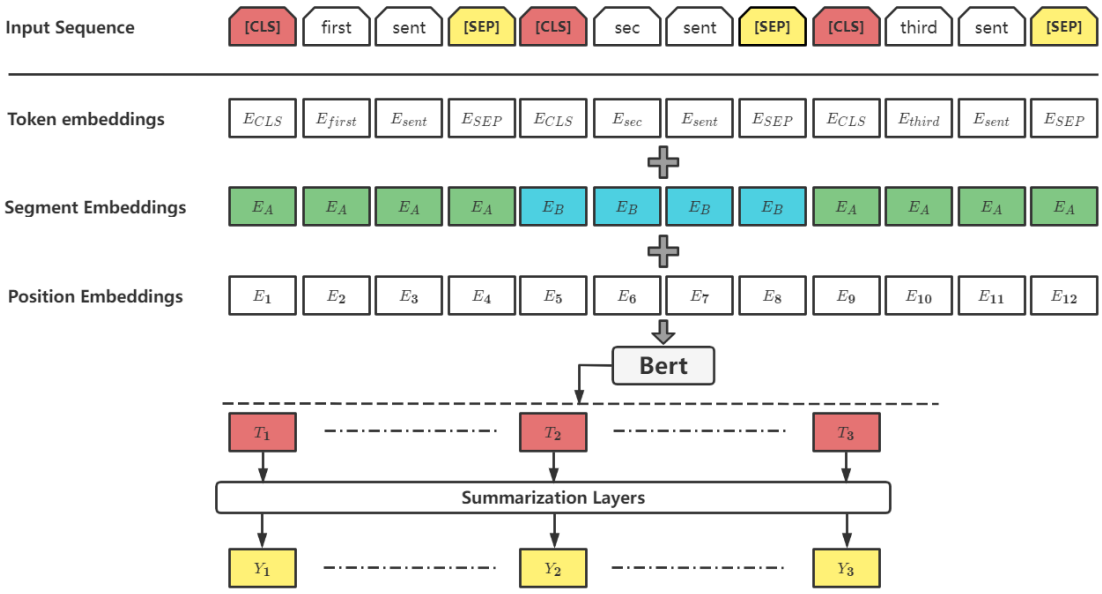

图 2-4 用于文章摘要任务的整体模型框架图

2.3.2 编码多个句子

在上图中的输入序列中有三个句子，在每个句子前面分别添加了一个[CLS]标记，同时在每个句子的后面添加了一个[SEP]标记。在原始的 Bert 中，[CLS]标记注意了所有的单词，聚合了整个句子或者句子对的特征，因此在 NLP 中，可以看成是整个句子的句向量。

通过使用多个[CLS]标记来对模型进行一定的修改，这样最后第 i 个[CLS]标志通过 Bert 模型输出的向量T_i就用来作为第 i 个句子的向量表示，在图中采

用红色作为对应关系。同时为了保证输入序列的长度一致，在最后输出的文档向量中，需要进行向量的填充工作。

2.3.3 间隔分割嵌入

与掩码模型不同，抽取式摘要需要区分文章中的多个句子，故使用了一种间隔分割嵌入的方法，具体做法是对于特定的句子 i，通过基于 i 是奇数还是偶数来分配给句子 i 一个不同的分割嵌入 Embedding A 或者 Embedding B。例如，对于$[\text{sen } t_1, \text{sen } t_2, \text{sen } t_3, \text{sen } t_4, \text{sen } t_5]$，将分配的分割嵌入是[$EA$，$EB$，$EA$，$EB$，$EA$]。

2.4 构建摘要判断层

从句子编码层获取文档中每个句子的句向量后，在输出上构建了 3 种特定于摘要的判断层，以通过获取每个句子在文档级特征下的重要性。对于每个句$sent_i$，计算出最终的预测分数$\hat{Y}_i$，模型的损失是$\hat{Y}_i$相对于标签Y_i的二元交叉熵。

2.4.1 简单的分类器

在 Bert 的原论文中，简单的分类器仅仅在 Bert 的输出后面添加了一个线性的全连接层，然后对计算的结果使用激活函数 Sigmoid 计算每个句子的预测分数，公式如下所示：

$$\hat{Y}_i = \sigma(W_o T_i + b_o) \tag{1}$$

其中σ表示 Sigmoid 函数。

2.4.2 Transformer 编码器

与简单的 Sigmoid 分类器不同，通过在对 Bert 输出的句子向量上使用一定数量的 Transformer 编码层以通过句子间的语义关系来提取文档级的特征信息，以更好的完成抽取式摘要任务。计算公式如下：

$$\tilde{h}^l = \mathrm{LN}(h^{l-1} + \mathrm{MHAtt}(h^{l-1})) \tag{2}$$

$$h^l = \mathrm{LN}\left(\tilde{h}^l + \mathrm{FFN}\left(\tilde{h}^l\right)\right) \tag{3}$$

其中$h^0 = \mathrm{PosEmb}\ (T)$，$T$为经过 Bert 输出的句向量；PosEmb 是一个把位置嵌入与 T 进行相加的函数，以更好的利用句子的位置信息；LN 是层标准化操作[5]，可以提高模型的训练速度和精度,使得模型更加稳健；MHAtt 是对嵌入进行多头注意力操作,可以帮助网络捕捉更丰富的特征与信息；上标 l 是对应的 encoder 的层数。具体的做法见图以及上文对 Transformer 的描述。

当然最后层的输出还是经过一个 Sigmoid 分类器进行句子分数的预测：

$$\hat{Y}_i = \sigma(W_o h_i^L + b_o) \tag{4}$$

其中h_i^L是句$sent_i$通过第 L 层的输出，而 L 则是我们最终使用 Transformer 编码层的总层数。在最后实验中我们分别令 L=1, 2, 3 来进行模型的训练。

2.4.3 循环神经网络

尽管 Transformer 模型在很多 NLP 任务中均取得了不错的效果，但是还是有很多实验表明循环神经网络在很多方面有它独特的优势[6]。因此在得到 Bert 模型的句向量输出后，来通过添加一个 LSTM 层学习特定于摘要任务的特征[7]。LSTM 神经网络的结构图如下所示：

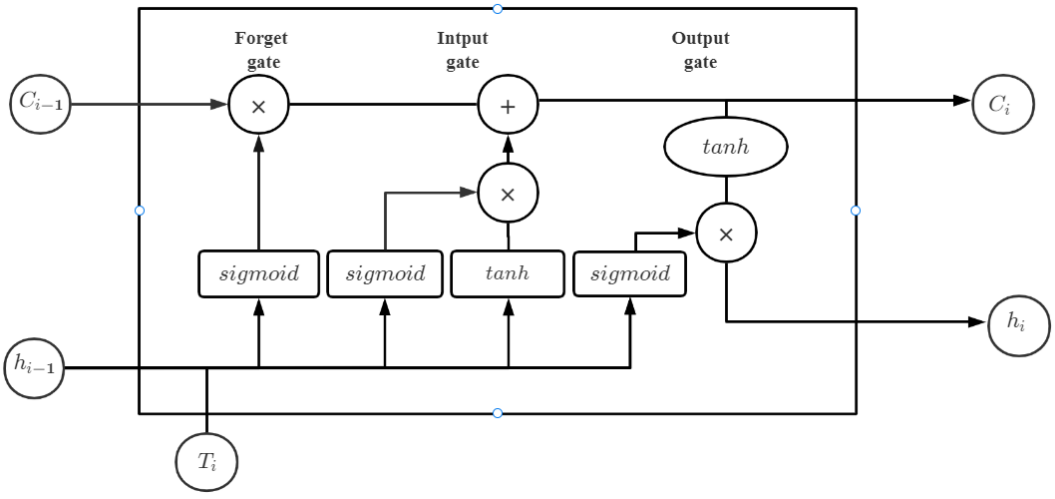

图 2-5 LSTM 神经网络结构图

在时间步 i，LSTM 层的输入是 BERT 输出T_i，而输出的计算公式如下所示：

$$\begin{pmatrix} F_i \\ I_i \\ O_i \\ G_i \end{pmatrix} = \mathrm{LN}_h(W_h h_{i-1}) + \mathrm{LN}_x(W_x T_i) \tag{5}$$

$$C_i = \sigma(F_i) \odot C_{i-1} + \sigma(I_i) \odot \tanh(G_{i-1}) \tag{6}$$

$$h_i = \sigma(O_t) \odot \tanh\left(\mathrm{LN}_c(C_t)\right) \tag{7}$$

F_i，I_i，O_i分别代表了遗忘门，输入门以及输出门；G_i代表隐藏向量，C_i代表记忆向量；h_i则代表了输出向量；LN_h，LN_x，LN_c分别代表不同的层归一化操作。

最后层的输出还是经过一个 Sigmoid 分类器进行句子分数的预测：

$$\hat{Y}_i = \sigma(W_o h_i + b_o) \tag{8}$$

2.5 模型训练与评估

2.5.1 实验过程

使用了 PyTorch, 以及 OpenNMT——一个用于神经机器翻译和神经序列学习的开源生态系统[8]，以及 Bert 预训练模型集合中的“bert-base-uncased”来实现该模型。

所有的模型都会在 1 个 GPU（GTX 3060Ti）上训练 50000 步。前一万步为整体学习率的预热，具体公式如下：

$$lr = 2e^{-3} \cdot \min(\text{step}^{-0.5}, \text{step} \cdot \text{warmup}^{-1.5}) \tag{9}$$

并且模型每训练 1000 步便进行模型的保存以及在验证数据集上计算损失，最后挑选损失最小的前三个模型在测试集上进行文章摘要工作，分别计算相关的指标来取平均得到最终模型的结果。

当预测新文档的摘要时，首先将使用训练好的模型来获得每个句子的最终标签值（得分）。然后将这些与信息重要程度的数值从高到低进行排序，并选择前 3 个关联的句子来作为摘要。

2.5.2 数据集

采用的数据集为 CNN/DailyMail 新闻数据集[9]，该数据集中的每篇新闻包含了文章的主体以及文章的简要概述。我们将数据集划分为训练集，验证集以及测试集，每个部分包含的具体新闻数量如下所示：

表 2-1 CNN/DailyMail 数据集统计指标

数据集	训练集	验证集	测试集
CNN	90266	1220	1093
DailyMail	196961	12148	10397

使用了 CoreNLP[10]，一套源自于斯坦福的使用 Java 语言编写的自然语言分析工具来对金融新闻进行分句。同时，尽管在两个数据集中的文章均含有文章的简要概述，但是不能满足我们抽取式摘要的任务，故我们需要手动计算出每篇文章的摘要以为数据集的标签，具体的做法是通过贪婪算法选择特定的语句最大化 ROUGE 分数，被选中作为摘要的语句标签值为 1，未被选中的标签值则为 0。

2.5.3 实验结果

ROUGE (Recall-Oriented Understudy for Gisting Evaluation)[11]，它是一套特定于自然语言处理中文本摘要与机器翻译领域的评价指标库。主要的工作原理是通过将机器或者人工生成的文本摘要或者翻译来进行通过数学公式的推导比较来生成最后的指标。我们通过计算不同的 Rouge 种类的分数，综合评估模型的性能，下面简单介绍一下使用的指标所代表的含义：

Rouge-N:系统和参考摘要之间 *n-gram* 的重叠。

Rouge-1:指系统与参考文献摘要之间的单字母(每个单词)重叠。

Rouge-2:指系统与参考文献摘要之间的 *2-gram* 重叠。

Rouge-L:基于统计的最长公共子序列，最长公共子序列问题自然地考虑了句子级结构的相似性，从而自动识别序列 *n* 元中的最长共现。

为了最终模型的效果比较，同时也采用了一些已经被提出来的文章摘要模型进行 ROUGE 分数的比较，这些方法包括：

LEAD：LEAD 是抽取式摘要的最基本的摘要模型，非常简单，具体做法是通过挑选原文章中前三个句子以作为文章摘要。

PGN[12]：PGN 是一种指针生成网络，它是一个基于编码器-解码器的抽象文章摘要模型。

REFRESH[13]：REFRESH 是通过强化学习进行全局最大化 ROUGE 分数的一种抽取式文章摘要模型。

同时对自己选择的不同的摘要层，分别计算最后的 Rouge-1，Rouge-2，Rouge-L 评分，所有的 ROUGE 得分结果如下表所示：

表 2-2 所有模型最终的 ROUGE 得分

Layer	Rouge-1	Rouge-2	Rouge-L
Simple Classifier	43.23	20.32	39.61
Transformer(L=2)	**43.26**	**20.34**	**39.69**
LSTM	43.22	20.15	39.56
LEAD	40.42	17.62	36.67
PGN	39.53	17.28	37.98
REFRESH	41.0	18.8	37.7

最终在三项指标中使用 *L*=2 的 transformer 编码器摘要分类层获得的效果最好，在 Rouge-1，Rouge-2，Rouge-L 指标中均取得最高的分数，故选取该模型来进行最终文章的摘要工作。

2.6 进行文本摘要

2.6.1 金融数据集

实验室使用网络爬虫技术，从数百个新闻网站中爬取了规模为 5G 的新闻数据集作为原始数据集，这些新闻数据与不同的金融公司相关，包含着大量金融领域的重要信息。同时为了方便研究，从这些原始的新闻评论数据集中通过一定的方式抽取了超过 1G 的数据量来作为实验的数据，这些未经标注的新闻数据可能会对在金融业务领域开发过程中起着不小的作用。数据集的相关统计

指标如下：

表 2-3 金融数据集的相关统计指标

金融公司数量	文章数量	文章平均长度
16	139559	12799

但同时考虑到网络爬取的数据具有不规范的特点，故采取一定的方法对数据进行了清洗，一是利用正则表达式去除了数据集合中的特殊字符，二是清除过短的不包含关键信息的新闻文章。

2.6.2 文章摘要

最后通过语料库 CNN/Daily Main 训练好的分类模型，对金融新闻数据集中某篇文章的语句进行评分，通过选取评分最高的 4 个语句（如果存在）作为该文章的摘要，历时 3 天最终完成了文章摘要的工作，处理之后数据集的相关统计指标如下表所示：

表 2-4 摘要数据集的数据统计指标

文章数量	文章平均长度
139119	546

最终的新闻文章的平均长度压缩比来到了 23:1，大大减少了文章分类工作时生成的文本向量空间的大小，降低了工作的复杂度与困惑度。

2.7 本章小结

本章基于对 Bert 输入序列以及分割嵌入空间的修改完成了利用 Bert 进行句向量表示的工作，同时采用了不同的摘要分类层对句向量进行评分以获得文章的摘要，最终使用 2 层 Transformer 编码器作为分类层，通过该模型对金融文本实现了文章摘要的工作，有效的完成了关键信息提取与错误信息的过滤任务，为接下来文本分类的工作打下了坚实的基础。

第 3 章 利用 LDA 主题模型进行金融摘要分类

3.1 LDA 主题模型

3.1.1 简介

主题建模是一种文本处理技术，其目的是通过在文本数据中寻找和展示模式来克服信息过载，这些模式被标识为主题。它不仅能够改善用户体验，还允许分析人员根据确定的主题快速浏览文本语料库或集合。

LDA (Latent Dirichlet Allocation)是一种用于主题建模的工具和技术[14]，它将文本分类到一个文档中，并将每个主题的单词进行分类，这些都是基于狄利克雷分布和过程进行建模的。LDA 做了两个关键假设，一是文档是主题和的混合体，二是主题是符号(或单词)的混合体。并且，这些主题利用概率分布来生成单词。在统计语言中，文档被称为主题的概率密度(或分布)，而主题是单词的概率密度(或分布)。

3.1.2 模型思想

按照上面的说法，LDA 看似平平无奇，但深入内部探其中的数学原理会发现其中涉及的知识点杂多且繁琐，但由于本篇文章侧重于 LDA 模型在摘要的应用而不是理论研究上，故仅简要阐述该模型的中心思想如下：

1.从狄利克雷分布α中取样生成文档 i 的主题分布θ_i

2.从主题的多项式分布θ_i中取样生成文档 i 第 j 个词的主题$z_{i,j}$

3.从狄利克雷分布中β取样生成主题$z_{i,j}$对应的词语分布$\Phi_{z_{i,j}}$

4.从词语的多项式分布$\Phi_{z_{i,j}}$中采样最终生成词语$w_{i,j}$

其中，狄利克雷β分布是二项式分布的共轭先验概率分布，而狄利克雷分布（Dirichlet 分布）是多项式分布的共轭先验概率分布。LDA 的图模型结构如

下图所示（类似贝叶斯网络结构）：

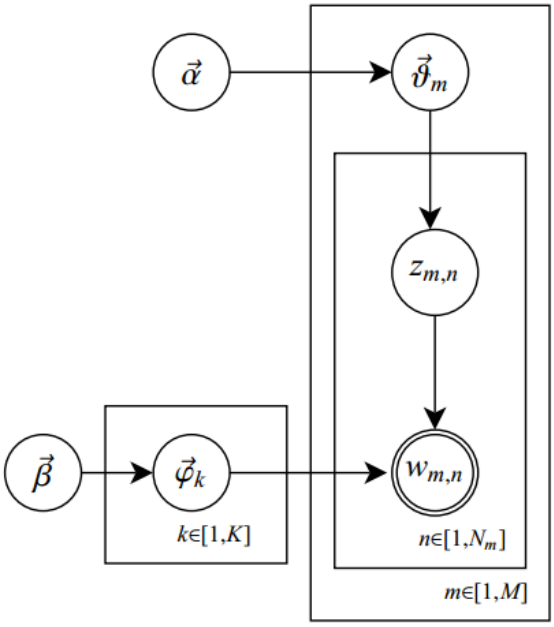

图 3-1 LDA 主题模型计算框架

3.2 数据预处理

尽管摘要后的新闻数据已经包含了关键信息，但是为了使其能够适应于 LDA 模型，获得更优的分类结果，故对其进行数据预处理[15]，具体流程图如下：

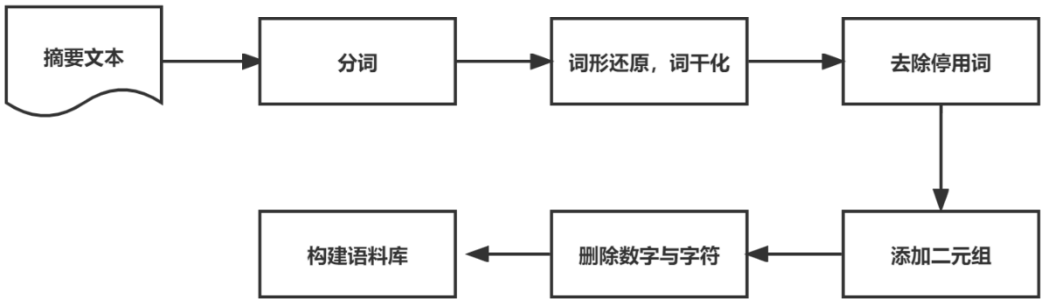

图 3-2 摘要数据集数据预处理流程图

其中分词就是将句子划分，而词形还原可以让最后的模型主题关键词的输出变的通俗易懂；同时去除停用词与删除数字字符可以进一步的过滤掉无用的信息，添加二元组则能够将高频连词转化为一个词汇，丰富最终的语料库，挖掘出更多的信息，这部分工作使用的是一个文本预处理的 python 库 nltk。

3.3 构建 LDA 模型法

3.3.1 构建文本词向量

首先根据语料库中的所有文档构建单词字典，统计出所有单词以及对应的编号，接着分别计算出每一篇文档的单词出现在字典中的次数，使用一个单词编号—出现次数字典来表示，最终整个字典列表就是摘要文本集合的向量化表示，以作为 LDA 模型的输入。构建语料库的相关统计如下：

表 3-1 摘要文本库字典数据统计

文档数量	字符字典大小
139119	22484

3.3.2 训练相关参数

在训练 LDA 模型之前，需要对模型的参数进行一定的设置，参数的合适与否能够影响到最终模型的优劣，下面来简单介绍一些主要的训练参数：

***ChunkSize*:** *ChunkSize* 控制模型一次训练的文档数量，采用越大的 *ChunkSize*
能够使模型训练的速度越快，当然最大不能超过计算机的内存大小，否则会出现错误。同时它也会影响到最终模型的质量，但是却可以忽略不计[15]。

Passes: *Passes* 控制对整个语料库进行训练的次数，需要足够高。

Iterations: *Iterations* 控制对一个 *ChunkSize* 的文档重复循环训练的次数，也需要足够高。

***Num_Topics*:** 最终主题模型的分类主题数量，对于特定的训练文本，挑选一个最佳的主题数量至关重要。

3.3.3 主题一致性

在衡量模型训练的好坏上，使用了主题一致性指标（Topic Coherence）—一种衡量主题可解释性与质量的数字指标，越高越好。通查它的计算过程包括

四个部分[16]:

分词: 分词模块负责创建成对的词子集，将用它们来评估主题的连贯性。

概率计算：一致性度量使用从文本语料库中提取的概率。概率计算步骤定义了如何计算这些概率。

确认度量：计算一致性的核心部分，这一步通过使用从语料库中计算出的概率来量化两个子集之间的“关系”。因此，如果 A 子集中的单词与 B 子集中的单词相连接(例如，经常出现在同一个文档中)，确认度量将会很高。

聚合：这是最后一步也是最简单的一步，它将前面步骤中计算的所有值汇总成一个值，这就是我们最后的主题一致性得分。

整个算法流程图如下所示：

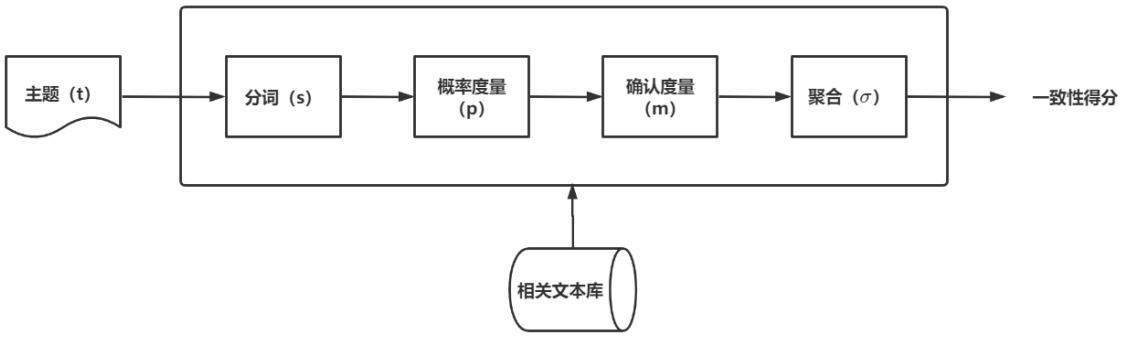

图 3-3 主题一致性得分计算过程

3.3.3 最佳主题数量的确定

由于原始数据集不存在标签，只知道与金融相关，故无法确定最终需要分类的主题数量，为此将通过超参暴力搜索的方式探索最佳的主题数量。设置初始值为 4 个主题，间隔为 2，最大的主题数目为 30，这个数字不可以太大以防止分类出的主题含义不够清晰明确。完成迭代以后，绘制一个主题数目—主题一致性得分折线图，通过观察分数的变化趋势以更直观的判断最佳数目：

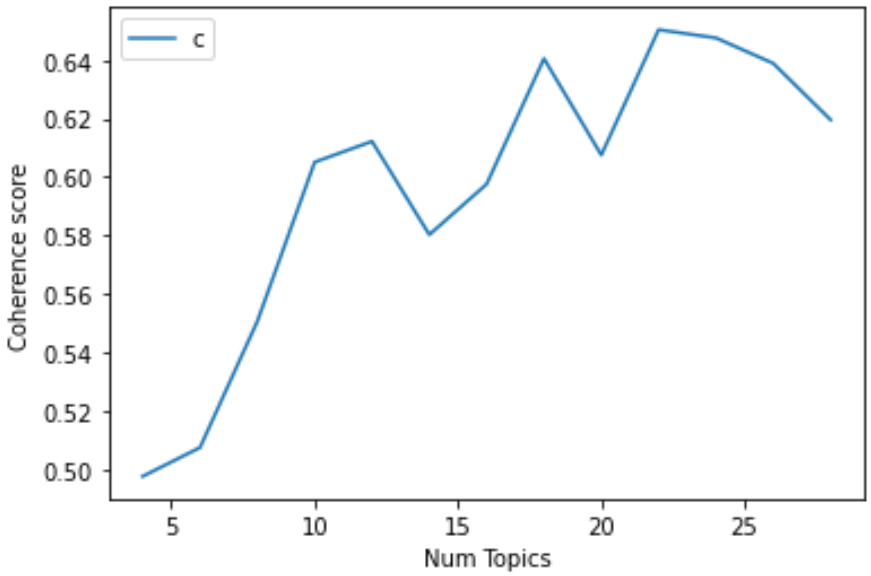

图 3-4 主题数量—一致性得分折线图

在分类的主题数为 22 时主题一致性得分达到最大值，同时由于整体分数的变化在这之后随着主题数量的增加趋近于平缓，故选择 22 作为最终分类的主题数量是非常合适的。在选取该数目之后，进行了模型的最终训练，为接下来的文本标注工作做铺垫。

3.4 结果可视化与文本标注

3.4.1 主题模型可视化

仅有主体一致性得分来衡量最终模型的效果略显单调，故在这基础之上使用了 pyldavis 库—一个专门用于可视化 LDA 主题模型的可视化工具来对最后的模型效果进行细致的展示[17]，使用的是可互动式的基于 web 的可视化方法，模型生成的 html 图像如下所示：

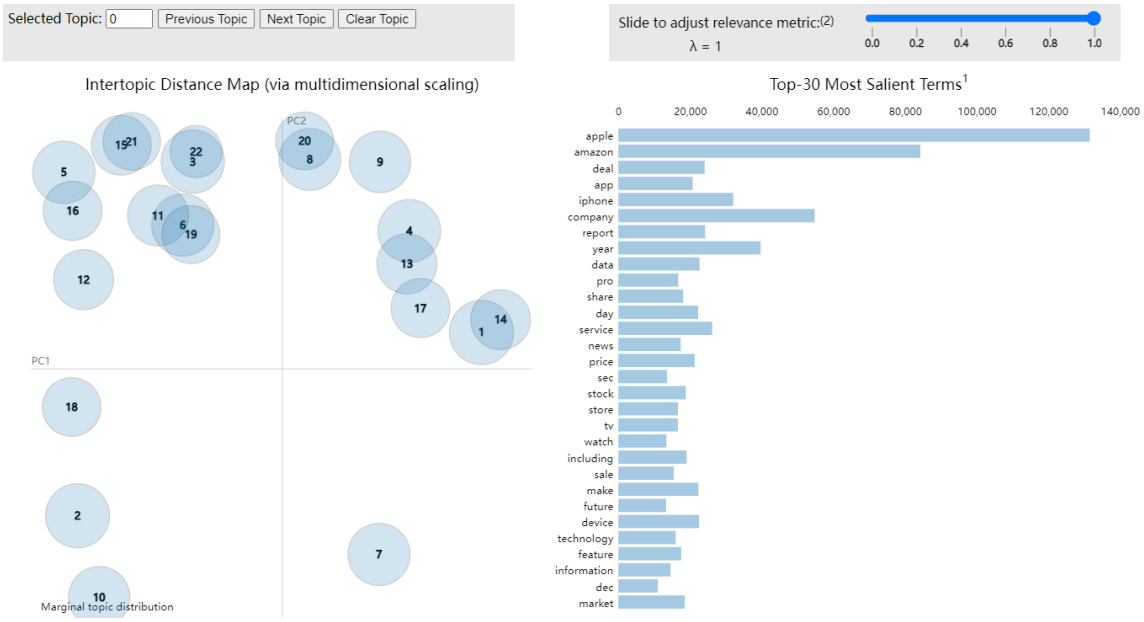

图 3-5 pyldavis 生成的主题模型 html 网页

左边的气泡表示的是已经分类的主题，通过气泡的大小来表示主题的重要程度，可以发现分类的主题大小相似，也间接说明了金融新闻文本所包含的信息密度大。而上方的文本框有一定的筛选功能，可以选特定序号的主题气泡。右边的纵向条形图则依据了左边所选的相关主题展示出与其相关联的主题词汇，同时按照词汇对主题的贡献程度从上到下进行了排序。

观察最终的结果可以发现大多数主题之间不存在重叠部分，同时尽管有小部分主题出现重叠但是重叠的部分很大，在最后进行文本标注的工作时可以将他们合并为一个主题，再结合主题一致性系数大于 0.6，可以得出最后使用主题模型的效果可圈可点的结论。

3.4.2 文本标注

通过训练好的主题模型，就可以将每一篇金融新闻进行主题标注，陈列出该主题的关键词，和这篇新闻摘要在分类的主题下的得分,下面给出前 10 篇文章标注示例如下：

	Document_No	Dominant_Topic	Topic_Perc_Contrib	Keywords
0	0	6.0	0.0924	pro, iphone, model, ipad, inch, macbook, mac, ...
1	1	18.0	0.6442	including, sec, data, information, gov, practi...
2	2	9.0	0.3047	space, bezos, year, ceo, jeff, company, blue, ...
3	3	17.0	0.4926	apple, app, store, apps, game, content, google...
4	4	6.0	0.2011	pro, iphone, model, ipad, inch, macbook, mac, ...
5	5	1.0	0.1198	dec, ago, hour, est, pm, adobe, security, site...
6	6	0.0	0.3404	december, week, tech, point, stock, high, time...
7	7	7.0	0.1941	apple, iphone, year, company, cook, china, rep...
8	8	20.0	0.2714	share, stock, year, billion, quarter, company,...
9	9	13.0	0.2534	company, tech, facebook, state, case, group, c...

图 3-6 前 10 个文档的标注信息

同时我们还可以列举出不同主题的关键词集合，以及新闻在主题下的得分来得到所有主题各自的头号文本来帮助去更好的理解隐藏在主题背后中除了关键词之外的具体含义，获得更好的分类效果，表格如下：

	Topic_Num	Topic_Perc_Contrib	Keywords	Text
0	0.0	0.5804	video, game, tv, service, streaming, content, ...	[safe, travel, expert, say, order, jiomart, vi...
1	1.0	0.7033	service, aws, data, cloud, customer, digital, ...	[discover, secure, future, ready, cloud, solut...
2	2.0	0.5382	year, month, customer, time, http, earlier, ba...	[per, monthnew, customer, onlycancel, anytime,...
3	3.0	0.6908	news, day, big, technology, story, tech, worki...	[es, cooky, no, permitem, coletar, alguns, dad...
4	4.0	0.5720	iphone, apple, pro, ipad, model, mac, macbook,...	[curiously, apple, charging, premium, new, cel...
5	5.0	0.6273	year, business, company, million, market, bill...	[source, business, insider, source, business, ...
6	6.0	0.5197	report, future, ad, browser, enable, enabled, ...	[national, shooting, sport, foundation, gun, m...
7	7.0	0.5673	company, amazon, cook, tech, giant, tim, googl...	[data, protection, bill, changed, fundamentall...
8	8.0	0.3963	time, thing, people, airbnb, home, good, make,...	[may, seem, like, extreme, approach, fashion, ...
9	9.0	0.6178	series, show, season, film, tv, star, world, l...	[apple, ecstatically, announced, afternoon, la...
10	10.0	0.5899	chip, amd, rating, based, intel, performance, ...	[xda, official, marketplace, buying, selling, ...

图 3-7 前 10 个主题的关键词，最佳新闻以及对应得分

最后由下表给出所有主题的新闻文本数量数以及在总体的占比情况：

	Dominant_Topic	Topic_Keywords	Num_Documents	Perc_Documents
21.0	0.0	video, game, tv, service, streaming, content, ...	5046	0.0363
11.0	1.0	service, aws, data, cloud, customer, digital, ...	7187	0.0517
19.0	2.0	year, month, customer, time, http, earlier, ba...	5155	0.0371
10.0	3.0	news, day, big, technology, story, tech, worki...	4073	0.0293
0.0	4.0	iphone, apple, pro, ipad, model, mac, macbook,...	6797	0.0489
15.0	5.0	year, business, company, million, market, bill...	7852	0.0564
18.0	6.0	report, future, ad, browser, enable, enabled, ...	8809	0.0633
9.0	7.0	company, amazon, cook, tech, giant, tim, googl...	7258	0.0522
14.0	8.0	time, thing, people, airbnb, home, good, make,...	2637	0.0190
3.0	9.0	series, show, season, film, tv, star, world, l...	3565	0.0256
20.0	10.0	chip, amd, rating, based, intel, performance, ...	5279	0.0379
8.0	11.0	deal, amazon, sale, day, black, friday, affili...	6085	0.0437
17.0	12.0	device, amazon, smart, airpods, home, alexa, w...	5181	0.0372

图 3-8 主题关键词以及文本数量统计

3.5 本章小结

本章基于 LDA 主题模型对摘要后的金融新闻进行了文本分类的工作，首先进行了数据预处理，其次确认了模型训练相关的参数，接着通过迭代主题一致性得分的方式来确认最终的最佳主题的分类数目，并利用了 pyldavis 可视化工具来更直观的体现模型训练的效果，最后通过训练好的 LDA 模型对文本进行了相关的标注与统计工作。

第 4 章 利用 K-means 算法进行文本聚类

4.1 K-means 算法

随着互联网的普及与飞速发展，产生的数据量是难以理解的巨大。尽管个人数据的本质是简单明了的，但要分析的数据量之大，即使是计算机也难以处理。数据挖掘方法和技术，结合机器学习，使我们能够以一种可理解的方式分析大量数据。而 K-means 就是一种数据聚类技术，可用于无监督机器学习[18]。它能够根据相似性(k)将未标记的数据分类到预定数量的簇中。

K-mean 聚类算法是一种无监督机器学习，该算法根据所提供的特征，迭代地将每个数据点分配给 K 个组中的一个。所有点分配完毕之后，再根据一个集群内的所有点重新计算该集群的中心点(取平均值)，依次迭代的进行分配点和更新类簇中心点的步骤，直至类簇中心点的变化很小，或者达到指定的迭代次数。整个算法流程图如下所示：

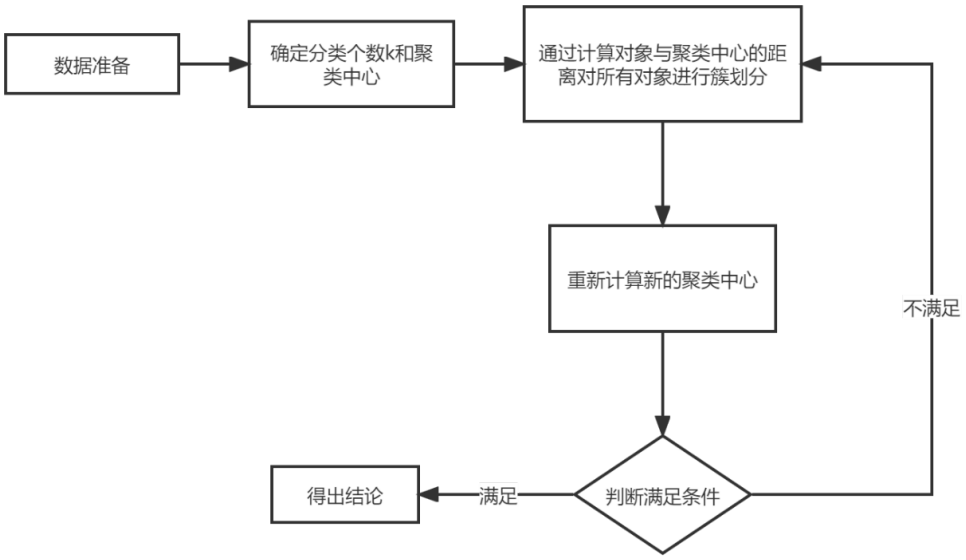

图 4-1 K-means 聚类算法流程图

4.2 准备工作

4.2.1 数据预处理

数据预处理采用的是和 LDA 主题模型一致的方法故不在此进行赘述。

4.2.2 文本向量化

TF-IDF 是一种统计度量，它可以评估一个单词与文档集合中的一个文档的相关性。一般通过相乘两个指标来实现:一个单词在文档中出现的次数，以及该单词在一组文档中的逆向文档频率。它有很多用途，最重要的是在自动文本分析中，并且在自然语言处理(NLP)的机器学习算法中对单词评分非常有用。TF-IDF 是为文档检索和信息检索而发明的。TF-IDF 的得分值是根据单词在文档中出现的次数成比例地增加，但会被包含该单词的文档的数量所抵消。因此，在每个文档中都常见的单词，如 this、what 和 if，即使它们可能多次出现，但排名较低，因为它们对该文档没有多大意义。但是，如果“Bug”一词在文档中多次出现，而在其他文档中却没有多次出现，这可能意味着它非常重要。TF-IDF 的计算公式为：

$$\text{tf-idf}(t,d,D) = \text{tf}(t,d) \cdot \text{idf}(t,D) \tag{10}$$

其中 *tf* 表示某个单词在文中出现的次数，简称词频，而 *idf* 则表示逆向文件频率，先通过文档的总数目除以所有存在该单词的文档数目，然后取对数得到最终的结果。两者的计算公式分别为：

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{11}$$

$$\text{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{12}$$

因此新闻摘要的文档向量化模式是首先通过文本语料库得到对应的类似于 LDA 主题模型中的单词字典，接着再对某篇文档，计算其每个词的 *tf-idf* 值，最后用这些值组成该篇文档的文本向量用作输出，这样做的好处是可以让单词的重要程度更合理的凸显出来，坏处则是并没有有效的利用好上下文的语义关系。

4.2.3 截断 SVD 降维

为了在机器学习中更准确地预测结果，我们需要更多的清理数据，如果数据集中的输入变量维度太高，尽管这不会影响到输出变量，但会影响总体结果。

如果未使用的变量数量很高，就必须将它们从数据集中删除，以便更准确地执行进一步的机器学习任务。而减少用于预测分析的输入变量的数量被称为降维。因此，将较少的数据输入变量放入预测模型中，可以使预测模型更简单、性能更高。

而奇异值分解算法 SVD 就是一种矩阵降维算法[19]，它通过将矩阵转化为不同的部分来简化计算。截断型 SVD 是一种类似于主成分分析(PCA)的矩阵分解技术。对于一个给定的$m \times n$型矩阵截断 SVD 将产生具有指定列数 k 的矩阵，这意味着它将删除除了提供给它的特性数量之外的所有特性。同时截断的 SVD 不会使数据处于中心位置，这使得它在处理稀疏数据时更加有效。整体算法的流程图如下所示：

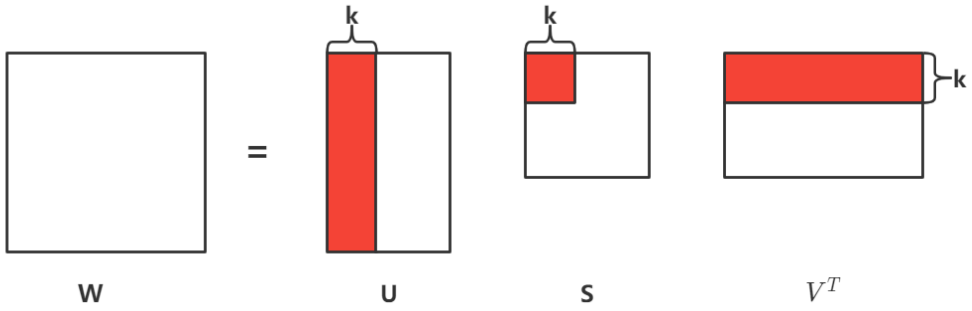

图 4-2 截断 SVD 算法图

分别选取特征维度分别为 5000，2000，1000，通过截断 SVD 对文本向量进行降维，可以有效的帮助从稀疏矩阵中提取出关键特征，在减少计算量的同时优化整体模型最终进行聚类的效果。

4.2.4 Mini-batch K-means 算法

小批量 K-means 算法[20]的主要思想是使用小批量随机固定大小的数据，这样它们就可以存储在内存中。每次迭代从数据集中获得一个新的随机样本，并用于更新集群，这是重复的，直到收敛。每个小批量使用原型值和数据的凸组合更新集群，应用随迭代次数减少的学习率。这个学习率与在此过程中分配给集群的数据数量成反比。随着迭代次数的增加，新数据的影响会降低，所以当连续几次迭代中簇内没有发生变化时，可以检测到收敛性。它的效果一般只略

差于标准算法，在聚类工作中有很重要的地位故我们采用 Mini-batch 聚类算法作为普通 K-means 算法的一种对比。

4.3 文本聚类

4.3.1 判断最佳聚类数

与求解 LDA 主题模型的最佳主题数量相似，我们也需要合适的设置一个最终聚类数量，但是需要注意聚类的数量的选取遵循尽量不要太多的原则，以防止聚类的类群淡化文本类群的意义，故采用手肘法确定 K 的最佳值，通过计算不同 K 值情况下的 SSE（簇类均方差）绘制折线图，寻找手肘点来找到最合适的簇的数目，绘制的图像如下所示：

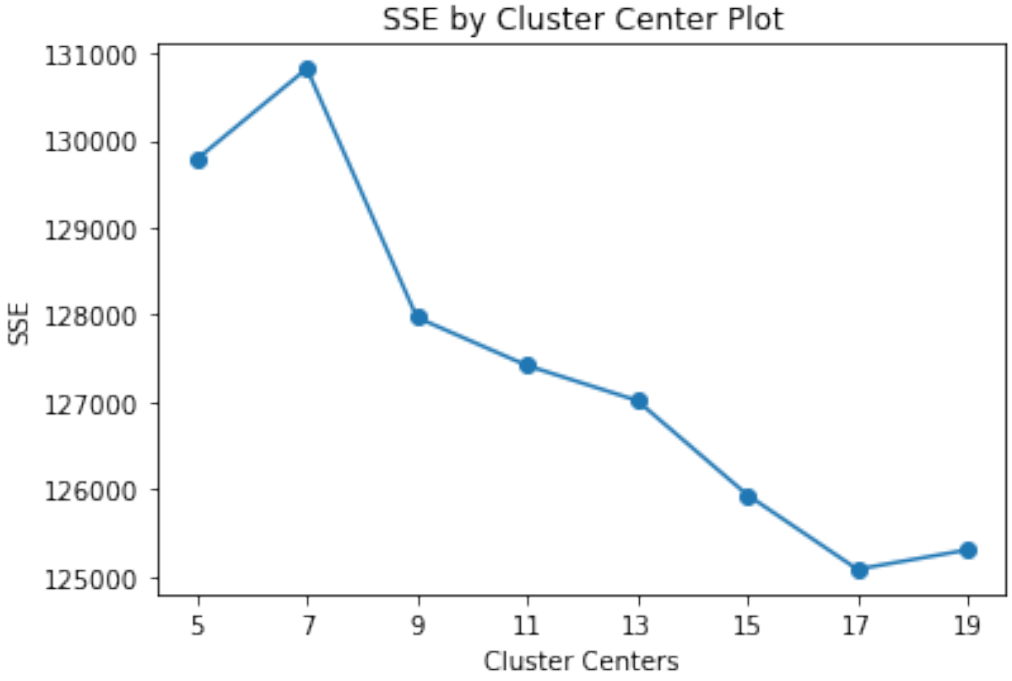

图 4-3 K 值-SSE 折线图

通过观察图像可以发现在 K 在 7 到 9 之间 SSE 值骤降，当 K 大于 9 时折线将逐渐趋近平缓，故选择 9 以作为最终算法聚类的数量。

4.3.2 聚类结果评估与对比

考虑到原有的金融数据集并不存在标签，故使用不需要标签的评价聚类效果的指标。

一是轮廓系数[21]，轮廓系数的大小一般与模型最终聚类效果成正比，每篇轮廓系数是依据进行定义，每个样例由两个得分组成，一个为样本与同一种类中所有其他点之间的平均距离。另一个为样本与距离最近的簇中的所有其他点之间的平均距离设置为 a 和 b，最后将单个样本的轮廓系数 s 给出：

$$s = \frac{b - a}{\max(a, b)} \tag{13}$$

将每个样本的轮廓系数相加起来求平均值就可以得到最终的结果了。最终大小越接近于 1，聚类的效果就越合理。

二是 Davies-Bouldin 指数（DBI）[22]，该指数被定义为每个聚类C_i与其最相似的聚类C_j之间的平均相似度。在这个指数的上下文中，相似性被定义为 $R_{i,j}$，分别由s_i和$d_{i,j}$来权衡，s_i为簇 i 中每个点与簇心之间的平均距离，也称为簇直径。$d_{i,j}$为则两个簇中心点之间的距离，计算$R_{i,j}$的公式为：

$$R_{ij} = \frac{s_i + s_j}{d_{ij}} \tag{14}$$

最后 DB 指数的公式被定义为：

$$DB = \frac{1}{k}\sum_{i=1}^{k} \underset{i \neq j}{max} R_{ij} \tag{15}$$

我们分别对是否进行数据降维，以及使用的聚类算法是否为 Mini-Batch K-means 来进行了 4 组聚类实验，最后分别统计出他们的轮廓系数与 DB 指数作为聚类效果的评估，最后的实验数据如下所示：

表 4-1 不同的聚类实验结果评估

Experiment	Silhouette Coefficient	Davies-Bouldin Index
K-means + svd	0.59	0.64
MinibatchK + svd	0.44	0.79
K-means	0.44	\
MiniBatchK	0.34	\

因为 DB 指标不适用于稀疏矩阵的评估，故仅对使用了截断 SVD 的实验进行聚类效果评估，观察上表可以发现使用了数据降维可以让最后的聚类效果变得更好，者很有可能与降维凸显出关键信息的作用有关，同时使用了 Mini-Batch K-means 的实验尽管效果不及 K-means，但是差距并不是很多且它的训练速度很快，在大数据集上还是有一定的意义，在最后的效果最好的实验中轮廓系数大于 0.55，DB 指数小于 1，可以说最终聚类的结果还是比较不错的。

4.4 文本标注

对于使用 TF-IDF 向量化的文本，我们在聚类后可以展示每个聚类结果中的一些高频词汇，下面展示最好的模型聚类后各个类群的关键词：

```
Top terms per cluster:
Cluster 0: aws service data cloud please new amazon make use like
Cluster 1: deal amazon price prime best offering time today day sale
Cluster 2: stock share market quarter nasdaq earnings amd company inc investor
Cluster 3: please blocker proceeding blocked frequently happening forum happen ensure feedback
Cluster 4: apple iphone new app pro watch io ipad store feature
Cluster 5: newsletter news intersect coindesk inner deeply crypto commentary miss sunday
Cluster 6: sec gov declare downloading traffic practice information data including best
Cluster 7: tv show amazon series apple alexa streaming new film video
Cluster 8: purchase rating price average traded estimated stock portfolio holding due
Cluster 9: amazon company said bezos year business new space billion million
```

图 4-4 聚类类群关键词

简单的观察就可以发现类群 4 可以看出与苹果公司相关，类群 2 则与 amazon 公司以及会员活动相关等等，通过这些关键词可以对某个类群中的所有文本进行关键词注释，以达到金融新闻摘要标注的效果。

4.5 本章小结

本章通过 K-means 算法对金融新闻摘要实现了文本聚类的工作，通过手肘法确定了最佳的类群数量，同时使用了截断 SVD 降维对稀疏矩阵进行处理以及 Mini-Batch K-means 方法用作对比实验，最后利用轮廓系数以及 DB 指数作为聚类效果评估的指标，最终的实验结果可圈可点。

第 5 章 结论

5.1 本文工作总结

本文主要实现了两个工作，第一个是基于 Bert 模型修改了它的输入序列以及分割嵌入以将适用于文本句向量的表示工作，然后采用了三种不同的摘要分类层，线性层，Transformer 编码器，以及 LSTM 神经网络，将利用 Bert 模型得到的句向量进行分类，在利用贪心算法标注初始数据集之后，通过比较不同的 ROUGE 分数来选择摘要效果最好的模型。然后对实验室提供的金融新闻数据集进行了预处理与摘要工作，最终的压缩比来到了 23:1，在提取出关键信息的同时能够大大减少接下来文本标注的工作量。第二个工作是实现文本分类，采用了 LDA 主题模型以及 K-means 算法，在 LDA 中通过主题一致性得分以及 pyldavis 库函数评估最后模型的效果，在 K-means 算法中使用了截断降维以及小批量 K-means 算法以作对比试验，通过轮廓系数与 DB 指标来评估模型，两种方法均采用了迭代的方法来求解出最佳的分类主题或者类群数量，完成了文本标注的工作，最后的实验都取得了不错的效果。

5.2 本文工作展望

主要有展望有两点，一是对金融分类的摘要模型的训练数据集为普通的新闻数据集，可以尝试网上爬虫的方法来寻找更多的与金融新闻数据，用贪心算法进行新闻摘要标签，以此来训练摘要模型，这样训练的模型与金融文本相关，可以在摘要信息的提取工作上更进一步。二是在使用 LDA 主题模型进行文本主题分类的最终结果中，各主题之间存在一定的关键词重合，我将试着找出其中原因，并在最终对模型的评估中采用除了主题一致性以外的评价指标来全面的衡量最终分类的结果。

致谢

韶光易逝，劝君惜取少年时。从初进大学时的青涩懵懂无知到现在，无论是在学习生活还是个人的成长中，都有过很多让人难忘的经历。临近本科生涯结束之际，百感交集，在此写下我最真挚的祝福，感谢这一路来老师同学以及家人的支持与鼓励。

记得是在一节研讨课中有幸认识了刘老师，老师对高性能计算领域充满热情的讲解深深打动了我。在后来高性能实验室的日子里，刘老师悉心的指导与温柔的性格不仅让我从开始接触这个方向生疏的担忧转为愿意去主动寻求机会深入，也给了我很多的机会。在此次毕业设计过程中，老师也耐心与我沟通选题，交流进展与提供服务器设施等等，给了我非常多的帮助。同时也要感谢实验室的张晨学姐在毕业设计过程中的指导，能让我在遇到困难时不会手足无措；也要感谢 18 级大数据班的所有同学们，大家在本科的后三年互相帮助，讨论问题，一起参加活动，让我感受到了家的温暖。

能够拥有这样的老师和同学是何等幸运的一件事啊，衷心祝愿大家在接下来人生的旅途中能够顺利，过得精彩。

参考文献：

[1] 王海宁.自然语言处理技术发展[J/OL].中兴通讯技术:1-11[2022-05-09].http://kns.cnki.net/kcms/detail/34.1228.TN.20220408.1420.004.html

[2] 熊云波. 文本信息处理的若干关键技术研究[D].复旦大学,2006.

[3] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[4] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.

[5] Ba J L, Kiros J R, Hinton G E. Layer normalization[J]. arXiv preprint arXiv:1607.06450, 2016.

[6] 杨丽,吴雨茜,王俊丽,刘义理.循环神经网络研究综述[J].计算机应用,2018,38(S2):1-6+26.

[7] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.

[8] Klein G, Kim Y, Deng Y, et al. Opennmt: Open-source toolkit for neural machine translation[J]. arXiv preprint arXiv:1701.02810, 2017.

[9] Hermann K M, Kocisky T, Grefenstette E, et al. Teaching machines to read and comprehend[J]. Advances in neural information processing systems, 2015, 28.

[10] Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., & McClosky, D. (2014, June). The Stanford CoreNLP natural language processing toolkit. In Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations (pp. 55-60).

[11] Lin C Y. Rouge: A package for automatic evaluation of summaries[C]//Text summarization branches out. 2004: 74-81.

[12] See A, Liu P J, Manning C D. Get to the point: Summarization with pointer-generator networks[J]. arXiv preprint arXiv:1704.04368, 2017.

[13] Narayan S, Cohen S B, Lapata M. Ranking sentences for extractive summarization with reinforcement learning[J]. arXiv preprint arXiv:1802.08636,

2018.

[14] Blei D M, Ng A Y, Jordan M I. Latent dirichlet allocation[J]. Journal of machine Learning research, 2003, 3(Jan): 993-1022.

[15] 许辉.数据挖掘中的数据预处理[J].电脑知识与技术,2022,18(04):27-28+31.DOI:10.14004/j.cnki.ckt.2022.0262.

[16] Hoffman M, Bach F, Blei D. Online learning for latent dirichlet allocation[J]. advances in neural information processing systems, 2010, 23.

[17] Röder M, Both A, Hinneburg A. Exploring the space of topic coherence measures[C]//Proceedings of the eighth ACM international conference on Web search and data mining. 2015: 399-408.

[18] Sievert C, Shirley K. LDAvis: A method for visualizing and interpreting topics[C]//Proceedings of the workshop on interactive language learning, visualization, and interfaces. 2014: 63-70.

[19] Arthur D, Vassilvitskii S. k-means++: The advantages of careful seeding[R]. Stanford, 2006.

[20] Alter O, Brown P O, Botstein D. Singular value decomposition for genome-wide expression data processing and modeling[J]. Proceedings of the National Academy of Sciences, 2000, 97(18): 10101-10106.

[21] Béjar Alonso J. K-means vs mini batch k-means: A comparison[J]. 2013.

[22] Rousseeuw P J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis[J]. Journal of computational and applied mathematics, 1987, 20: 53-65.

[23] Davies D L, Bouldin D W. A cluster separation measure[J]. IEEE transactions on pattern analysis and machine intelligence, 1979 (2): 224-227.

附录 1 英文原文

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova

Google AI Language

{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Abstract

We introduce a new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement).

1 Introduction

Language model pre-training has been shown to be effective for improving many natural language processing tasks (Dai and Le, 2015; Peters et al., 2018a; Radford et al., 2018; Howard and Ruder, 2018). These include sentence-level tasks such as natural language inference (Bowman et al., 2015; Williams et al., 2018) and paraphrasing (Dolan and Brockett, 2005), which aim to predict the relationships between sentences by analyzing them holistically, as well as token-level tasks such as named entity recognition and question answering, where models are required to produce fine-grained output at the token level (Tjong Kim Sang and De Meulder, 2003; Rajpurkar et al., 2016).

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning *all* pre-trained parameters. The two approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations.

We argue that current techniques restrict the power of the pre-trained representations, especially for the fine-tuning approaches. The major limitation is that standard language models are unidirectional, and this limits the choice of architectures that can be used during pre-training. For example, in OpenAI GPT, the authors use a left-to-right architecture, where every token can only attend to previous tokens in the self-attention layers of the Transformer (Vaswani et al., 2017). Such restrictions are sub-optimal for sentence-level tasks, and could be very harmful when applying fine-tuning based approaches to token-level tasks

such as question answering, where it is crucial to incorporate context from both directions.

In this paper, we improve the fine-tuning based approaches by proposing BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. BERT alleviates the previously mentioned unidirectionality constraint by using a "masked language model" (MLM) pre-training objective, inspired by the Cloze task (Taylor, 1953). The masked language model randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked word based only on its context. Unlike left-to-right language model pre-training, the MLM objective enables the representation to fuse the left and the right context, which allows us to pre-train a deep bidirectional Transformer. In addition to the masked language model, we also use a "next sentence prediction" task that jointly pre-trains text-pair representations. The contributions of our paper are as follows:

- We demonstrate the importance of bidirectional pre-training for language representations. Unlike Radford et al. (2018), which uses unidirectional language models for pre-training, BERT uses masked language models to enable pre-trained deep bidirectional representations. This is also in contrast to Peters et al. (2018a), which uses a shallow concatenation of independently trained left-to-right and right-to-left LMs.
- We show that pre-trained representations reduce the need for many heavily-engineered task-specific architectures. BERT is the first fine-tuning based representation model that achieves state-of-the-art performance on a large suite of sentence-level *and* token-level tasks, outperforming many task-specific architectures.

- BERT advances the state of the art for eleven NLP tasks. The code and pre-trained models are available at https://github.com/google-research/bert.

2 Related Work

There is a long history of pre-training general language representations, and we briefly review the most widely-used approaches in this section.

2.1 Unsupervised Feature-based Approaches

Learning widely applicable representations of words has been an active area of research for decades, including non-neural (Brown et al., 1992; Ando and Zhang, 2005; Blitzer et al., 2006) and neural (Mikolov et al., 2013; Pennington et al., 2014) methods. Pre-trained word embeddings are an integral part of modern NLP systems, offering significant improvements over embeddings learned from scratch (Turian et al., 2010). To pre-train word embedding vectors, left-to-right language modeling objectives have been used (Mnih and Hinton, 2009), as well as objectives to discriminate correct from incorrect words in left and right context (Mikolov et al., 2013).

These approaches have been generalized to coarser granularities, such as sentence embeddings (Kiros et al., 2015; Logeswaran and Lee, 2018) or paragraph embeddings (Le and Mikolov, 2014). To train sentence representations, prior work has used objectives to rank candidate next sentences (Jernite et al., 2017; Logeswaran and Lee, 2018), left-to-right generation of next sentence words given a representation of the previous sentence (Kiros et al., 2015), or denoising auto-encoder derived objectives (Hill et al., 2016).

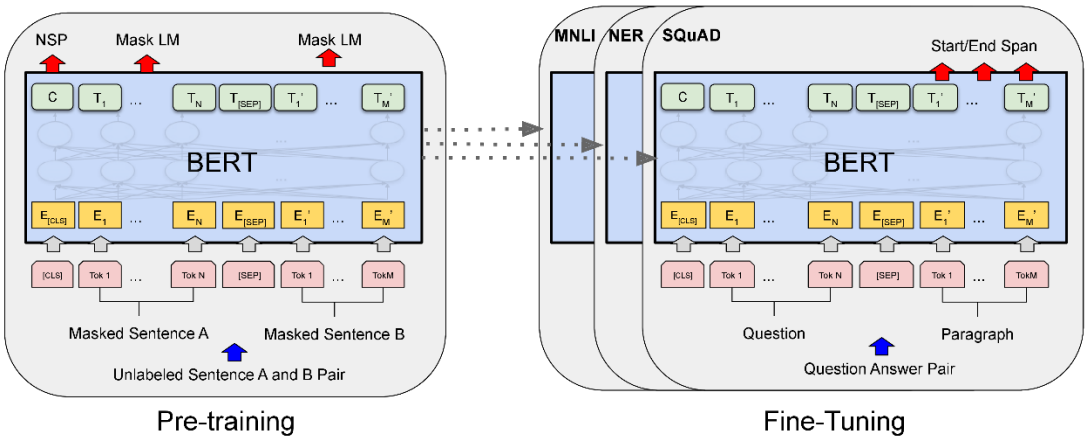

Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).

ELMo and its predecessor (Peters et al., 2017, 2018a) generalize traditional word embedding research along a different dimension. They extract *context-sensitive* features from a left-to-right and a right-to-left language model. The contextual representation of each token is the concatenation of the left-to-right and right-to-left representations. When integrating contextual word embeddings with existing task-specific architectures, ELMo advances the state of the art for several major NLP benchmarks (Peters et al., 2018a) including question answering (Rajpurkar et al., 2016), sentiment analysis (Socher et al., 2013), and named entity recognition (Tjong Kim Sang and De Meulder, 2003). Melamud et al. (2016) proposed learning contextual representations through a task to predict a single word from both left and right context using LSTMs. Similar to ELMo, their model is feature-based and not deeply bidirectional. Fedus et al. (2018) shows that the cloze task can be used to improve the robustness of text generation models.

2.2 Unsupervised Fine-tuning Approaches

As with the feature-based approaches, the first works in this direction only pre-trained word embedding parameters from unlabeled text (Collobert and Weston, 2008).

More recently, sentence or document encoders which produce contextual token representations have been pre-trained from unlabeled text and fine-tuned for a supervised downstream task (Dai and Le, 2015; Howard and Ruder, 2018; Radford et al., 2018). The advantage of these approaches is that few parameters need to be learned from scratch. At least partly due to this advantage, OpenAI GPT (Radford et al., 2018) achieved previously state-of-the-art results on many sentence-level tasks from the GLUE benchmark (Wang et al., 2018a). Left-to-right language modeling and auto-encoder objectives have been used for pre-training such models (Howard and Ruder, 2018; Radford et al., 2018; Dai and Le, 2015).

2.3 Transfer Learning from Supervised Data

There has also been work showing effective transfer from supervised tasks with large datasets, such as natural language inference (Conneau et al., 2017) and machine translation (McCann et al., 2017). Computer vision research has also demonstrated the importance of transfer learning from large pre-trained models, where an effective recipe is to fine-tune models pre-trained with ImageNet (Deng et al., 2009; Yosinski et al., 2014).

3 BERT

We introduce BERT and its detailed implementation in this section. There are two steps in our framework: *pre-training* and *fine-tuning*. During pre-training, the model is trained on unlabeled data over different pre-training tasks. For fine-tuning, the BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. Each downstream task has separate fine-tuned models, even though they

are initialized with the same pre-trained parameters. The question-answering example in Figure 1 will serve as a running example for this section.

A distinctive feature of BERT is its unified architecture across different tasks. There is minimal difference between the pre-trained architecture and the final downstream architecture.

Model Architecture

BERT's model architecture is a multi-layer bidirectional Transformer encoder based on the original implementation described in Vaswani et al. (2017) and released in the tensor2tensor library.[1] Because the use of Transformers has become common and our implementation is almost identical to the original, we will omit an exhaustive background description of the model architecture and refer readers to Vaswani et al. (2017) as well as excellent guides such as "The Annotated Transformer."[2]

In this work, we denote the number of layers (i.e., Transformer blocks) as *L*, the hidden size as *H*, and the number of self-attention heads as *A*.[3] We primarily report results on two model sizes: **BERT$_{BASE}$** (L=12, H=768, A=12, Total Parameters=110M) and **BERT$_{LARGE}$** (L=24, H=1024, A=16, Total Parameters=340M).

BERT$_{BASE}$ was chosen to have the same model size as OpenAI GPT for comparison purposes. Critically, however, the BERT Transformer uses bidirectional self-attention, while the GPT Transformer uses constrained self-attention where every token can only attend to context to its left.[4]

Input/Output Representations

To make BERT handle a variety of down-stream tasks, our input representation is able to unambiguously represent both a single sentence and a pair of sentences (e.g., ⟨ Question, Answer ⟩) in one token sequence. Throughout this work, a "sentence" can be an arbitrary span of contiguous text, rather than an actual linguistic sentence. A "sequence" refers to the input token sequence to BERT, which may be a single sentence or two sentences packed together.

We use WordPiece embeddings (Wu et al., 2016) with a 30,000 token vocabulary. The first token of every sequence is always a special classification token ([CLS]). The final hidden state corresponding to this token is used as the aggregate sequence representation for classification tasks. Sentence pairs are packed together into a single sequence. We differentiate the sentences in two ways. First, we separate them with a special token ([SEP]). Second, we add a learned embedding to every token indicating whether it belongs to sentence A or sentence B. As shown in Figure 1, we denote input embedding as E, the final hidden vector of the special [CLS] token as $C \in \mathbb{R}^H$, and the final hidden vector for the i^{th} input token as $T_i \in \mathbb{R}^H$.

For a given token, its input representation is constructed by summing the corresponding token, segment, and position embeddings. A visualization of this construction can be seen in Figure 2.

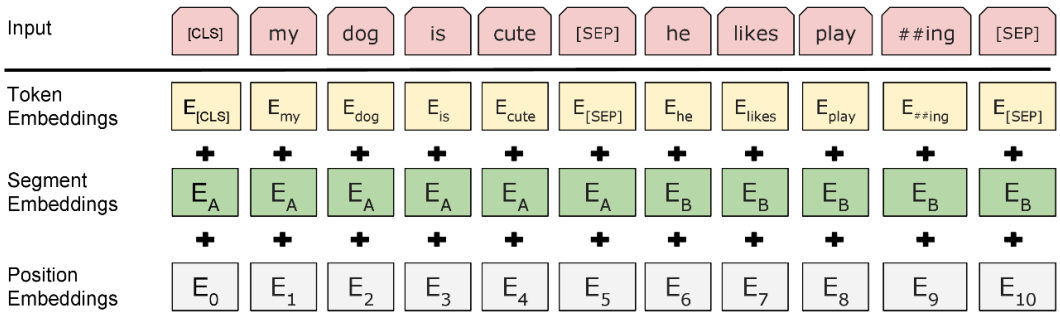

Figure 2:BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

3.1 Pre-training BERT

Unlike Peters et al. (2018a) and Radford et al. (2018), we do not use traditional left-to-right or right-to-left language models to pre-train BERT. Instead, we pre-train BERT using two unsupervised tasks, described in this section. This step is presented in the left part of Figure 1.

Task #1: Masked LM

Intuitively, it is reasonable to believe that a deep bidirectional model is strictly more powerful than either a left-to-right model or the shallow concatenation of a left-to-right and a right-to-left model. Unfortunately, standard conditional language models can only be trained left-to-right *or* right-to-left, since bidirectional conditioning would allow each word to indirectly "see itself", and the model could trivially predict the target word in a multi-layered context.

In order to train a deep bidirectional representation, we simply mask some percentage of the input tokens at random, and then predict those masked tokens. We refer to this procedure as a "masked LM" (MLM), although it is often referred to as a *Cloze* task in the literature (Taylor, 1953). In this case, the final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocabulary, as in a standard LM. In all of our experiments, we mask 15% of all WordPiece tokens in each sequence at random. In contrast to denoising auto-encoders (Vincent et al., 2008), we only predict the masked words rather than reconstructing the entire input.

Although this allows us to obtain a bidirectional pre-trained model, a downside is that we are creating a mismatch between pre-training and fine-tuning, since the [MASK] token does not appear during fine-tuning. To mitigate this, we do not always replace "masked" words with the actual [MASK] token. The training data generator chooses 15% of the token positions at random for prediction. If

the *i*-th token is chosen, we replace the *i*-th token with (1) the [MASK] token 80% of the time (2) a random token 10% of the time (3) the unchanged *i*-th token 10% of the time. Then, T_i will be used to predict the original token with cross entropy loss. We compare variations of this procedure in Appendix C.2.

Task #2: Next Sentence Prediction (NSP)

Many important downstream tasks such as Question Answering (QA) and Natural Language Inference (NLI) are based on understanding the *relationship* between two sentences, which is not directly captured by language modeling. In order to train a model that understands sentence relationships, we pre-train for a binarized *next sentence prediction* task that can be trivially generated from any monolingual corpus. Specifically, when choosing the sentences A and B for each pre-training example, 50% of the time B is the actual next sentence that follows A (labeled as IsNext), and 50% of the time it is a random sentence from the corpus (labeled as NotNext). As we show in Figure 1, *C* is used for next sentence prediction (NSP).[5] Despite its simplicity, we demonstrate in Section 5.1 that pre-training towards this task is very beneficial to both QA and NLI. [6] The NSP task is closely related to representation-learning objectives used in Jernite et al. (2017) and Logeswaran and Lee (2018). However, in prior work, only sentence embeddings are transferred to down-stream tasks, where BERT transfers all parameters to initialize end-task model parameters.

Pre-training data

The pre-training procedure largely follows the existing literature on language model pre-training. For the pre-training corpus we use the BooksCorpus (800M words) (Zhu et al., 2015) and English Wikipedia (2,500M words). For Wikipedia we extract only the text passages and ignore lists, tables, and headers. It is critical to use a document-level corpus rather than a shuffled sentence-level

corpus such as the Billion Word Benchmark (Chelba et al., 2013) in order to extract long contiguous sequences.

3.2 Fine-tuning BERT

Fine-tuning is straightforward since the self-attention mechanism in the Transformer allows BERT to model many downstream tasks—whether they involve single text or text pairs—by swapping out the appropriate inputs and outputs. For applications involving text pairs, a common pattern is to independently encode text pairs before applying bidirectional cross attention, such as Parikh et al. (2016); Seo et al. (2017). BERT instead uses the self-attention mechanism to unify these two stages, as encoding a concatenated text pair with self-attention effectively includes *bidirectional* cross attention between two sentences.

For each task, we simply plug in the task-specific inputs and outputs into BERT and fine-tune all the parameters end-to-end. At the input, sentence A and sentence B from pre-training are analogous to (1) sentence pairs in paraphrasing, (2) hypothesis-premise pairs in entailment, (3) question-passage pairs in question answering, and (4) a degenerate text-$\varnothing$ pair in text classification or sequence tagging. At the output, the token representations are fed into an output layer for token-level tasks, such as sequence tagging or question answering, and the [CLS] representation is fed into an output layer for classification, such as entailment or sentiment analysis.

Compared to pre-training, fine-tuning is relatively inexpensive. All of the results in the paper can be replicated in at most 1 hour on a single Cloud TPU, or a few hours on a GPU, starting from the exact same pre-trained model.[7] We describe the task-specific details in the corresponding subsections of Section 4.

4 Ablation Studies

In this section, we perform ablation experiments over a number of facets of BERT in order to better understand their relative importance.

4.1 Effect of Pre-training Tasks

We demonstrate the importance of the deep bidirectionality of BERT by evaluating two pre-training objectives using exactly the same pre-training data, fine-tuning scheme, and hyperparameters as $BERT_{BASE}$:

No NSP: A bidirectional model which is trained using the "masked LM" (MLM) but without the "next sentence prediction" (NSP) task.

LTR & No NSP: A left-context-only model which is trained using a standard Left-to-Right (LTR) LM, rather than an MLM. The left-only constraint was also applied at fine-tuning, because removing it introduced a pre-train/fine-tune mismatch that degraded downstream performance. Additionally, this model was pre-trained without the NSP task. This is directly comparable to OpenAI GPT, but using our larger training dataset, our input representation, and our fine-tuning scheme.

	Dev Set				
Tasks	MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
$BERT_{BASE}$	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

Table 5: Ablation over the pre-training tasks using the $BERT_{BASE}$ architecture. "No NSP" is trained without the next sentence prediction task. "LTR & No NSP" is trained as a left-to-right LM without the next sentence prediction, like OpenAI

GPT. “+ BiLSTM” adds a randomly initialized BiLSTM on top of the “LTR + No NSP” model during fine-tuning.

We first examine the impact brought by the NSP task. In Table 5, we show that removing NSP hurts performance significantly on QNLI, MNLI, and SQuAD 1.1. Next, we evaluate the impact of training bidirectional representations by comparing “No NSP” to “LTR & No NSP”. The LTR model performs worse than the MLM model on all tasks, with large drops on MRPC and SQuAD.

For SQuAD it is intuitively clear that a LTR model will perform poorly at token predictions, since the token-level hidden states have no right-side context. In order to make a good faith attempt at strengthening the LTR system, we added a randomly initialized BiLSTM on top. This does significantly improve results on SQuAD, but the results are still far worse than those of the pre-trained bidirectional models. The BiLSTM hurts performance on the GLUE tasks.

We recognize that it would also be possible to train separate LTR and RTL models and represent each token as the concatenation of the two models, as ELMo does. However: (a) this is twice as expensive as a single bidirectional model; (b) this is non-intuitive for tasks like QA, since the RTL model would not be able to condition the answer on the question; (c) this it is strictly less powerful than a deep bidirectional model, since it can use both left and right context at every layer.

4.2 Effect of Model Size

In this section, we explore the effect of model size on fine-tuning task accuracy. We trained a number of BERT models with a differing number of layers, hidden units, and attention heads, while otherwise using the same hyperparameters and training procedure as described previously.

Results on selected GLUE tasks are shown in Table 6. In this table, we report the average Dev Set accuracy from 5 random restarts of fine-tuning. We can see that larger models lead to a strict accuracy improvement across all four datasets, even for MRPC which only has 3,600 labeled training examples, and is substantially different from the pre-training tasks. It is also perhaps surprising that we are able to achieve such significant improvements on top of models which are already quite large relative to the existing literature. For example, the largest Transformer explored in Vaswani et al. (2017) is (L=6, H=1024, A=16) with 100M parameters for the encoder, and the largest Transformer we have found in the literature is (L=64, H=512, A=2) with 235M parameters (Al-Rfou et al., 2018). By contrast, $\text{BERT}_{\text{BASE}}$ contains 110M parameters and $\text{BERT}_{\text{LARGE}}$ contains 340M parameters.

Hyperparams				Dev Set Accuracy		
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

Table 6: Ablation over BERT model size. #L = the number of layers; #H = hidden size; #A = number of attention heads. “LM (ppl)” is the masked LM perplexity of held-out training data.

It has long been known that increasing the model size will lead to continual improvements on large-scale tasks such as machine translation and language modeling, which is demonstrated by the LM perplexity of held-out training data shown in Table 6. However, we believe that this is the first work to demonstrate convincingly that scaling to extreme model sizes also leads to large

improvements on very small scale tasks, provided that the model has been sufficiently pre-trained. Peters et al. (2018b) presented mixed results on the downstream task impact of increasing the pre-trained bi-LM size from two to four layers and Melamud et al. (2016) mentioned in passing that increasing hidden dimension size from 200 to 600 helped, but increasing further to 1,000 did not bring further improvements. Both of these prior works used a feature-based approach — we hypothesize that when the model is fine-tuned directly on the downstream tasks and uses only a very small number of randomly initialized additional parameters, the task-specific models can benefit from the larger, more expressive pre-trained representations even when downstream task data is very small.

4.3 Feature-based Approach with BERT

All of the BERT results presented so far have used the fine-tuning approach, where a simple classification layer is added to the pre-trained model, and all parameters are jointly fine-tuned on a downstream task. However, the feature-based approach, where fixed features are extracted from the pre-trained model, has certain advantages. First, not all tasks can be easily represented by a Transformer encoder architecture, and therefore require a task-specific model architecture to be added. Second, there are major computational benefits to pre-compute an expensive representation of the training data once and then run many experiments with cheaper models on top of this representation.

In this section, we compare the two approaches by applying BERT to the CoNLL-2003 Named Entity Recognition (NER) task (Tjong Kim Sang and De Meulder, 2003). In the input to BERT, we use a case-preserving WordPiece model, and we include the maximal document context provided by the data. Following standard practice, we formulate this as a tagging task but do not use a CRF layer in the output. We use the representation of the first sub-token as the input to the token-level classifier over the NER label set.

To ablate the fine-tuning approach, we apply the feature-based approach by extracting the activations from one or more layers *without* fine-tuning any parameters of BERT. These contextual embeddings are used as input to a randomly initialized two-layer 768-dimensional BiLSTM before the classification layer.

Results are presented in Table 7. $BERT_{LARGE}$ performs competitively with state-of-the-art methods. The best performing method concatenates the token representations from the top four hidden layers of the pre-trained Transformer, which is only 0.3 F1 behind fine-tuning the entire model. This demonstrates that BERT is effective for both fine-tuning and feature-based approaches.

System	Dev F1	Test F1
ELMo (Peters et al., 2018a)	95.7	92.2
CVT (Clark et al., 2018)	-	92.6
CSE (Akbik et al., 2018)	-	93.1
Fine-tuning approach		
$BERT_{LARGE}$	96.6	92.8
$BERT_{BASE}$	96.4	92.4
Feature-based approach ($BERT_{BASE}$)		
Embeddings	91.0	-
Second-to-Last Hidden	95.6	-
Last Hidden	94.9	-
Weighted Sum Last Four Hidden	95.9	-
Concat Last Four Hidden	96.1	-
Weighted Sum All 12 Layers	95.5	-

Table 7: CoNLL-2003 Named Entity Recognition results. Hyperparameters were selected using the Dev set. The reported Dev and Test scores are averaged over 5 random restarts using those hyperparameters.

5 Conclusion

Recent empirical improvements due to transfer learning with language models have demonstrated that rich, unsupervised pre-training is an integral part of many language understanding systems. In particular, these results enable even low-resource tasks to benefit from deep unidirectional architectures. Our major contribution is further generalizing these findings to deep *bidirectional* architectures, allowing the same pre-trained model to successfully tackle a broad set of NLP tasks.

附录 2 中文翻译

BERT：用于语言理解的深度双向 Transformer 预训练模型

Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova

Google AI Language

{jacobdevlin,mingweichang,kentonl,kristout}@google.com

摘要

我们引入一个新的语言表示模型 BERT，即 idirectional Encoder Representations from Transformers。 与最近的语言表示模型（Peters 等人，2018a；Radford 等人，2018）不同，BERT 的设计是通过对所有层中的左右上下文进行联合调节，从未标注的文本中预训练深度双向表示。 因此，只需使用一个额外的输出层就可以对经过预训练的 BERT 模型进行微调，以创建适用于各种任务（例如问题解答和语言推论）的最新模型，而无需大量与特定任务相关的结构改动。BERT 在概念上很简单，在实际应用中非常强大。 它在 11 种自然语言处理任务上获得了最新的最优结果，包括将 GLUE 得分提高到 80.5%（提升绝对值 7.7%），MultiNLI 准确度达到 86.7%（提升绝对值 4.6%），SQuAD v1.1 问答测试 F1 达到 93.2（提升绝对值 1.5 个点）以及 SQuAD v2.0 测试集 F1 达到 83.1（提升绝对值 5.1 个点）。

1 简介

语言模型预训练已经被证明可以有效地改善许多自然语言处理任务（Dai 和 Le，2015； Peters 等人，2018a； Radford 等人，2018； Howard 和 Ruder，2018）。 这些任务包括句子级任务和词符级任务；句子级任务如自然语言推理（Bowman 等人，2015；Williams 等人，2018）和释义（Dolan 和 Brockett，2005）旨在通过对句子进行整体分析来预测句子之间的关系，词符级任务如命

名实体识别和问题回答要求模型在词符级产生细粒度的输出（Tjong Kim Sang 和 De Meulder，2003；Rajpurkar 等人，2016）。

有两种现有的策略可以将预先训练好的语言表示应用到下游任务中：*基于特征*和*微调*。基于特征的方法，如 ELMo（Peters 等人，2018a），使用特定于任务的架构，将预先训练好的表示作为额外的特征。微调方法，如生成式预训练变换器（OpenAI GPT）（Radford 等人，2018），引入最少的任务相关的参数，通过简单的微调*所有*预训练的参数，对下游任务进行训练。这两种方法在预训练期间具有相同的目标函数，它们使用单向语言模型来学习通用语言表示形式。

我们认为，当前的技术限制了预训练表示的能力，特别是对于微调方法。主要的限制是标准语言模型是单向的，并且这限制了可以在预训练期间使用的结构的选择。例如，在 OpenAI GPT 中，作者使用从左到右的结构，每个词符只能在 Transformer 的自关注层中关注之前的词符（Vaswani 等人，2017 ）。这种限制对于句子级的任务来说不是最优的，而且在应用基于微调的方法来处理词符级任务如问题回答时可能是非常有害的，因为在这种情况下从两个方向结合上下文至关重要。

在本文中，我们通过提出 BERT 对基于微调的方法进行改进：**B**idirectional **E**ncoder **R**epresentations from **T**ransformers。BERT 受 Cloze 任务(Taylor，1953)的启发，通过使用"屏蔽语言模型"（masked language model，MLM）预训练目标，缓解了前面提到的单向性约束。屏蔽语言模型从输入中随机屏蔽部分词符，目标是仅根据上下文预测屏蔽掉的单词的原始词汇 ID。与从左到右的语言模型预训练不同，MLM 目标使表示形式能够融合左右上下文，让我们可以预训练深层双向 Transformer。除了屏蔽语言模型，我们还使用“预测下一个句子”任务来联合预训练文本对的表示。本文的贡献如下：

- 我们演示了双向预训练对于语言表示的重要性。不同于 Radford 等人（2018），它使用单向语言模型进行预训练，BERT 使用屏蔽语言模型

来实现预训练的深度双向表示。 这也不同于 Peters 等人 （2018a），它使用经过独立训练的从左到右和从右到左 LM 的浅层连接。

- 我们表明，经过预训练的表示可以减少许多精心设计特定于任务的结构的需求。 BERT 是第一个在大量句子级*以及*词符级任务上实现最先进性能的基于微调的表示模型，优于许多特定于任务的结构。
- BERT 推进了 11 项 NLP 任务的最先进结果。 可以在 https://github.com/google-research/bert 上找到代码和经过预先训练的模型。

2 相关工作

预先训练通用语言表示形式已有很长的历史，本节中我们简要回顾使用最广泛的方法。

2.1 基于特征的无监督方法

几十年来，学习可广泛应用的单词表示方法一直是一个活跃的研究领域，包括非神经网络方法（Brown 等人，1992；Ando 和 Zhang，2005；Blitzer 等人，2006）和神经网络方法（Mikolov 等人，2013； Pennington 等人，2014）。 预训练的词嵌入是现代 NLP 系统不可或缺的一部分，与从零开始学习的嵌入相比有显着改进（Turian 等人，2010）。 为了预训练词嵌入向量，已使用从左到右的语言建模目标（Mnih 和 Hinton，2009），以及在左右上下文中区分正确单词和错误单词的目标（Mikolov 等人，2013）。

这些方法已经推广到更粗粒度，例如句子嵌入（Kiros 等人，2015；Logeswaran 和 Lee，2018）或段落嵌入（Le 和 Mikolov，2014）。 为了训练句子表示，以前的工作使用对候选的下一句进行评分排序（Jernite 等人，2017；Logeswaran 和 Lee，2018）的目标，给定前一个句子的表示从左到右生成下一个句子的单词（Kiros 等人，2015）或去噪自动编码器派生的目标（Hill 等人，2016）。

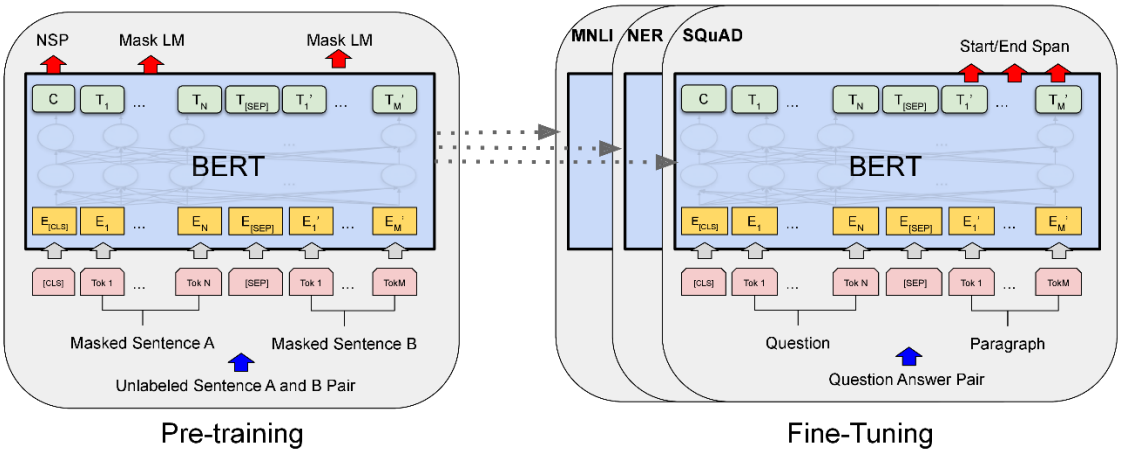

图 1：BERT 的总体预训练和微调程序。 除了输出层，在预训练和微调中都使用相同的体系结构。 相同的预训练模型参数用于初始化不同下游任务的模型。在微调期间，所有参数都将进行微调。 [CLS]是在每个输入示例前添加的特殊符号，而[SEP]是特殊的分隔符（例如，分隔问题/答案）。

ELMo 及其前身（Peters 等人，2017，2018a）沿不同维度推广了传统单词嵌入的研究。 他们结合从左到右和从右到左的语言模型提取*上下文相关*特征。每个词符的上下文表示是从左至右和从右至左表示的首尾相连。 当将上下文词嵌入与现有的特定于任务的体系结构集成时，ELMo 改进了几种主要的 NLP 基准的最先进结果（Peters 等人，2018a），包括问题回答（Rajpurkar 等人，2016）、情感分析（Socher 等人，2013）和命名实体识别（Tjong Kim Sang 和 De Meulder，2003）。 Melamud 等人 （2016）提出通利用 LSTM 从左右上下文中预测单个单词的任务学习上下文表征。 与 ELMo 相似，它们的模型是基于特征的并且不是深度双向的。 Fedus 等人 （2018）表明，cloze 任务可用于提高文本生成模型的鲁棒性。

2.2 无监督微调方法

与基于特征的方法一样，在这个方向上的早期工作只是在未标记的文本上预训练单词的嵌入参数（Collobert 和 Weston，2008）。

最近，产生上下文词符表示的句子或文档编码器已经从未标记的文本中预训练，并针对有监督的下游任务进行微调（Dai 和 Le，2015；Howard 和 Ruder，

2018；Radford 等人，2018）。 这些方法的优点是几乎不需要从头学习参数。至少部分由于此优势，OpenAI GPT（Radford 等人，2018）在 GLUE 基准测试中许多句子级任务上获得了以前的最先进结果（Wang 等人，2018a）。 从左到右的语言建模和自动编码器目标已被用于预训练此类模型（Howard 和 Ruder，2018；Radford 等人，2018；Dai 和 Le，2015）。

2.3 从监督数据迁移学习

也有工作显示了从大数据集监督任务的有效转移，如自然语言推理（Conneau 等人，2017）和机器翻译（McCann 等人，2017）。 计算机视觉研究还证明了从大型预训练模型进行迁移学习的重要性，其中一个有效的方法是微调用 ImageNet 预训练的模型（Deng 等人，2009； Yosinski 等人，2014）。

3 BERT

我们将在本节中介绍 BERT 及其详细实现。 我们的框架有两个步骤：*预训练*和*微调*。 在预训练期间，通过不同的预练任务在未标记的数据上进行模型训练。 对于微调，首先使用预训练的参数初始化 BERT 模型，然后使用下游任务中的标记数据对所有参数进行微调。 每个下游任务都有单独的微调模型，即使它们使用相同的预训练参数进行了初始化。 图 1 中的问答示例将作为本节的运行示例。

BERT 的一个独有的特征是其跨不同任务的统一结构。 预训练的结构和最终的下游结构之间的差异很小。

模型结构

BERT 的模型结构是一种多层 Transformer 编码器，它基于的原始实现的描述位于 Vaswani 等人 （2017）并发布在 tensor2tensor 库中。[1] 因为 Transformer 的使用已经很普遍以及我们的实现与原始版本几乎相同，我们将省略模型结构的详尽背景说明并请读者参考 Vaswani 等人 （2017）以及优秀的指南如“The Annotated Transformer”。[2]

在这项工作中，我们将网络层（即 Transformer 的网络模块）的数量表示为 L，将隐藏层大小表示为 H，并将自注意头的数量表示为 A。[3] 我们主要报告两种模型大小的结果：$\mathbf{BERT}_{\mathrm{BASE}}$ (L=12, H=768, A=12, Total Parameters=110M) 和 $\mathbf{BERT}_{\mathrm{LARGE}}$ (L=24, H=1024, A=16, Total Parameters=340M) 。

为了进行比较，选择 $\mathrm{BERT}_{\mathrm{BASE}}$ 具有与 OpenAI GPT 相同的模型大小。 但是，至关重要的是，BERT Transformer 使用双向自关注，而 GPT Transformer 使用受限的自我关注，其中每个词符只能关注其左侧的上下文。[4]

输入/输出表示

为了使 BERT 处理各种下游任务，我们的输入可以用一个词符序列明确地表示单个句子和一对句子（例如⟨Question, Answer⟩）。 在整个工作中，“句子”可以是连续文本的任意范围，而不是实际的语言句子。 “序列”是指 BERT 的输入词符序列，它可以是一个句子或两个句子封装在一起。

我们使用 WordPiece 嵌入（Wu 等人，2016），具有 30,000 个词符的词汇表。 每个序列的第一个标记始终是特殊分类标记（[CLS]）。 与此标记对应的最终隐藏状态用作分类任务的聚合序列表示。 句子对封装在一起形成单个序列。 我们通过两种方式区分句子。 首先，我们使用特殊词符（[SEP]）将它们分开。 其次，我们向每个词符添加一个学习型嵌入，表明它是属于句子 A 还是句子 B。 如图 1 所示，我们将输入嵌入表示为 E，将特殊的 [CLS] 词符的最终隐藏向量表示为 $C \in \mathbb{R}^H$，第 i 个输入词符为的最终隐藏向量为 $T_i \in \mathbb{R}^H$。

对于给定的词符，其输入表示是通过将相应的词符嵌入、分段嵌入和位置嵌入相加来构造的。 在图 2 中可以看到这种构造的可视化。

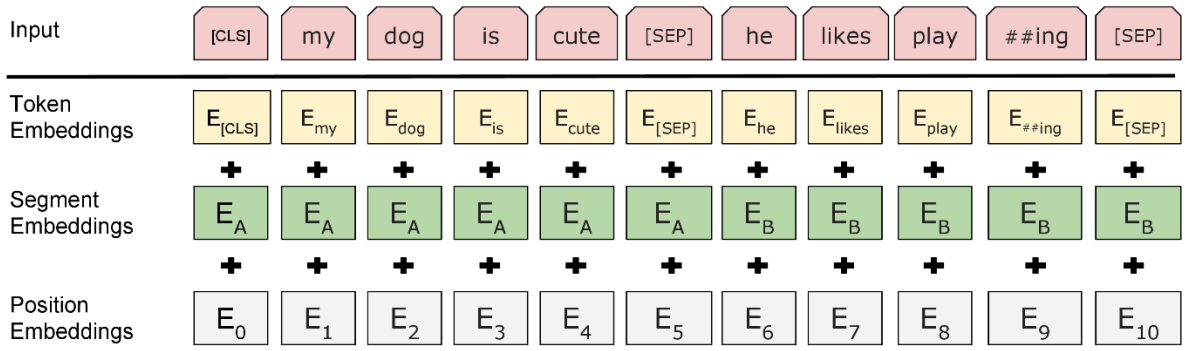

图 2：BERT 输入表示。 输入的嵌入是词符嵌入、分段嵌入和位置嵌入的总和。

3.1 预训练 BERT

不同于 Peters 等人 （2018a）和 Radford 等人 （2018），我们没有使用传统的从左到右或从右到左的语言模型对 BERT 进行预训练。 相反，我们使用本节中描述的两个无监督任务对 BERT 进行预训练。 该步骤显示在图 1 的左侧。

任务＃1：屏蔽 LM

直观地讲，我们有理由相信，一个深层的双向模型严格来说比从左到右的模型或从左到右和从右到左的浅层模型连接更强大。 不幸的是，标准的条件语言模型只能从左至右*或*从右至左进行训练，因为双向条件会让每个词间接地"看到自己"，从而模型在多层上下文中可以十分容易地预测目标词。

为了训练深度双向表示，我们简单地随机屏蔽一定百分比的输入词符，然后预测这些屏蔽的词符。 尽管此过程在文献中通常被称为 *Cloze* 任务（Taylor，1953），但我们将该过程称为“屏蔽 LM”（MLM）。 这种情况下，和在标准 LM 中一样，与屏蔽词符相对应的最终隐藏向量被送入词汇表上的输出 softmax 中。 在所有实验中，我们随机屏蔽每个序列中所有 WordPiece 词符的 15％。 与去噪自动编码器（Vincent 等人，2008）不同，我们只预测被屏蔽的单词，而不重构整个输入。

尽管这可以使我们获得双向的预训练模型，但缺点是我们在预训练和微调之间造成了不一致，因为 [MASK] 词符不会在微调期间出现。 为了缓解这种情况，实际上我们并不总是用 [MASK] 词符替换“被屏蔽”的单词。 训练数据生成器随机选择词符位置的 15％ 进行预测。 如果选择第 i 个词符，我们替换这

第 i 个词符为（1）80% 的时间用 [MASK] 词符（2）10% 的时间用随机词符（3）10% 的时间维持第 i 词符不变。 然后，将用 T_i 以交叉熵损失预测原始词符。 我们在附录 C.2 中比较了此过程的各种变体。

任务 #2：下一句预测（NSP）

许多重要的下游任务，例如问答（QA）和自然语言推理（NLI）是是基于理解两个句子之间的*关系*，而这不是语言建模直接捕获的。 为了训练能够理解句子关系的模型，我们预训练了可以从任何单语语料库很容易生成的二值 *下一个句子预测* 任务。 具体来说，当为每个预训练样本选择句子 A 和 B 时，50% 的时间 B 是紧随其后的实际下一个句子 A（标记为 IsNext），50% 的时间是语料库中的随机句子（标记为 NotNext）。 如图 1 所示，C 用于下一句预测（NSP）。[5]，尽管非常简单，我们在 5.1 节中展示此任务的预训练对 QA 和 NLI 都非常有益。 [6] NSP 任务与表示学习目标很接近，参见 Jernite 等人（2017）和 Logeswaran 和 Lee（2018）。 但是，在之前的工作中，只有句子嵌入被传输到下游任务，而 BERT 传输所有参数以初始化最终任务模型的参数。

预训练数据

预训练过程在很大程度上遵循有关语言模型预训练的现有文献。 对于预训练语料库，我们使用 BooksCorpus（800 万个单词）（Zhu 等人，2015）和英语 Wikipedia（25 亿个单词）。 对于 Wikipedia，我们仅提取文本段落，而忽略列表、表格和标题。 为了提取长的连续序列，使用文档级语料库而不是洗乱的句子级语料库如 Billion Word Benchmark（Chelba 等人，2013）至关重要。

3.2 微调 BERT

微调很简单，因为 Transformer 中的自我关注机制允许 BERT 通过交换适当的输入和输出来建模许多下游任务（无论它们涉及单个文本还是文本对）。 对于涉及文本对的应用，常见的模式是在应用双向交叉注意之前对文本对进行独立编码，例如 Parikh 等人 （2016）； Seo 等人 （2017）。 BERT 使用自我

注意机制来统一这两个阶段，因为使用自我注意编码连接的文本对行实际上包括了两个句子之间的*双向*交叉注意。

对于每个任务，我们只需将特定于任务的输入和输出送入 BERT，并端到端微调所有参数。 在输入处，来自预训练的句子 A 和句子 B 类比于（1）paraphrasing 中的句子对（2）entailment 中的假设-前提对（3）question answering 中使用的 question-passage 对，以及（4）在文本分类或序列标记中使用退化的 text-∅ 对。 在输出处，将词符表示输入到输出层中以进行词符级任务如序列标记或问题回答，将 [CLS] 表示输入到输出层中进行分类如蕴含或情感分析。

与预训练相比，微调代价相对较小。 从完全相同的预训练模型开始，论文中的所有结果都可以复现，在单个 Cloud TPU 上最多 1 个小时，在 GPU 上最多需要几个小时。[7] 我们在第 4 节的相应小节中描述特定任务的细节。**表格 1：** GLUE 测试结果，由评估服务器评分。 每个任务下方的数字表示训练样本的数量。 “平均”列与官方 GLUE 得分略有不同，因为我们排除了有问题的 WNLI 集。[8] BERT 和 OpenAI GPT 是单模型、单任务。 QQP 和 MRPC 报告的是 F1 分数，STS-B 报告的是 Spearman 相关性，其他任务报告的是准确率分数。 我们不包括使用 BERT 作为其组件之

4 细分研究

在本节中，我们将对 BERT 的多个方面进行细分实验，以更好地了解它们的相对重要性。 其它细分研究可在附录 C 中找到。

4.1 预训练任务的效果

我们通过使用与 $\text{BERT}_{\text{BASE}}$ 完全相同的预训练数据、微调方案和超参数来评估两个预训练目标，证明了 BERT 深度双向性的重要性。

No NSP： 使用 "屏蔽 LM"（MLM）训练的双向模型，但没有 "下一句预测"（NSP）任务。

LTR & No NSP：一个仅有左侧上下文的模型，它是用标准的从左到右（LTR）LM，而不是 MLM 训练。 左约束也应用于微调，因为删除它会引入预训练/微调不匹配，从而降低下游性能。 此外，该模型无需 NSP 任务即可进行预训练。 这可以直接与 OpenAI GPT 相提并论，但要使用更大的训练数据集，输入表示形式和微调方案。

任务	MNLI-m（Acc）	QNLI（Acc）	MRPC（Acc）	SST-2（Acc）	SQuAD（F1）
$BERT_{BASE}$	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

表 5：使用 $BERT_{BASE}$ 架构细分预训练任务。 “No NSP”使用没有下一个句子预测任务进行训练。 “LTR & No NSP”以从左到右的 LM 训练，没有下一个句子预测，和 OpenAI GPT 一样。 在微调期间，“+ BiLSTM”在“LTR + No NSP”模型的顶部添加一个随机初始化的 BiLSTM。

我们首先研究 NSP 任务带来的影响。 在表 5 中，我们显示删除 NSP 会严重损害 QNLI、MNLI 和 SQuAD 1.1 的性能。 接下来，我们通过比较“No NSP”与“LTR & No NSP”来评估训练双向表示的影响。 在所有任务上，LTR 模型的性能都比 MLM 模型差，而 MRPC 和 SQuAD 的性能下降很大。

对于 SQuAD，直观上很清楚，因为词符级别的隐藏状态没有右侧上下文，所以 LTR 模型在词符预测时的性能会很差。 为了使 LTR 系统得到加强，我们在上面添加了一个随机初始化的 BiLSTM。 这确实可以显着改善 SQuAD 上的结果，但结果仍然比预训练的双向模型的结果差很多。 BiLSTM 损害了 GLUE 任务的性能。

我们认识到，也有可能像 ELMo 一样训练单独的 LTR 和 RTL 模型并将每个词符表示为两个模型的连接。 但是：（a）代价是单个双向模型的两倍；（b）

对于 QA 这样的任务，这是不直观的，因为 RTL 模型将无法确定问题的答案；（c）这绝对不像深度双向模型那么强大，因为它可以在每一层使用左右上下文。

4.2 模型大小的影响

在本节中，我们探索模型大小对微调任务准确性的影响。 我们训练了许多具有不同层数，隐藏单元和注意头的 BERT 模型，而其他方面则使用了与之前所述相同的超参数和训练过程。

表 6 中显示了选定的 GLUE 任务的结果。 在此表中，我们报告了 5 次随机微调重新启动后的平均 Dev Set 精度。 我们可以看到较大的模型使得所有四个数据集的准确性提高，即使对于只有 3,600 个带标签的训练样本且与预训练任务大不相同的 MRPC 也是一样。 我们能够在相对于现有文献而言已经相当大的模型的基础上实现如此显着的改进，这也许也令人惊讶。 例如，在 Vaswani 等人 （2017）（L = 6、H = 1024、A = 16）参数为 100M 的编码器，我们在文献中找到的最大 Transformer 是（L = 64、H = 512、A = 2）具有 235M 个参数（Al-Rfou 等人，2018）。 相比之下，$BERT_{BASE}$ 包含 110M 参数，而 $BERT_{LARGE}$ 包含 340M 参数。

超参数				开发集准确率		
#L	#H	#A	LM（ppl）	MNLI-m	MRPC	SST-2
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

表 6： BERT 模型大小的分解。 #L = 层数；#H = 隐藏大小； #A = 注意头的数量。“ LM（ppl）”是保留的训练数据的屏蔽 LM 困惑度。

众所周知，增加模型大小将导致大规模任务如机器翻译和语言建模的不断改进，表 6 所示的 LM 对持有的训练数据的迷惑性就证明了这一点。 但是，我们认为，这是第一个有说服力的工作，证明只要模型已经过充分的预训练，将模型缩放到极端的模型大小也可以在非常小的规模的任务上带来很大的改进。 Peters 等人 （2018b）公开了将预训练的 bi-LM 大小从两层增加到四层在下游任务上的混合结果以及 Melamud 等人 （2016）顺便提到了将隐藏大小从 200 增加到 600 有帮助，但进一步增加到 1000 并没有带来进一步的改进。 这两个先前的工作都使用基于特征的方法-我们假设当直接在下游任务上微调模型并且仅使用很少数量的随机初始化的附加参数时，特定于任务的模型可以从较大的模型中受益，即使下游任务数据非常小，也可以提供更具表现力的预训练表示形式。

4.3 基于特征的 BERT 方法

到目前为止，所有提出的 BERT 结果都使用微调方法，即在预训练模型中添加一个简单的分类层，在下游任务上对所有参数进行共同微调。 但是，基于特征的方法，即从预训练的模型中提取固定的特征，也有一定的优势。 首先，并非所有任务都可以由 Transformer 编码器结构轻松表示，因此需要添加特定于任务的模型结构。 其次，预计算一次训练数据的昂贵表示，然后在这个表示之上用更便宜的模型运行许多实验，有很大的计算优势。

在本节中，我们通过将 BERT 应用于 CoNLL-2003 命名实体识别（NER）任务（Tjong Kim Sang 和 De Meulder，2003）来比较这两种方法。 在 BERT 的输入中，我们使用一个大小写保留的 WordPiece 模型，并且最大程度包含数据提供的文档上下文。 按照标准惯例，我们将其公式化为标记任务，但在输出中不使用 CRF 层。 我们使用第一个子词符的表示作为 NER 标签集上词符级分类器的输入。

为了详细分解研究微调方法，我们应用基于特征的方法，在*没有*微调 BERT 任何参数的情况下，提取一个或多个层的激活。 这些上下文嵌入用作分类层之前的随机初始化的两层 768 维 BiLSTM 的输入。

结果显示在表 7 中。 $\text{BERT}_{\text{LARGE}}$ 与最先进的方法相比具有竞争力。 表现最好的方法将来自预训练的 Transformer 的顶部四个隐藏层的词符表示连接起来，这仅比微调整个模型低 0.3 F1。 这表明 BERT 对于微调和基于特征的方法均有效。

系统	开发集 F1	测试集 F1
ELMo （Peters 等人， 2018a ）	95.7	92.2
CVT （Clark 等人， 2018 ）	-	92.6
CSE （Akbik 等人， 2018 ）	-	**93.1**
微调方法		
$\text{BERT}_{\text{LARGE}}$	96.6	92.8
$\text{BERT}_{\text{BASE}}$	96.4	92.4
基于特征的方法 ($\text{BERT}_{\text{BASE}}$)		
嵌入层	91.0	-
倒数第二个隐藏层	95.6	-
最后的隐藏层	94.9	-
最近四个隐藏层的加权总和	95.9	-
最后四个隐藏层的连接	96.1	-
所有 12 层的加权总和	95.5	-

表 7：CoNLL-2003 命名实体识别结果。 超参数的选择使用开发集。 报告的开发集和测试集得分是用这些超参数在 5 次随机重启后的平均值。

5 结论

最近由于语言模型的迁移学习经验所带来的改进表明，丰富的、无监督的预训练是许多语言理解系统中不可或缺的一部分。 尤其是，这些结果使即使是资源很少的任务也可以从深度单向结构中受益。 我们的主要贡献是将这些发现进一步推广到更深的双向体系结构中，从而使相同的预训练模型能够成功解决各种 NLP 任务。