arXiv:2301.00495v1 [cs.SE] 2 Jan 2023

面向软件需求数据多类别分类的自适应微调

Savas Yildirim[1], Mucahit Cevik[1*]、Devang Parikh[2] 和 Ayse Basar[1]

[1] 多伦多都会大学，安大略省多伦多 Gerrard 街东 44 号，M5B 1G3，加拿大。[2] IBM，北卡罗来纳州卡里，27709，美国。

*通讯作者。电子邮箱：mcevik@ryerson.ca；

摘要

近年来，使用自然语言处理（NLP）方法分析软件需求规格说明（SRS）已成为软件工程领域一项重要的研究方向。尤其是得益于自然语言处理领域深度学习和迁移学习方法带来的进展，SRS数据可更轻松地用于各种学习任务。本研究针对软件需求的三项预测任务，采用三阶段领域自适应微调方法，以提升模型在真实分布偏移下的鲁棒性。这些多类分类任务旨在预测用户指定的需求文本的 *type*、 *priority* 和*severity* 。我们将结果与强大的分类基线方法进行比较，例如词嵌入池化和 Sentence BERT，并表明自适应微调能够提升模型在各项任务中的表现。我们发现，自适应微调后的模型可以针对特定数据分布进行专门化，从而生成准确的结果，并从软件工程任务管理系统中大量可用的文本数据中学习。

关键词： 软件需求规格说明（SRS）、Transformer、迁移学习、领域自适应、深度学习

1 引言

软件需求规格说明（SRS）详细描述了软件项目的功能和特性，也有助于评估项目启动时制定的标准和需求是否得到遵循。SRS数据通常以自然语言编写，用于促进项目协作团队之间的沟通。SRS语言存在一定程度的主观性，可能导致误解、矛盾、歧义和冗余。自然语言处理（NLP）方法带来的自动化能够有效避免这些问题。

NLP方法可用于促进需求工程中的各类任务，包括需求分类和提取，从而大幅节省这些任务所耗费的时间和精力。该领域的研究受益于NLP在抽取式摘要、标注和实体识别等多种任务方面的最新进展 [1]。NLP方法学的这些最新进展可归因于迁移学习，它已成为通用语言建模的最先进方法 [2]。大多数迁移学习方法依赖序列学习；由于其简单易行，序列学习一直是NLP领域的常见做法。随着语言模型通过序列学习提升泛化能力，许多NLP任务得以按序逐步完成；序列学习还能够利用先前任务中获取的信息，从而在后续任务中取得更好的性能。

文献中的大多数研究都采用两阶段迁移学习方案：预训练和微调。该方案的一个主要问题是，源数据的分布可能与目标数据的分布不同。因此，将模型从一种训练分布适配到目标任务的不同分布，可能会带来显著益处。自适应微调可以利用目标领域中的无标签文本数据来解决这一问题。因此，两阶段方案可以扩展为三个或更多阶段，在预训练步骤之后进行额外训练。这种额外训练也称为自适应预训练（APT）、预微调（PFT）和持续预训练（CPT）。

研究目标与范围

在本研究中，我们重点探讨如何利用迁移学习方法提升对软件需求规格说明数据进行文本分类的性能。具体而言，我们设计了一种自适应微调框架，利用目标领域中大量可用的无标签数据。我们使用从 IBM Rational DOORS Next Generation 产品 [3] 获得的软件需求规格说明数据开展分析。该产品是一种需求管理工具，旨在优化软件开发团队之间的协作与沟通，从而改善项目内的信息流。DOORS数据集用于三项不同的分类任务，即预测需求的优先级、严重程度和类型 。

贡献

标准迁移学习通常分为两个阶段进行。第一阶段包括从源数据中学习，第二阶段则是针对主要目标任务对源知识进行微调。然而，在现实问题中，源数据与目标数据具有相似分布这一假设可能并不成立，因为目标数据的分布可能会随着时间推移或跨领域而不同于源数据的分布，这被称为分布外（OOD）问题 [4]。解决方案是将模型从源分布自适应到目标任务的分布。在本研究中，我们展示了如何应用自适应，并利用三阶段领域自适应微调来处理软件需求数据集 DOORS 上的三个多类分类任务。我们的研究主要贡献总结如下：

- 我们利用词嵌入和 Sentence BERT 作为特征提取模型来构建强大的分析基线，其中模型参数被冻结，并将其输出作为线性分类器的输入。此外，我们针对分类任务采用了两阶段和三阶段微调机制，其中使用了 Transformer模型的基础版本（例如 BERT-base 和 RoBERTa-base）、大型版本（例如 BERT-large 和 RoBERTa-large）以及蒸馏版本。

- 我们利用额外的未标注域内数据，在域内数据上进一步预训练 Transformer 模型检查点。这是一个额外的自适应微调步骤，旨在提升文本分类性能。随后，我们再次微调这些经自适应训练的模型，以用于三个下游任务。设计一种能够提升性能的自适应微调机制，是本研究的主要创新之处，同时，本研究还提出了一个新颖的应用问题。

- 我们开展了广泛的实证研究，并详细分析了各种模型在软件需求规格说明数据集上的表现。这些分析有助于我们了解最先进的文本分类方法在软件工程重要实际问题中的优势与局限。

论文结构

本文其余部分安排如下。在第2节中，我们简要回顾与本研究最相关的工作，特别关注软件工程中的基于自然语言处理的自动化以及迁移学习的方法学进展。第3节将介绍所采用的方法，并讨论数据集，同时回顾 Transformer模型和领域适应。第4节将讨论数值研究的结果。最后，我们在第5 节总结研究发现，并探讨研究局限性和未来研究方向。

2 文献综述

在文献中，研究人员使用各种文本分类技术（如文档分类或情感分析）分析SRS数据，以通过高质量的预测改进软件开发过程。在早期研究中，传统机器学习模型成功应用于此类任务。Hussain 等人 [5] 在传统机器学习流程中采用文档分类来检测软件需求规格说明书文档中的歧义。他们训练决策树，将软件需求规格说明书文档中的段落映射为含歧义或无歧义标签，这是一项二分类任务。Zhang 等人 [6] 专注于利用情感分析提取有问题的 API设计。他们从包含大量非结构化数据的在线资源中提取有用信息，例如错误报告和一些在线讨论。Hou 等人 [7]使用朴素贝叶斯模型，根据内容对 API 讨论进行分类。机器学习已在软件需求数据集上得到广泛应用。例如，Asabadi 等人 [8]采用模糊集理论，利用歧义术语列表识别含歧义的软件需求规格说明陈述。除了机器学习方法之外，基于规则的方法也被用于软件工程领域的分类任务，例如需求分类 [9]。近年来，深度学习架构已频繁应用于需求工程中的文本分析任务。Navarro 等人 [10]采用卷积神经网络（CNN）模型对软件需求进行分类，无需使用人工设计的特征。Onyeka 等人 [11] 在隐式需求框架中利用常识知识本体，其中基于 CNN 的自编码器可从大型软件需求规格说明书文档中的表格和图像中识别隐式需求。

软件工程和需求工程领域中最新的自然语言处理相关研究受到了迁移学习的影响。预训练语言模型（PLM），尤其是Transformer的编码器部分，已成为最适合利用迁移学习的模型。在这方面，基于Transformer的双向编码器表示（BERT） [12] 已被证明非常有用，尤其适用于文本分类和词元分类问题，例如情感分析和命名实体识别。近年来，BERT的许多变体相继出现，并被用于软件分析，包括涉及SRS数据 [1, 13, 14]的分类任务。Hey等人 [13]针对特定任务对BERT进行了微调，使模型能够预测某项需求是非功能性需求还是功能性需求。在另一项研究中，Sainani等人 [14] 以不同方式使用BERT：他们首先从大型软件工程合同中提取需求，然后对这些需求进行分类。他们的方法基于这样一种理念：商业合同有助于识别高级需求，从而提升软件工程项目的绩效。Kici等人 [1]针对不同的SRS任务对多种预训练语言模型进行了微调。他们使用不同的BERT模型及其变体，在三个不同的数据集上进行测试，以验证其结果的可泛化性。

上述所有 Transformer模型均采用两阶段的直接微调流程实现。具体而言，它们解决了

无需领域特定适应，仅通过使用标注数据训练预训练模型检查点，即可完成下游任务。然而，利用大多数问题中易于获取的无标签数据进行自适应微调，可以提高迁移学习能力。自适应微调是一种领域适应技术，旨在寻找一种方法，将基于通用源数据训练的预训练模型应用于不同的目标领域 [15]。然而，源领域和目标领域可能具有不同的分布，例如词汇分布。一些研究 [16, 17] 表明，自适应能够处理先前源分布与目标分布不匹配所产生的影响。BioBERT 等领域特定模型 [18], SciBERT [19],HateBERT [20], ClinicalBERT [21],NetBERT [22], MathBERT [23], News [16]和 GraphCodeBERT [24] 已开始频繁出现在文献中。所有这些模型或检查点都试图构建更好的模型，以便在目标领域的分布不同于源领域分布的情况下仍能发挥作用。一些研究探讨了 Transformer架构中掩码率的作用。在 MLMs 预训练期间，15% 的选择通常被视为一种有效的默认掩码率 [25]。另一方面，在另一项研究中，Wettig 等人 [26]指出，将约 40% 的输入标记进行掩码，而不是 15%，可以获得更好的下游任务性能。

据我们所知，目前没有其他研究专注于软件需求领域适应。在我们的研究中，我们弥补了软件工程领域的这一研究空白，并在软件需求数据上应用自适应微调，以提升特定任务的文本分类性能。

3 方法

在深度学习模型中，神经网络架构经过训练的参数可以基于两种迁移方式进行存储，以供将来使用：特征提取和微调。前者获得的参数是固定的，例如词嵌入中的参数，不参与梯度下降；而后者获得的参数则会发生变化，例如 BERT模型中的参数。本节将简要说明我们如何采用特征提取，介绍标准的两阶段训练过程，并详细阐述自适应微调框架。

3.1 DOORS数据集

我们的数据集由使用 IBM 的下一代动态面向对象需求系统（DOORS）产品收集的软件需求规格说明组成。原始数据集包含 83,837 份文档。我们选取需求规格说明的 摘要 ，针对三个独立类别进行分类：优先级、严重程度和类型。因此，DOORS 数据集可视为任务管理数据集，与文献中的其他软件需求规格说明数据集不同（例如，参见 NFR-PROMISE [13]），

6 软件需求数据多类别分类

通常用于需求类型分类（例如，功能性与非功能性）。表1 提供了来自DOORS数据集的数据实例。

表1：DOORS数据样本

摘要	Type	优先级	严重程度
创建新链接时，默认选择必须是模块，而不是文件夹	增强	中	主要
提供便捷机制，以更新服务器和测试服务器的名称	Task	中	普通
作为开发人员，我应该能够实现对视图查询中长整型大型列表的支持	用户故事	未分配	普通
无法将制品导出为 PDF 文件	缺陷	High	主要
组件选择页面中的“所有项目”标签应更新	增强	未分配	轻微
调查自托管环境中涉及包装器资源的潜在数据损坏问题	问题	Low	普通

表2 展示了每个类别标签的类别分布。优先级(未分配、高、中和低）共有四个类别；严重程度(普通、主要、轻微、阻塞项、严重和未决定）共有六个类别。类型类别最初包含20个不同类别。经过合并和预处理操作后，我们为类型类别整理出七个类别。此外，我们删除了“nan”值，这些值对应于严重程度和优先级类别中的缺失值。值得注意的是，虽然类型类别的类别分布相对均衡，但其他类别存在严重的数据不平衡。

表2：每个类别标签的类别分布

(a) 优先级

类别	数量	百分比（%）
未分配	11,443	68.97
High	2,677	16.13
中	1,990	11.99
Low	480	2.89

（b）严重程度

类别	数量	百分比（%）
普通	14,348	86.48
主要	1,127	6.79
未决定	478	2.88
轻微	284	1.71
严重	245	1.47
阻塞项	108	0.65

(c) 类型

类别	数量	百分比（%）
增强	5,026	30.29
用户故事	4,565	27.50
维护	2,180	13.14
其他	1,774	10.69
测试任务	1,615	9.73
计划项	885	5.33
JUnit	545	3.28

完成预处理步骤后，剩余实例总数为 16,590，文本长度平均为 13.1 个词，中位数为 12。图 1 展示了按类别划分的类型标签文本长度箱线图，作为一个代表性示例。我们发现，各类别的文本长度分布为类型标签提供了一些信息（例如，用户故事类的文本最长，而 JUnit 的文本最短）。然而，对于优先级和严重程度标签，各类别之间的文本长度分布则更加均匀。

3.2 特征提取方法

通过对词的固定向量取平均来学习文本表示，已被先前的研究证明是一种简单而有效的方法。词

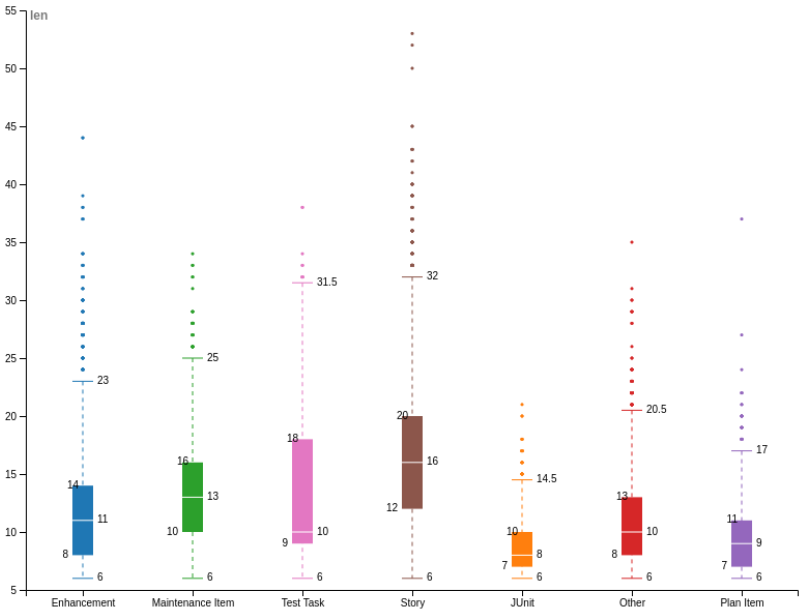


Fig. 1：DOORS 数据集中各类型标签类别的文本长度箱线图 t

嵌入技术，例如 FastText [27]，Glove [28] 和 Word2vec [29]，能够提供强大的静态词向量，这些向量在通用机器学习流程中保留下来，用于另一种目的，称为特征提取。也就是说，词向量（或文档向量）保持冻结，并作为线性分类器或聚类算法的输入。

这类通过静态词向量平均值构建的文档表示，已成为出乎意料的强大基线方法。在某些情况下，这些方法的性能甚至高于更复杂的模型 [30]。此外，它们还为判断数据中是否存在某种模式提供了依据。例如，Cer 等人 [31] 提出了一种基于平均值的模型，称为 DAN 网络，并表明该模型的性能与更复杂的架构相近。另一方面，平均方法也有一定的局限性，因为它基于固定不变的词嵌入。除了平均方法之外，人们还提出了不同的句子级方法，这些方法基于特征提取来实现文本表示，例如 Doc2vec [32]、Sentence BERT [33]，和 Skip Thought [34]。

在我们的实验中，我们采用两种特征提取方法作为强大的基线方法：词嵌入池化和 Sentence BERT。下面介绍这两种方法的设置。

- 词嵌入池化：在词嵌入池化中，我们只需对句子中的词嵌入取平均值；这些嵌入由预训练的 FastText 和 Glove 向量获取并拼接而成。最终向量的长度

为 400。在其上添加一个线性层，并通过监督式流程训练模型。

- *SBERT*: Sentence BERT 嵌入已被证明是一种高效的文本表示方法，尤其适用于句子相似度或语义搜索等语义相关问题。SBERT 方法背后的主要动机是，BERT 模型不适合单独用于词向量或句子向量，因为它需要针对下游任务进行端到端微调。基于孪生网络结构，SBERT 能够生成具有语义意义且彼此独立的句子嵌入，因此适合作为一种基线方法。

3.3 两阶段训练：预训练与微调

为了将具有参数 Θ_S 的预训练源模型微调至不同的目标任务，需设计并添加任务专用参数 Θ_T （或层）Θ_S。在特征提取中（通常应用于词嵌入）， Θ_S 会被冻结，且针对特定任务仅更新 Θ_T 参数。在微调中（应用于 ELMo [35]和 BERT 模型时 [12], ，会更新全部参数 $\Theta_S \cup \Theta_T$，通常采用端到端方式。微调期间，学习率通常设置得比预训练设置中的学习率更低，以免过于频繁地更新主模型。这是因为已有观察表明，大多数句法和语义信息已编码在预训练模型中。另一方面，一个重要问题是目标数据与源数据之间的分布差异。例如，在预训练阶段未出现、但在微调阶段出现的词会导致未登录词问题 [4]。

ELMo [35] 和Transformer模型 [36]为自然语言理解和生成问题作出了巨大贡献。它们的成功可归因于这样一个事实：这些架构通过将信息分布到深度网络的各个层中来进行编码，而不是将信息编码在一个简单的一维向量中。它们已成功应用于微调阶段，并且能够迁移先前训练阶段所获得的知识。近年来，Transformer架构 [36] 由于采用了自注意力机制，在许多任务中展现出了良好的效果。Transformer架构最重要的优势之一在于，它们能够通过可并行化的架构创建可靠的预训练模型，从而使迁移学习更快速、更有效。除了非常适合领域适应之外，Transformer模型对分布外样本 [4]也具有很强的鲁棒性。

与基于词嵌入的平均模型不同，Transformer模型采用端到端方式进行训练。在预训练的源模型 Θ_S之上添加一个薄层 Θ_T，然后将整个架构 $\Theta_S \cup \Theta_T$作为整体进行训练，这称为微调。在某些情况下，对整个架构进行微调的计算成本可能很高。该方向的一些方法采用了适配器 [37, 38]，其基本思想是通过在各层之间添加简单的可训练适配器，避免训练整个架构。使用适配器的另一个原因是，这些模型可以

会遭遇灾难性遗忘，这意味着在先前阶段学到的信息可能会在添加新任务时丢失。在我们的分析中，我们通过观察损失和性能，将这种负迁移概念纳入考量；这对于避免干扰预训练模型，以及避免遭遇灾难性遗忘和负迁移都很重要。

对于各种问题实例，微调策略已被证明优于静态特征提取。数值研究中，我们考虑了广泛使用的模型及其用于微调的检查点，简要总结如下。

- **BERT 检查点：**BERT 是最知名的编码器，同时采用掩码语言模型（MLM）和下一句预测（NSP）目标。基础模型 *BERT-base-uncased* 的隐藏层维度为 768，对应 12 个注意力头，每个注意力头的嵌入维度为 64。模型共有 12 层。较大的对应模型 *BERT-large-uncased* 的隐藏层维度为 1024，共有 24 层和 16 个注意力头。BERT 检查点使用的原始学习率为 1e-4。对其进行微调时，会选择较小的学习率（例如 1e-5 或 2e-5），以免干扰预训练期间的学习。

- **DistilBERT检查点：** 我们使用*distilbert-base-uncased* DistilBERT的检查点 [39],这是一个通过蒸馏BERT-base检查点训练得到的小型轻量模型。它在与预训练BERT-base模型相同的数据上进行预训练。该语料库由多伦多图书语料库和完整的英文维基百科组成，并在*BERT-base-uncased* 版本的监督下进行蒸馏。该模型有6层，隐藏层大小为768，配备12个头，并包含6600万个参数。

•**RoBERTa检查点**：RoBERTa（鲁棒优化的BERT预训练方法） [40] 是广为人知的BERT重新实现。与修改架构设计相比，RoBERTa训练在训练策略方面带来了更多改进。例如，NSP训练目标被移除，静态掩码被动态改变掩码模式的策略所取代。在我们的分析中，预训练期间使用Adam优化器，其方式与其他BERT检查点相同，学习率为6e-4。

3.4 自适应微调

尽管用于Transformer架构的微调方法通常表现良好，但源数据与目标数据的分布差异通常会显著影响微调的效果 [2]。如果源数据集与目标数据集彼此差异很大，微调可能难以发挥作用。通常，自然语言处理模型评估依赖于这样的假设：源（训练）实例与目标（测试）实例独立同分布；但在大多数现实世界应用中，这一假设并不成立。出现这种差异的原因可能是目标数据集难以表征整个分布，而且目标分布通常

会随时间变化 [41, 42]。以往研究表明，预训练的Transformer是在现实世界分布偏移下最稳健的架构，其泛化能力也强于其他架构，不过，仍有改进空间 [4]。

在文献中，已有多种策略被提出，用于将预训练模型适配到目标领域 [2]。在我们的实现中，我们使用一些额外的域内数据继续训练预训练模型，期望让模型专门适应目标数据，从而提升下游任务的性能。也就是说，使用预训练目标函数在目标数据（即 $J_S = J_A$）上持续训练已完成预训练的模型，以使其更接近目标分布。最终，我们得到预训练模型的另一个版本 $f(\theta_S)$，它仍需针对下游任务进行微调。图2总结了我们的自适应微调框架。

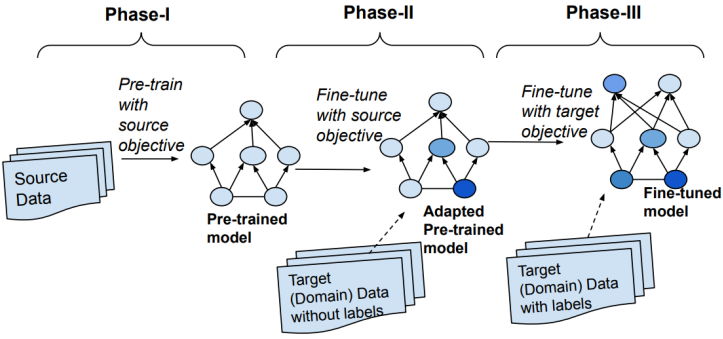


图 2：自适应微调框架

自适应微调能够持续有效地应对分布偏移。因此，在微调阶段开始前，可以使用目标数据集进一步训练模型。换言之，如图2所示，我们分为三个阶段。第一阶段包括预训练过程，用于通过源目标函数 θ_S 学习参数 J_S，此过程不需要任何标注数据。目标函数通常是掩码语言模型。事实上，文献中已经有许多为此目的训练的检查点。因此，我们不必从头开始预训练模型。这些预训练模型使用种类繁多的数据进行训练，其架构深度和参数数量也足够充分，正如第3.3节所述。

在第二阶段，继续训练预训练模型，再次使用相同的源目标函数 $J_S = J_A$，但采用目标数据集进行自适应。需要注意的是，还可以利用其他类型的辅助目标函数来改进自适应。在此阶段，我们不改变 Transformer模型架构，也不添加任何额外参数。

此外，在第三阶段，架构中会新增参数 θ_T。这些参数通常位于新层中，该层置于最后一层之上。随后，以端到端方式使用目标目标函数 J_T和目标标注数据训练新架构。在这一阶段，我们将学习率设得低于预训练和自适应微调阶段的学习率，以免大幅扰动主模型，因为到这一阶段，模型已经学习到了最重要的句法和语义信息。

3.5 实验设置

表 3 列出了不同 Transformer模型所使用的超参数。具体而言，我们对不同规模的 BERT检查点进行了实验。学习率（lr）和激活函数参数通过超参数调优实验确定。

表 3：Transformer模型超参数

检查点	层	隐藏单元	标题	参数总量	lr	激活
DistilBERT	6	768	12	66M	5e-4	gelu
BERT-base	12	768	12	110M	1e-4	gelu
BERT-large	24	1024	16	340M	1e-4	gelu
RoBERTa-base	12	768	12	125M	6e-4	gelu
RoBERTa-large	24	1024	16	355M	4e-4	gelu

在典型的监督训练流程中，无标签数据会被剔除，并在标注数据集上进行微调阶段。在 DOORS数据集中，需求文本超过 80,000 条，但其中大多数都没有标注。因此，我们使用整个数据集中的所有需求文本，通过 MLM 这一无监督学习目标进行自适应微调。输入中的词元会以 0.15 的概率随机遮蔽，以实现 MLM。此过程会产生一个新的适配预训练模型，其参数已更新，θ_S。

在其余训练过程中，我们遵循常见做法进行超参数选择。数据集分为三部分：训练集、验证集和测试集。学习率维持在约 1e-4，与 BERT 检查点等其他预训练超参数设置相同。模型最多训练 30 个训练轮次，并使用 $AdamW$ 优化器 [43]。训练结束时，加载训练过程中找到的最佳模型。随后，微调阶段作为最后一个阶段进行。

我们还遵循自适应阶段之后的微调阶段中的常见做法。具体而言，我们将优先级、严重程度和类型 这三个分类目标作为下游任务。对于每个任务，我们都对一个预训练模型进行了微调，该模型是在第二阶段获得的适配后预训练模型，

最多训练 3 轮。我们发现，在这一阶段将学习率设得非常低至关重要，这样才不会损害适配后的语言模型。

4 数值研究

本节将介绍详细数值研究的结果。首先，我们展示超参数调优实验中具有代表性的结果。然后，我们呈现与不同基线方法和 Transformer 及其自适应微调变体进行比较分析的结果。接下来，我们根据类别特定的结果，考察表现最佳模型的性能。最后，我们讨论模型预测，并详细分析误分类的潜在原因。

4.1 超参数调优结果

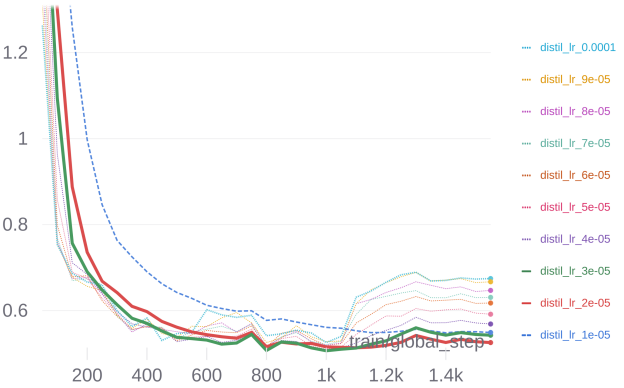
我们针对表 3 中列出的各项模型参数进行了超参数调优。图 3a 展示了对选定的 Transformer 模型（DistilBERT）进行微调时，不同学习率下类型预测这一下游任务的微调损失。评估的学习率范围为 1e-05 至 1e-04。我们发现，对于 1e-05 这类较低的学习率，尽管训练最终会收敛到理想水平，但可能需要很长时间，例如本例中最多需要 1500 步。学习率过高时，模型训练起初进展很快，但始终无法收敛并稳定下来，从而导致发散。当学习率选在 2e-05 至 3e-05 之间时（图中以粗直线表示），我们可以获得令人满意的结果，微调也能实现收敛。

图 3b展示了 DistilBERT 检查点进行自适应微调时的训练损失和验证损失。我们可以看到，经过 15 个训练轮次后，验证损失的下降几乎停止，模型趋于收敛。我们还观察到，训练过程中不存在方差和偏差问题。

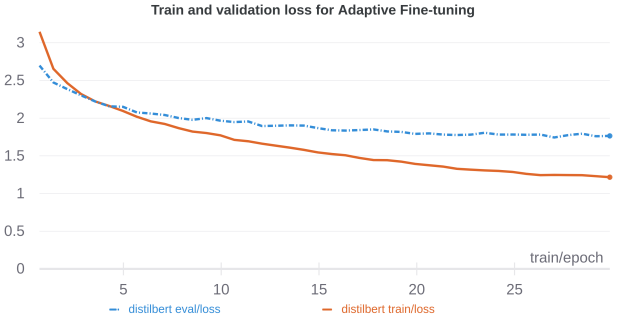
4.2 比较性能分析结果

我们根据准确率（ACC）、F1值和加权F1值（w-F1）报告模型性能，如表 4 所示。我们采用了三种基线模型，即多数类分类器、词嵌入（WE）和 SBERT。表格其余部分列出了 Transformer 模型的性能。为便于比较，我们按照表中所示，将两阶段（预训练和微调）学习与三阶段自适应学习的结果前后排列。三阶段过程用“+适配”表示。

这些结果表明，无论是否经过自适应，Transformer模型的表现都优于多数类分类器基线模型。在优先级 和类型 任务中，它们的表现也优于词嵌入池化和 SBERT 基线方法。这可归因于 Transformer 的泛化能力以及自适应带来的益处。然而，我们在严重程度 类别中观察到了相互矛盾的结果。



(a) 微调损失（x轴：训练步数）



(b) 自适应微调损失

图 3：所选预训练模型 DistilBERT 在超参数调优期间观测到的损失值示例（y轴：损失值）。

另一个观察结果是，自适应在所有模型设置下都带来了改进。我们没有发现任何负迁移或性能损失。此外，我们通过详细分析，考察了这些差异中哪些具有统计显著性，因为有些改进幅度很小。我们还注意到，所有适配模型的表现都相近，如图 4所示。

我们报告了自适应带来的改进，如表 5 所示。我们列出了自适应相对于其原始对应项和多数类分类器基线方法所取得的改进。 p值显示差异是否具有统计显著性。为此，我们采用 5×2交叉验证F检验，该检验能够判断两个分类器的性能是否存在显著差异 [44]。此方法对待比较的两个分类器进行 5 次 2 折交叉验证。因此，在相同条件下、使用相同折次得到的两个模型的值可以进行比较。如果 p值小于 α （0.05），则拒绝原假设，这意味着两个模型存在显著差异。

性能改进表明，适配模型始终能够带来积极贡献。也就是说，整体性能

表4：分类模型准确率对比（*：自适应微调版本）。

准确率 / F1值 /加权F1值	优先级	严重程度	Type
多数类分类器	68.39 / 20.31 / 55.56	86.11 / 15.42 / 76.69	30.16 / 6.62 / 13.98
特征提取基线			
词嵌入（Glove ＋ FastText）	71.52 / 31.82 / 66.18	86.60 / **27.76** / 81.74	75.92 / 73.81 / 75.10
SBERT	70.85 / 32.79 / 67.88	86.62 / **25.59** / 82.52	77.22 / 74.78 / 76.33
Transformer模型			
DistilBERT	72.09 / 33.20 / 65.34	86.43 / 15.76 / 79.84	78.80 / 73.88 / 76.72
＋ adapted	***72.54*** */ 33.21 / 65.45*	***86.61*** */ 16.9 / 80.89*	***81.15*** */ 77.57 /* ***79.24***
BERT-base	71.89 / 32.51 / 65.0	86.49 / 16.3 / 80.48	77.24 / 70.87 / 74.62
＋ adapted	**72.63 /33.8 /65.87**	86.56/ **22.3** / **81.64**	80.06 / 75.6 / 77.76
BERT-large	68.39 / 20.31 / 55.56	86.12 / 15.42 / 79.68	79.48 / 75.91 / 78.13
＋adapted	71.94 / 30.64 / 65.29	86.11 / 15.4 / 79.70	79.52 / 76.06 / 78.19
RoBERTa-base	71.33 / 31.04 / 62.11	86.32 / 15.11 / 79.29	80.12 / 77.11 / 78.29
＋adapted	71.78 / 32.12 / 65.12	86.21 / 29.43 / 81.49	80.45 / **78.71** / 79.15
RoBERTa-large	68.81 / 30.12 / 56.17	86.03 / 15.71 / 79.98	79.59 / 75.94 / 78.17
＋adapted	72.02 / 32.77 / 65.11	86.23 / 15.67 / 80.40	79.94 / 76.16 / 78.18

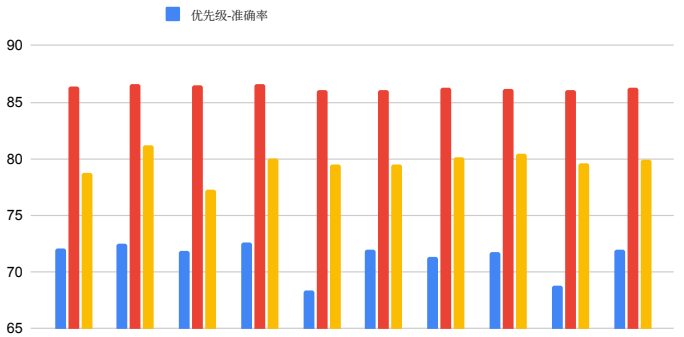


图 4：基于准确率对适配模型进行性能比较

适配模型很有前景。然而，正如表中所示，某些情况下的正向差异并未发现具有统计显著性，因为相应的 p值不小于 0.05。

我们可以清楚地看到，优先级和类型任务中，需求与目标类别之间存在很强的模式。因此，我们观察到这些任务取得了更好的改进。然而，在这些任务中，严重程度类别是最难准确分类的，这主要是因为我们无法观察到需求文本与目标之间存在紧密关系。在我们设计的许多模型和各种不同的实验设置中，

表5：与原始 Transformer 模型变体和多数类分类器基线相比，自适应带来的改进及其 p 值（报告为“+改进（p值）”）

	原始对应项			多数类分类器		
优先级	Acc.	F1	w-F1	Acc.	F1	w-F1
BERT+适配	+0.73(0.15)	+1.29(0.047)	+0.86(0.53)	+3.56(0.002)	+13.37(5.2e-6)	+9.44(1.1e-5)
DistilBERT+适配	+0.45(0.04)	+0.012(0.67)	+0.11(0.68)	+3.4(1.5e-7)	+12.78(2.2e-8)	+9.02(2.8e-6)
严重程度						
Bert+适配	+0.07(0.48)	+5.96（0.0015）	+1.16(0.0052)	+0.14(0.42)	+6.83(0.0015)	1.5(0.008)
DistilBERT+适配	+0.17(0.1)	+1.44(0.0029)	+1.04(0.08)	+0.183(0.065)	+1.45(0.0024)	+0.74(0.007)
Type						
BERT+适配	+2.82(0.0041)	+4.72(0.0043)	+3.14(0.0097)	+49.48(2.8e-6)	+68.91(3.5e-10)	63.44(2.8e-8)
DistilBERT+适配	+2.35（ 0.0023）	+3.69(0.0005)	+2.52(0.0033)	+50.57(2.5e-9)	+70.88(3.5e-10)	+64.92(2.8 e-8)

都没有观察到该类别存在显著模式。因此，Transformer模型无法取得相较于基线方法的任何显著改进，在自适应微调过程中也没有看到改进。基于特征提取的模型（词嵌入和 SBERT）表现更好，尤其是在该任务的 F1 分数方面。

4.3 详细性能评估结果

混淆矩阵以及精确率和召回率等其他指标，有助于深入了解预测错误。我们通过 DistilBERT 模型对此进行考察，并选择它作为具有代表性的 Transformer 模型。需要注意的是，我们在其他模型中也观察到了类似的表现，因此这里只是选用 DistilBERT 作为代表模型。表 6展示了 DistilBERT 自适应微调版本的详细性能值。根据这些结果，我们发现，优先级分类出错的主要原因之一是模型对“未分配”等高频标签过于敏感。因此，“中”和“低”等低频标签的表现受到类别分布不平衡的影响。

严重性分类任务表现出与优先级类别类似的情况。这同样可归因于该标签存在严重的数据不平衡：“普通”类别的占比明显高于其他类别。我们发现，“主要”类别的预测效果可能相当差，许多“主要”严重程度的需求被预测为“普通”严重程度。另一方面，类型标签的类别分布相对较为均衡。我们观察到，微调在这项任务上表现相当不错。只有一个类别标签“其他”的表现明显逊于其他类别。

4.4 模型预测讨论

接下来，我们将考察在 DOORS 数据集的三个分类任务中，模型未能正确预测实际类别的示例需求

表 6：针对优先级、严重程度和类型预测任务，经自适应微调的 Distil-BERT 模型的详细性能结果（混淆矩阵中的对角线以粗体显示）。

(a) 按类别划分的优先级性能值

	精确率	召回率	F1 分数	支持度
High	0.582	0.507	0.542	682
Low	1.000	0.022	0.044	132
中	0.466	0.191	0.271	497
未分配	0.785	0.926	0.849	2,837

(b) 优先级混淆矩阵

		预测			
		High	**Low**	**中**	**未分配**
Ground Truth	**High**	**346**	0	37	299
	Low	13	**3**	12	104
	中	86	0	**95**	316
	未分配	149	0	60	**2,628**

(c) 按类别划分的严重程度性能值

	精确率	召回率	F1 分数	支持度
阻塞项	0.800	0.150	0.250	26
严重	1.000	0.066	0.125	60
主要	0.390	0.195	0.262	291
轻微	1.000	0.134	0.237	67
普通	0.890	0.977	0.933	3,572
未决定	0.695	0.431	0.532	132

(d) 严重性混淆矩阵

		预测					
		阻塞项	**严重**	**主要**	**轻微**	**普通**	**未决定**
Ground Truth	**阻塞项**	**4**	0	7	0	14	1
	严重	0	**4**	16	0	39	1
	主要	1	0	**57**	0	232	1
	轻微	0	0	2	**9**	56	0
	普通	0	0	59	0	**3,491**	22
	未决定	0	0	2	0	73	**57**

(e) 按类别输入性能值

	精确率	召回率	F1 分数	支持度
增强	0.763	0.823	0.792	1,251
JUnit	0.922	0.902	0.912	132
维护	0.825	0.895	0.858	562
其他	0.589	0.354	0.442	466
计划项	0.749	0.809	0.778	199
用户故事	0.916	0.932	0.924	1,131
测试任务	0.920	0.936	0.928	407

(f) 类型混淆矩阵

		预测						
		增强	**JUnit**	**维护**	**其他**	**计划项**	**用户故事**	**测试任务**
Ground Truth	**增强**	**1,029**	1	70	80	21	48	2
	JUnit	1	**119**	1	2	1	6	2
	维护	44	0	**503**	10	1	4	0
	其他	212	3	33	**165**	12	17	24
	计划项	18	0	0	7	**161**	12	1
	用户故事	41	1	2	11	18	**1,054**	4
	测试任务	4	5	1	5	1	10	**381**

（见表7）。下面，我们总结有关这些误分类的一般观察结果。

- 总的来说，模型预测受到类别分布和输入长度的显著影响。在优先级、严重程度和类型分类任务中，大多数测试实例分别被分类为“未分配”、“普通”和“增强”类别。

- 在类型分类任务中，“用户故事”类别通常分配给提及某个特定软件需求的实例，而“增强”类别通常分配给提及对现有需求进行改进的已有任务。某些词语（如“改进”“确保”和“报告”）可能在多个类别中重叠，从而导致误分类。

- 严重性标签通常表示某个缺陷或任务对软件产品可能造成的影响程度。然而，严重性标记的级别可能带有主观性。因此，带有“严重”或“主要”标签的需求文本可能会被分类为“普通”，因为模型无法区分该需求是否属于系统中的重要功能和用例。

•优先级分类任务可以被视为根据业务需求以及解决特定缺陷的紧迫性，将需求/缺陷排序为“高”“低”和“中”。与严重性标签类似，

这些优先级值也可能具有很强的主观性。此外，由于“未分配”类别占绝大多数，且可能包含来自其他三个类别的实例，因此，优先级分类可以被视为比其他分类任务更具挑战性。

表7 列举了错分类案例的具体示例。在优先级分类中， R_1 被归类为“未分配”（表示无法确定适当的优先级），然而，需求文本表明这是一项高优先级任务，因为重命名命令无法刷新预期功能。同样， R_2 也被归类为“未分配”，尽管其优先级较低。在严重性分类中， R_3 被预测为“普通”，但从需求文本可以明显看出，该标签应为“严重”，因为用户无法在 Linux 和 Mac 工作站上访问图形编辑器插件，这可能导致 Mac 和 Linux 用户的应用程序完全停止运行。 R_4 也表明这是一项重要任务，因为用户无法在适当的时间锁定变更集交付，因此该任务需要紧急处理，这解释了其“主要”严重程度。然而，模型预测其严重程度为“普通”。在类型分类中， R_5 任务讨论了提升某个基线查询性能的改进，这很容易被识别为现有系统中的维护任务。“改进”之类的词可能使模型产生混淆，导致其将 R_5 归类为“用户故事”。 R_6 中出现“评估”一词，明确表明它是“测试任务”，而“Web 客户端”一词也表明涉及用户。因此，在这种情况下，模型无法区分“用户故事”和“测试任务”类别。

表7：错误分类实例示例

分类 **task**	**需求文本**	**实际类别** **类别**	**预测** **类别**
优先级	R_1：类型系统重命名命令不会刷新 TRS 信息流中受影响的形状。	High	未分配
	R_2：审核邮件可以发送给没有 RRC 项目访问权限的用户。	Low	未分配
严重程度	R_3：支持浏览器图形编辑器插件 Linux 和 Mac工作站	严重	普通
	R_4：变更集交付未能适当地锁定 time	主要	普通
Type	R_5：提升大型基线比较查询的性能 查询	维护 Item	用户故事
	R_6：评估现有 Web 客户端的测试用例	测试任务	用户故事

5 结论

现代语言模型在处理知识迁移时可能仍显不足，因为目标领域的分布可能与源领域的分布显著不同。尽管众所周知，与其他深度学习架构相比，Transformer 更能应对分布外问题，但在这方面仍有改进空间。本文表明，自适应微调有助于软件需求规格说明领域的多种文本分类任务。在三阶段训练流程中，我们在分布更接近目标数据的语料上微调预训练模型，从而有助于缩小源分布和目标分布之间的差异。采用此方法时，只需具备来自目标分布的无标签数据，因为模型是通过预训练目标进行微调的。我们将领域自适应微调用于软件需求规格说明文档分类的三项任务（优先级、严重程度和类型分类），并提升了模型在实际分布偏移情况下的性能。我们发现，严重程度分类是最具挑战性的任务，因为在这一类别中，需求文本与任务之间不存在显著模式。我们将模型与强基线方法进行比较，结果表明，增加自适应阶段可显著提升模型性能。

我们的工作可以从多个方向进行扩展。首先，由于缺乏公开可用的数据，我们仅使用单一数据源进行了实验。可以通过使用多种数据集，进一步研究自适应微调在软件需求规格说明分类任务中的有效性。其次，我们注意到，本研究考察的三项分类任务都存在数据不平衡问题。因此，可以结合自适应微调采用数据增强策略，以提升分类性能。最后，基线模型均在通用语料库上进行预训练，而使用针对软件工程语料库训练的Transformer模型，可能会在我们的分类任务中表现更好。

声明与陈述

作者声明不存在潜在的利益冲突。

数据可用性声明

DOORS数据集为专有数据，未公开提供。

参考文献

[1] D. Kici、G. Malik、M. Cevik、D. Parikh、A. Ba¸sar，《基于 BERT 的迁移学习方法：软件需求规格说明的文本分类》，载于：第34届加拿大人工智能会议论文集，2021年。

[2]S. Ruder, M. E. Peters, S. Swayamdipta, T. Wolf，自然语言处理中的迁移学习，载于：2019年北美计算语言学协会分会会议论文集：教程，2019年，第15–18页。[3] J. Dick, E. Hull, K. Jackson，DOORS：一种管理需求的工具，载于：《需求工程》，Springer，2017年，第187–206页。[4] D. Hendrycks, X. Liu, E. Wallace, A. Dziedzic, R. Krishnan, D. Song，预训练变换器提升分布外鲁棒性，arXiv 预印本 arXiv:2004.06100（2020年）。[5] H. Hussain，使用文本分类自动检测软件需求规格说明书文档中的歧义，博士论文，康考迪亚大学，2007年。[6] Y. Zhang, D. Hou，从论坛讨论中提取有问题的 API 特征，载于：2013年第21届国际程序理解会议（ICPC），IEEE，2013年，第142–151页。[7] D. Hou, L. Mo，API 讨论的内容分类，载于：2013年 IEEE 国际软件维护会议，IEEE，2013年，第60–69页。[8] M. R. Asadabadi, M. Saberi, O. Zwikael, E. Chang，模糊需求：一种在大型项目中识别并澄清歧义的半自动化方法，《计算机与工业工程》149（2020年）106828。[9]P. Singh, D. Singh, A. Sharma，一种基于规则的系统，用于从需求规格说明中自动分类非功能需求，载于：2016年计算、通信与信息学进展国际会议（ICACCI），IEEE，2016年，第620–626页。[10] R. Navarro-Almanza, R. Juarez-Ramirez, G. Licea，利用深度学习支持软件工程：软件需求分类案例，载于：2017年第5届软件工程研究与创新国际会议（CONISOFT），IEEE，2017年，第116–120页。[11] E. Onyeka, V. Anu, A. S. Varde，在软件需求规格说明书大数据中识别隐式需求，载于：2019年 IEEE 国际大数据会议，IEEE，2019年，第6169–6171页。[12] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova，BERT：用于语言理解的深度双向变换器预训练，arXiv 预印本 arXiv:1810.04805（2018年）。[13] T. Hey, J. Keim, A. Koziolek, W. F. Tichy，Norbert：用于需求分类的迁移学习，载于：2020年第28届 IEEE 国际需求工程会议（RE），IEEE，2020年，第169–179页。[14] A. Sainani, P. R. Anish, V. Joshi, S. Ghaisas，从软件工程合同中提取并分类需求，载于：2020年第28届 IEEE 国际需求工程会议（RE），IEEE，2020年，第147–157页。[15] A. Ramponi, B. Plank，自然语言处理中的神经网络无监督领域自适应综述，arXiv 预印本 arXiv:2006.00632（2020年）。

[16]S. Gururangan, A. Marasovic, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, N. A. Smith，别停止预训练：让语言模型适应领域和任务，CoRR abs/2004.10964（2020年）。[17] S. Diao, R. Xu, H. Su, Y. Jiang, Y. Song, T. Zhang，利用 n 元语法表示改造预训练语言模型，以实现低资源领域自适应，载于：第59届计算语言学协会年会和第11届自然语言处理国际联合会议论文集（第1卷：长篇论文），2021年，第3336–3349页。[18] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, J. Kang，BioBERT：用于生物医学文本挖掘的预训练生物医学语言表示模型，CoRR abs/1901.08746（2019年）。[19] I. Beltagy, K. Lo, A. Cohan，SciBERT：面向科学文本的预训练语言模型，arXiv 预印本 arXiv:1903.10676（2019年）。[20] T. Caselli, V. Basile, J. Mitrovic, M. Granitzer，HateBERT：针对英语辱骂性语言检测重新训练 BERT，CoRR abs/2010.12472（2020年）。[21]E. Alsentzer, J. Murphy, W. Boag, W.-H. Weng, D. Jindi, T. Nau-mann, M. McDermott，公开可用的临床 BERT 嵌入，载于：第二届临床自然语言处理研讨会论文集，计算语言学协会，明尼阿波利斯，明尼苏达州，美国，2019年，第72–78页。网址：https://aclanthology.org/W19-1909。doi：10.18653/v1/W19-1909。[22]A. Louis，NetBERT：面向计算机网络的预训练语言表示模型，2020年。[23] S. Peng, K. Yuan, L. Gao, Z. Tang, MathBERT：用于数学公式理解的预训练模型，CoRR abs/2105.00377（2021年）。[24] D. Guo, S. Ren, S. Lu, Z. Feng, D. Tang, S. Liu, L. Zhou, N. Duan, A. Svyatkovskiy, S. Fu, M. Tufano, S. K. Deng, C. B. Clement, D. Drain, N. Sundaresan, J. Yin, D. Jiang, M. Zhou，GraphCodeBERT：利用数据流预训练代码表示，CoRR abs/2009.08366（2020年）。[25]K. Clark, M. Luong, Q. V. Le, C. D. Manning，ELECTRA：将文本编码器预训练为判别器而非生成器，CoRR abs/2003.10555（2020年）。[26] A. Wettig, T. Gao, Z. Zhong, D. Chen，在掩码语言建模中是否应将 15% 的词元设为掩码？，2022年。doi：10.48550/ARXIV.2202.08005。[27] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov，利用子词信息丰富词向量，《计算语言学协会汇刊》5（2017年）135–146。[28]J. Pennington, R. Socher, C. D. Manning，Glove：词表示全局向量，载于：2014年自然语言处理实证方法会议（EMNLP）论文集，2014年，第1532–1543页。[29] T. Mikolov, K. Chen, G. Corrado, J. Dean，在向量空间中高效估计词表示，arXiv 预印本 arXiv:1301.3781（2013年）。

[30]J. Wieting, M. Bansal, K. Gimpel, K. Livescu，迈向通用释义句子嵌入，arXiv 预印本 arXiv:1511.08198（2015）。[31] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Con- stant, M. Guajardo-C´ espedes, S. Yuan, C. Tar 等，通用句子编码器，arXiv 预印本 arXiv:1803.11175（2018）。[32] Q. Le, T. Mikolov，句子和文档的分布式表示，载于：国际机器学习会议，PMLR，2014年，第1188–1196页。[33] N. Reimers, I. Gurevych，Sentence-BERT：使用孪生 BERT 网络的句子嵌入，arXiv 预印本 arXiv:1908.10084（2019）。[34] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Tor- ralba, S. Fidler, Skip-thought 向量，载于：神经信息处理系统进展，2015年，第3294–3302页。[35] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer，深度上下文化词表示，arXiv 预印本 arXiv:1802.05365（2018）。[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin，Attention Is All You Need，载于：神经信息处理系统进展，2017年，第5998–6008页。[37] J. Pfeiffer, A. R¨uckl´e, C. Poth, A. Kamath, I. Vuli´c, S. Ruder, K. Cho, I. Gurevych，AdapterHub：用于调整 Transformer 的框架，arXiv 预印本 arXiv:2007.07779（2020）。[38] S.-A. Rebuffi, H. Bilen, A. Vedaldi，使用残差适配器学习多个视觉领域，arXiv 预印本 arXiv:1705.08045（2017）。[39] V. Sanh, L. Debut, J. Chaumond, T. Wolf，DistilBERT：BERT 的精简版本，更小、更快、更便宜、更轻量，arXiv 预印本 arXiv:1910.01108（2019）。[40] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, RoBERTa：一种稳健优化的 BERT 预训练方法，arXiv 预印本 arXiv:1907.11692（2019）。[41] A. Torralba, A. A. Efros，对数据集偏差的无偏审视，载于：CVPR 2011，IEEE，2011年，第1521–1528页。[42] J. Qui˜nonero-Candela, M. Sugiyama, N. D. Lawrence, A. Schwaighofer, 机器学习中的数据集偏移，MIT 出版社，2009年。[43] I. Loshchilov, F. Hutter，解耦权重衰减正则化，arXiv 预印本 arXiv:1711.05101（2017）。[44] E. Alpaydm，用于比较监督分类学习算法的组合 5×2 cv f 检验，Neural Computation 11（1999）1885–1892。