《自动化软件工程》期刊稿件编号。（将由编辑插入）

软件工程跨平台设置下深度学习情感检测工具的实证研究

Gias Uddin、Md Abdullah Al Alamin 和 Ajoy Das

arXiv:2301.06661v1 [cs.SE] 17 Jan 2023

摘要 软件工程（SE）中的情感检测已展现出支持多种开发活动的潜力。然而，鉴于软件工程平台（如代码审查、开发者论坛）多种多样，软件工程领域特定情感检测工具在跨平台设置下的性能可能会受到影响，即在基于某个平台（如代码审查）创建的数据集上训练，然后在从另一个平台（如 Jira 问题）整理的数据集上测试。近期研究发现，基于深度学习（DL）的软件工程领域特定情感检测工具，其性能优于基于浅层机器学习（ML）或基于规则的工具。然而，深度学习工具在跨平台设置下的表现尚不明确。本文研究软件工程领域特定的深度学习情感检测器在跨平台设置下是否比基于浅层机器学习或基于规则的情感检测器更有效。我们在三个数据集上研究了三种深度学习工具（SentiMoji、BERT4SentiSE、RNN4SentiSE），并将其与三个基线进行比较：两种浅层学习工具（Senti4SD、SentiCR）和一种基于规则的工具（SentiStrengthSE）。这些数据集此前曾被 Novielli 等人用于研究基线的跨平台性能。研究结果显示：（1）在大多数情况下，面向软件工程的深度学习情感检测工具 BERT4SentiSE 在跨平台设置下优于其他监督式工具，但在大多数情况下又不及基于规则的工具 SentiStrengthSE。（2）在三个数据集的平台内设置下，BERT4SentiSE 的表现大幅优于 SentiStrengthSE；而在六种跨平台设置中的四种设置下，其表现仅略逊于 SentiStrengthSE。这一发现让我们对进一步提升 BERT4SentiSE 这类预训练 Transformer 模型在跨平台设置下的性能抱有希望。（3）两种表现最佳的深度学习工具（BERT4SentiSE 和 SentiMoji）在三个数据集上的性能下降程度各不相同。我们发现，这种不一致主要源于“人工标注人员在标注时的主观性”。进一步调查表明，六个数据集在文本内容方面并无太大差异。因此，要提升所研究的监督式工具在跨平台设置下的性能，可能需要修复数据集（例如，处理极性标注中的不一致性）。

关键词 情感分析、深度学习、自然语言处理、跨平台数据集

Gias Uddin、Md Abdullah Al Alamin 和 Ajoy Das，DISA 实验室，卡尔加里大学，电子邮件：{gias.uddin,mdabdullahal.alamin,ajoy.das}@ucalgary.ca

1 引言

与自然语言处理（NLP）类似，软件工程（SE）中的文本内容大致可分为两类：事实和观点 [28, 63, 57]。事实是客观表述（例如“我使用这个工具”），而观点是主观表述（例如“我喜欢这个工具”）。观点可以有多种形式，而 SE 研究主要关注情感检测，即将一个句子/帖子归为三种极性标签之一：正向、负向或中性（即不存在正向/负向极性）。

软件工程领域的研究发现，观点是软件工程中许多活动的关键决定因素，例如分析开发者的倦怠情况 [34]，、协助开发者进行 API 选择 [59, 61]，以及确定在线论坛中共享知识的质量 [65, 64, 60, 57]。因此，准确检测软件工程制品中的情感极性至关重要，这样才能收集正确的信息以支持分析。

由于无法正确理解软件领域特定术语，现成工具对软件工程领域情感的检测可能不够准确 [25, 27]。因此，研究人员基于多种技术开发了若干软件工程领域特定情感检测工具（下文简称“SD”），包括*rule-based*（例如 SentistrengthSE [22]，、Opiner [59]，、POME [26]）、 *supervised shallowlearning based*（例如 Senti4SD [8]，、SentiCR [1]）以及 *supervised deep learning based*（例如 RNN4SentiSE [7]，、BERT4SentiSE [6]，、SentiMoji [13]）。

研究发现，在若干软件工程情感数据集中，软件工程专用情感检测工具的表现优于现成的情感检测工具 [13, 2]。每个数据集都是针对特定平台构建的（例如，Jira 代码评审、Stack Overflow 讨论等）。软件工程平台的多样性仍使得将在一个软件工程平台（例如，Jira 代码评审数据集）上训练的软件工程专用情感检测工具应用于另一平台（例如，Stack Overflow 开发者讨论）颇具挑战。最近，Novielli 等人 [38, 39] 研究了跨平台设置下的若干有监督浅层学习软件工程专用情感检测工具，即在一个软件工程情感数据集上训练，然后在另一个软件工程情感数据集上测试。他们发现，有监督情感检测工具在跨平台设置下的性能会下降。然而，尚不清楚使用 BERT4SentiSE [6]，、SentiMoji [13]，和 RNN4SentiSE [7]等软件工程专用深度学习情感检测工具时，是否也会出现类似结果。

深入了解面向软件工程的深度学习情感检测工具在跨平台设置中的表现，以及深度学习工具是否优于其他工具及其优越之处，具有以下必要性：

1. **性能优势的一致性**。Novielli 等人 [38] 报告称，面向软件工程的浅层机器学习情感检测工具在跨平台设置下性能会下降。多项研究发现，在平台内设置下（即在单个数据集上进行训练和测试时），面向软件工程的深度学习情感检测工具优于浅层学习情感检测工具 [6, 13]。当新的软件工程平台没有可用的标注数据集时，针对各种软件工程数据集训练情感检测工具可能很困难，有时甚至不可能。因此，我们有必要了解面向软件工程的深度学习情感检测工具在跨平台设置下是否能比浅层学习情感检测工具表现更好；如果确实如此，那么在跨平台设置下，深度学习工具仍可能优于浅层学习情感检测工具，尤其是在这种情况下可能无法重新训练工具。

因此，合理的预期是，深度学习工具在跨平台设置下也会优于浅层学习情感检测工具。任何偏离这一预期的情况，都可能表明底层模型或数据集存在问题，例如模型在平台内设置下过拟合，而这又可能导致其在跨平台设置下性能下降。

2. **模型的预训练设置**。根据设计，我们研究的所有软件工程深度学习情感检测工具都基于预训练 Transformer 模型的理念构建。这意味着，每种工具都已在大型领域无关文本数据集语料库上进行预训练，以学习人类对话中通用的底层句法和语境。例如，BERT4SentiSE 基于先进的语言预训练 Transformer 模型 BERT 构建 [15]。随后，这些工具通过在预训练领域无关层之上添加额外层，基于软件工程专用数据集进行训练。相比之下，浅层学习情感检测工具未经预训练；也就是说，这些工具仅通过在软件工程专用数据集上训练来学习底层语境。这意味着，由于预训练和训练的方式，深度学习情感检测工具直观上比浅层学习情感检测工具掌握的知识更多。深度学习情感检测工具的这种预训练与训练双重设置，使我们认为，在跨平台设置下，深度学习情感检测工具的表现可能优于浅层学习情感检测工具。这是因为此类预训练模型的优势在于，它们已经在大量非软件工程、但跨领域的数据集上进行训练，从而学习特定语言的嵌入。这些见解有助于工具在训练数据很少的相对新领域中实现更精准的分类。这也表明，此类模型在跨平台设置下可能更为实用。

本文报告了一项实证研究的结果。我们开展这项研究，旨在评估三种面向软件工程的深度学习（DL）情感检测工具（BERT4SentiSE [6], SentiMoji [13], RNN4SentiSE [7]）在三个数据集上的有效性：GitHub [38], Jira [42], 和 Stack Overflow [8]。Novielli 等人 [38, 39]也使用这三个数据集，研究了面向软件工程的浅层学习情感检测工具在跨平台设置下的性能。

对于每种基于深度学习的情感检测工具，我们检查了六种跨平台设置，即针对三个基准数据集各有两种设置（也就是说，在其中一个数据集上训练，在另外两个数据集上测试）。我们的研究得出了若干发现：（1）在跨平台设置中，BERT4SentiSE 的表现最佳，其次是 SentiMoji。在每种设置中，与平台内设置相比，我们都观察到这些工具的表现有所下降。例如，在平台内设置中，BERT4SentiSE 在 GitHub 数据集上的宏平均 F1 分数为 0.92（也就是说，它也在 GitHub 数据集上训练），但在跨平台设置中，当工具在 Stack Overflow（SO）数据集上训练时，其在 GitHub 数据集上的表现降至 0.88；当工具在 Jira 数据集上训练时，则降至 0.65。平台内设置与跨平台设置之间的表现下降幅度差异，以 RNN4SentiSE 最大（在 SO 上训练后，再在 Jira 上测试时下降 31%）。这一发现与 Novielli 等人 [38],的研究结果一致，他们也观察到，面向软件工程的浅层学习情感检测工具在跨平台设置中的表现有所下降。这一观察促使我们根据以下研究问题，将深度学习工具与非深度学习工具的表现进行比较。（2）我们运行了表现最佳的两种浅层学习情感检测工具（Senti4SD 和 SentiCR）以及基于规则的工具 Sen-

tistrengthSE Novielli 等人 [38]。我们发现，在跨平台设置中，浅层与深度学习情感检测工具之间的比较结果显示，BERT4SentiSE 在六种设置中的五种表现最佳。然而，在跨平台设置中将 BERT4SentiSE 与基于规则的 SentistrengthSE 进行比较时，我们发现 BERT4SentiSE 仅在六种设置中的两种优于 SentistrengthSE。（3）我们从三个数据集中选取了 400 个误分类案例，其中 BERT4SentiSE 和 SentiMoji 均判断错误。我们总共发现了七种误分类类别。最常见的原因是“标注中的主观性”，即数据集中的人工标注不一致。这意味着，对于两个相似的句子，人工编码员给出的极性标签存在不一致（例如，在多个数据集中，一些编码员将“Thank you”标为“正向”，而另一些编码员则标为“中性”）。这类不一致会使深度学习工具产生混淆。（4）我们发现，两种表现最佳的深度学习情感检测工具（BERT4SentiSE 和 SentiMoji）之间的一致性，高于其他工具彼此之间的一致性。事实上，当我们将这两种深度学习工具结合起来，构建一个多数投票分类器（也就是说，综合 BERT4SentiSE 和 SentiMoji 给出的极性标签来为句子标注）时，它们仅在六种情形中的两种优于该设置下单独表现最佳的工具。因此，在跨平台设置中，仅通过对工具的结果进行多数投票并不足以解决问题，还需要进一步改进这些工具，例如通过修复数据标注中的不一致来改进数据集。

复现实验包：https://tinyurl.com/yf8nwbv6

2 研究工具与数据集

本节介绍本文研究的六种软件工程专用情感检测工具（第2.1节）和三个软件工程情感数据集（第2.2节）。

2.1 研究工具

本研究聚焦于近期推出的三种软件工程专用深度学习情感检测工具：1. RNN4SentiSE [7], 2、SentiMoji [13], 和 3. BERT4SentiSE [6]。近期，Zhang等人 [70] 指出，先进的预训练 Transformer 模型（预训练模型），例如 BERT，其表现优于浅层学习的软件工程专用情感检测工具。BERT4SentiSE [6]与 Zhang等人的研究同期推出。 [70]因此，我们没有使用 Zhang等人的任何预训练模型。 [70]在这三种工具中，BERT4SentiSE [6] 是软件工程领域最新的情感检测工具，且研究发现其表现优于所有其他软件工程专用情感检测工具。

我们将这三种工具的性能与 Novielli 等人 [38] 在跨平台研究中表现最佳的两种基于浅层机器学习的软件工程专用情感检测工具进行比较：Senti4SD [8] 和 SentiCR [1]。此外，鉴于 Novielli 等人 [38], 发现基于规则的工具 SentiStrengthSE [22] 表现更优，我们也将其性能与基于深度学习的情感检测工具进行比较。因此，我们研究的工具可分为三类：**1. 基于深度学习：** *SentiMoji*、*BERT4SentiSE*、*RNN4SentiSE*；**2. 基于浅层机器学习：** *Senti4SD*、*SentiCR*；**3. 基于规则：** *Sen-tiStrengthSE*。下面我们将简要介绍每种工具。

1. **SentiMoji** [13] 是基于 DeepMoji [16], 定制的。DeepMoji 使用从 565 亿条推文中学习到的词嵌入，并采用两个双向长短期记忆层和一个注意力层 [13]。SentiMoji 通过利用表情符号的使用方式，学习文本的向量表示

以及 Twitter 和 GitHub 上的其他文本。SentiMoji 在 Lin 等人 [27]的数据集上优于浅层情感检测工具和基于规则的情感检测工具。SentiMoji 的超参数通过一亿条 GitHub 表情符号 [30]帖子 [13] 使用链式解冻方法进行微调 [16]。在最后阶段，DeepMoji 的各层保持不变，并使用训练数据集更新 softmax 层最后的 64 个维度。在本研究中，我们使用 SentiMoji 表现最佳的架构，在这些数据集上重新训练。SentiMoji 的预训练版本使用 Stack Overflow 帖子作为词嵌入。它还利用 Twitter 推文学习通用特征，并利用 GitHub 帖子学习软件工程领域特定情感行话。

2. **BERT4SentiSE** [6] 基于 Google 的 De-vlin 等人开发的 BERT 模型 2018 [15]。它通过考察文本中某个词前后的词来学习词语的上下文表示。该模型在与任务无关的语言数据上进行预训练。它采用迁移学习 [47]和基于注意力机制的架构。Biswas 等人 [6] 提出了一种基于 BERT 的预训练模型，迄今为止，该模型在软件工程数据集上的表现最佳。在本研究中，我们使用了原作者报告的最佳模型参数。我们采用了预训练的 BERT[1] 模型，该模型包含 12 层，编码器层维度为 768，总计包含 1.1 亿个参数。

3. **RNN4SEntiSE** [7] 是一种基于循环神经网络的模型，使用来自 Google 新闻数据的通用词嵌入。作者还报告称，来自 Stack Overflow 帖子的特定于软件领域的词嵌入并未显著提升性能。在本研究中，我们使用 Google 新闻词嵌入，并采用与作者在原始研究中相同的基于长短期记忆网络的循环神经网络配置。与原始研究类似，我们设置长短期记忆网络单元数为 30、丢弃率为 0.2、批次大小为 30、训练轮数为 100，这些设置取得了最佳结果。

4. **Senti4SD** [8] 最初使用 4K 条 Stack Overflow 帖子进行训练。它使用词袋（BoW）、情感词典和词嵌入作为特征。词嵌入是利用 Stack Overflow 数据转储创建的，以提供特定领域的信息。在本研究中，我们使用该工具的 Python 版本 [10]。我们还采用了原始研究中使用的最佳超参数，这些参数在我们的实验中取得了最佳结果。与原始研究一样，我们使用了完整的特征集，即词典特征、语义特征和关键词特征。

5. **SentiCR** [2] 使用梯度提升树（GBT）算法。它使用词袋作为特征，并利用 SMOTE [12] 通过过采样来处理类别不平衡。按照 Uddin 等人 [63], 的方法，我们训练 SentiCR 来检测三个极性类别。在本研究中，我们采用了原作者报告的最佳配置和超参数。我们使用 GBT 算法进行分类。

6. **SentiStrengthSE** [22] 是在 SentiStrength [56] 的基础上开发的，通过引入专用于 SE 的规则和情感词实现。每个负向词的分数范围为 -2 至 -5；正向词的分数范围为 +2 至 +5。极性分数为 *a priori*，即不取决于某个单元中所表达情感的语境性质。对于输入文本，它会同时输出正向和负向分数。输入文本的总体极性分数通过正向分数与负向分数的代数和计算得出。如果分数之和大于 0，则将文本标记为“正向”；如果分数之和小于 0，则标记为“负面”；否则标记为“中性”。该工具的规则由 Ortu 等人在 Jira 数据集上开发并测试。 [43]

[1] http://goo.gl/language/bert

2.2 研究的数据集

我们研究了三个数据集：Stack Overflow（SO） [8]，Jira [42]，和 GitHub [38]。这三个数据集此前也曾被 Novielli 等人用于 [38] 其跨平台研究中。表1 给出了这些数据集的汇总统计信息。

表1：所研究基准数据集的描述性统计

数据集	#单位数	+VE	VN	-VE
GitHub [38]	7,122	2,013 (28%)	3,022 (43%)	2,087 (29%)
Jira [42]	5,869	1,128 (19%)	3,955 (67%)	786 (14%)
Stack Overflow [8]	4,423	1,527 (35%)	1,694（38%）	1,202（27%）

1. **Stack Overflow（SO）数据集** [8] 包含 4,423 条 Stack Overflow 帖子，包括由十二名受训编码员手动标注的问题、评论和回答。每条帖子由三名评审员进行标注，并依据 Shaver 框架 [50]的指南，通过多数投票确定最终极性。该数据集的分布较为均衡：35% 为积极情绪，27% 为消极情绪，38% 为中性情绪，即不包含情绪。Cohen 的 [14] Kappa, k为 0.74，表明评审员间一致性 [66]较高。

2. **Jira 数据集** [42] 包含来自不同开源软件项目（例如 Spring、Apache）的 5,869 个句子和问题评论。软件开发人员最初使用六种情绪（即爱、喜悦、愤怒、恐惧、悲伤和惊讶）对该数据集进行了标注。为保持与其他数据集的一致性，我们遵循 Novielli 等人 [38] 的类似研究，将 *“love”* 和 *“joy”* 标记为积极情感，将 *“anger”*和 *“sadness”* 标记为消极情感。 *“surprise”* 句子被丢弃，因为根据语境，它们可能是正向的，也可能是负向的。最后，不包含情绪的内容被标记为中性。该数据集的分布不均衡：19% 为积极情绪，14% 为消极情绪，67% 为中性情绪。

3. **GitHub 数据集** [38] 包含 7,122 个拉取请求和提交消息。三位作者对该数据集进行了标注。评审员接受了训练，依据 Shaver 等人提出的情绪分类模型进行标注。 [51],该模型采用树状层次结构对情绪进行分类。层次结构的顶层包含六种基本情绪：爱、喜悦、惊讶、愤怒、悲伤和恐惧。Shaver 等人的论文 [51] 详细列出了各主要层级对应的关键词和情绪子类型，这些内容易于理解。在 GitHub 数据集中，编码员检测到句子/帖子中的底层情绪为爱或喜悦时，将其标记为正向极性；底层情绪为愤怒、悲伤或恐惧时，则标记为负向极性。未检测到任何情绪时，标记为中性。对于包含“惊讶”情绪的句子/帖子，编码员会根据底层语境，将其标记为两种情感极性标签之一，即正向或负向。该数据集的情感极性标签中，正向约占 28%，负向约占 29%，中性约占 43%。

3 实证研究

我们的实证研究主要关注：与平台内设置相比，深度学习软件工程专用情感检测工具在跨平台设置中的表现如何。平台内设置指在同一数据集中训练和测试情感检测工具，并将数据集划分为训练集和测试集。跨平台设置指在一个数据集（例如 GitHub）中训练工具，并在另一个数据集（例如 Jira/Stack Overflow）中测试工具。具体而言，我们回答五个研究问题（RQ）：

RQ1. 深度学习情感检测工具在平台内设置和跨平台设置下的表现是否同样出色？（第3.1节）**RQ2.** 在跨平台设置下，深度学习情感检测工具的表现能否优于非深度学习工具？（第3.2节）**RQ3.** 跨平台设置下，这些工具的错误类别有哪些？（第3.3节）**研究问题4.** 在跨平台设置下，深度学习工具与非深度学习工具的集成能否优于单个工具？（第3.4节）

3.1 RQ_1 软件工程情感检测工具在平台内设置和跨平台设置下的表现是否同样出色？

3.1.1 Motivation

软件工程领域的研究发现，跨领域情感检测工具在软件工程数据集上的准确度可能不足。这些发现促使研究人员开发软件工程专用情感检测工具。深度学习技术的最新进展推动了多种面向软件工程的深度学习情感检测工具的开发。最新的面向软件工程的深度学习情感检测工具包括 BERT4SentiSE、SentiMoji 和 RNN4SentSE。研究发现，BERT4SentiSE 和 SentiMoji 的表现优于其他软件工程专用情感检测工具。然而，这些实验是在平台内设置下进行的，也就是说，工具使用同一数据集进行训练和测试。软件工程领域的近期研究还发现，软件工程专用浅层学习情感检测工具（Senti4SD 和 SentiCR）在跨平台设置下的表现可能会下降，即在一个软件工程数据集上训练，并在另一个软件工程数据集上测试。因此，有必要了解这些工具在跨平台设置下是否表现相似，因为这将表明这些工具具有更广泛的适用性。

3.1.2 Approach

我们研究了三种最新的面向软件工程的深度学习情感检测工具的性能：BERT4SentiSE、RNN4SentiSE 和 SentiMoji。我们比较了每种工具在平台内设置和跨平台设置下的性能。对于每种工具，我们检查了六种跨平台设置：每个数据集对应两种（即在一个数据集上训练，并在另一个数据集上测试）。我们按如下方式计算每种工具在这些设置下的性能。

– 在平台内。 我们使用k折交叉验证 [54] 在给定数据集上训练和测试工具。在本研究中，我们采用10折交叉验证设置

其中，数据集被划分为十折，每一折都会在某个阶段用作测试集。首先，我们使用Scikit StratifiedKFold [2] 技术将每个数据集划分为10折，以确保每一折中的极性类别占比与原始数据集相同。我们针对每个数据集，用十轮迭代训练和测试每个有监督情感检测工具。在第一轮迭代中，我们将数据集的第1折用作测试集，其余9折用作训练集。在第二轮迭代中，我们将第2折用作测试集，其余9折用于训练监督式工具。这个过程持续十轮，直到所有十折都用作测试集。随后，我们合并工具对每个测试折的预测结果，并计算该工具在该数据集上的总体表现。

– 跨平台。我们在一个数据集（例如 SO）上训练每种工具，并在另外两个数据集（例如 GitHub 和 Jira）上进行测试。我们报告每种工具在各测试数据集上的表现。这意味着，对于给定的待测数据集，我们会检查该数据集中所有句子/分析单元的正确和错误案例。因此，与平台内设置类似，我们会在跨平台设置中检查整个数据集，以分析其表现。在平台内设置中，我们采用 10 折交叉验证，进行十次迭代。对于跨平台设置，我们在一个训练数据集上运行一次工具（以创建训练好的模型），然后在另外两个测试数据集上各运行一次该模型。

我们使用信息检索领域的三种标准指标报告表现：精确率（P）、召回率（R）和 F1 分数（$F1$） [53]。 $Precision$ 表示在所有预测为正向的样本中，被正确预测为正向的样本数量。$Recall$ 表示在所有实际类别为正向的样本中，被正确识别为正向的样本数量。 $F1$ $measure$ 是精确率（P）和召回率（R）的调和平均值。

$$P = \frac{\sum_{i=1}^{l} TP_i}{\sum_{i=1}^{l} TP_i + FP_i}, \; R = \frac{\sum_{i=1}^{l} TP_i}{\sum_{i=1}^{l} TP_i + FN_i}, \; F1 = 2 * \frac{P * R}{P + R} \tag{1}$$

TP = 真正例数量， FP = 假正例数量， FN = 假负例数量。

在我们的多类别情感问题中，每个类别都会计算 TP、 FP和 FN。例如，我们有三个可能的情感类别（正向、中性和负向）。对于某个特定类别（例如中性情感）， TP 当工具预测为“中性”时发生。对于同一类别， FN 当工具预测为其他任何类别（例如正向或负向）时发生；而 TN 当工具预测为“中性”，但原始标签并非“中性”时发生。遵循最先进水平的做法，我们使用表2 中的混淆矩阵和公式 [53] (1)进行评估。我们通过微型和宏平均值，以及各极性类别，报告总体性能。宏平均会先计算每个类别的指标，再取平均，因此对每个类别一视同仁。另一方面，微型平均会汇总所有类别的贡献。微型平均会受到多数类性能的影响 [49] ，因为大多数样例都属于该类别。按照标准做法，我们使用宏平均 F1 分数报告表现最佳的工具 [40, 33]。

[2] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html

表 2：多分类（三类）情感检测的混淆矩阵

		预测类别		
		正向（P）	**负向（N）**	**中性（O）**
实际类别	**正向（精确率）**	TP_P	假正例$_N$，假负例$_P$	假正例$_O$，假负例$_P$
	负向（N）	假正例$_P$，假负例$_N$	TP_N	假阳性$_O$，假阴性$_N$
	中性（O）	假阳性$_P$，假阴性$_O$	假阳性$_N$，假阴性$_O$	TP_O

表3：软件工程特定情感分析工具在跨平台设置中的性能（SentiSE = SentistrengthSE）。

设置	极性	基于深度学习的软件开发工具									基于浅层机器学习						基于规则		
		SEntiMoji			BERT4SEntiSE			RNN4SentiSE			Senti4SD			SentiCR			SentiSE		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
测试数据集：GitHub																			
测试：GitHub 训练：GitHub（在平台内）	-NEG	.89	.85	.87	.89	.91	.90	.68	.69	.68	.90	.89	.90	.87	.69	.77	.81	.71	.76
	中性	.87	.92	.89	.92	.90	.91	.73	.66	.69	.89	.92	.91	.77	.91	.84	.74	.86	.79
	+POS	.94	.92	.93	.93	.94	.94	.64	.72	.68	.94	.91	.93	.89	.84	.86	.85	.76	.80
	微型	.90	.90	.90	**.92**	**.92**	**.92**	.68	.68	.68	.91	.91	.91	.83	.83	.83	.79	.79	.79
	宏	.90	.89	.90	**.91**	**.92**	**.92**	.68	.69	.68	**.91**	.91	.91	.84	.81	.82	.80	.78	.78
测试：GitHub 训练：Stack Overflow（交叉-平台）	-NEG	.85	.62	.72	.83	.73	.78	.69	.49	.58	.79	.56	.66	.83	.37	.51	.81	.71	.76
	中性	.67	.91	.77	.71	.91	.80	.61	.84	.71	.71	.82	.76	.60	.94	.73	.74	.86	.79
	+POS	.88	.65	.75	.88	.63	.73	.78	.57	.66	.76	.81	.78	.86	.62	.72	.85	.76	.80
	微型	.75	.75	.75	.78	.78	.78	.66	.66	.66	.74	.74	.74	.68	.68	.68	**.79**	**.79**	**.79**
	宏	.80	.73	.75	**.81**	.76	.77	.69	.63	.65	.75	.73	.73	.76	.64	.66	.80	**.78**	**.78**
测试：GitHub 训练：Jira（跨平台）	-NEG	.91	.39	.55	.90	.43	.58	.75	.39	.52	.85	.52	.64	.88	.27	.41	.81	.71	.76
	中性	.56	.98	.72	.59	.97	.73	.54	.95	.69	.60	.96	.74	.52	.98	.68	.74	.86	.79
	+POS	.94	.45	.60	.91	.51	.65	.80	.30	.44	.90	.46	.61	.89	.38	.53	.85	.76	.80
	微型	.66	.66	.66	.68	.68	.68	.60	.60	.60	.69	.69	.69	.60	.60	.60	**.79**	**.79**	**.79**
	宏	**.80**	.61	.62	**.80**	.64	.65	.70	.55	.55	.78	.65	.67	.76	.54	.54	**.80**	**.78**	**.78**
测试数据集：Stack Overflow (SO)																			
测试：SO 训练：SO（同一平台内）	-NEG	.84	.83	.83	.86	.85	.85	.74	.73	.74	.80	.83	.82	.80	.72	.76	.76	.76	.76
	中性	.83	.84	.84	.86	.85	.85	.75	.77	.76	.84	.79	.81	.79	.82	.81	.73	.79	.75
	+POS	.93	.93	.93	.93	.94	.93	.85	.85	.85	.90	.93	.91	.88	.91	.89	.91	.82	.86
	微型	.87	.87	.87	**.88**	**.88**	**.88**	.78	.78	.78	.85	.85	.85	.82	.82	.82	.79	.79	.79
	宏	.87	.87	.87	**.88**	**.88**	**.88**	.78	.78	.78	.85	.85	.85	.82	.82	.82	.80	.79	.79
测试：Stack Overflow 训练：GitHub（跨-平台）	-NEG	.82	.76	.79	.80	.86	.83	.66	.60	.63	.69	.76	.72	.64	.49	.55	.76	.76	.76
	中性	.75	.85	.80	.84	.80	.82	.61	.64	.63	.73	.76	.75	.65	.85	.74	.73	.79	.75
	+POS	.93	.85	.89	.93	.91	.92	.73	.74	.74	.92	.82	.87	.88	.74	.80	.91	.82	.86
	微型	.83	.83	.83	**.85**	**.85**	**.85**	.67	.67	.67	.78	.78	.78	.71	.71	.71	.79	.79	.79
	宏	.83	.82	.82	**.85**	**.86**	**.85**	.67	.66	.66	.78	.78	.78	.72	.69	.70	.80	.79	.79
测试：SO 训练：Jira（跨平台）	-NEG	.93	.31	.47	.88	.55	.68	.66	.31	.42	.75	.30	.43	.67	.22	.33	.76	.76	.76
	中性	.54	.98	.70	.63	.94	.76	.50	.91	.65	.54	.93	.69	.50	.95	.66	.73	.79	.75
	+POS	.98	.62	.76	.97	.74	.84	.83	.42	.56	.96	.64	.76	.96	.52	.67	.91	.82	.86
	微型	.67	.67	.67	.76	.76	.76	.58	.58	.58	.66	.66	.66	.60	.60	.60	**.79**	**.79**	**.79**
	宏	.82	.64	.64	**.83**	.74	.76	.66	.55	.54	.75	.62	.63	.71	.56	.55	.80	**.79**	**.79**
测试数据集：Jira																			
测试：Jira 训练：Jira（在平台内）	-NEG	.84	.68	.75	.82	.76	.79	.69	.46	.55	.71	.55	.62	.79	.66	.72	.68	.73	.71
	中性	.89	.91	.90	.90	.90	.90	.83	.89	.86	.84	.88	.86	.89	.89	.89	.92	.82	.87
	+POS	.79	.80	.79	.78	.80	.79	.73	.72	.72	.73	.71	.72	.75	.85	.80	.68	.92	.78
	微型	**.86**	**.86**	**.86**	**.86**	**.86**	**.86**	.80	.80	.80	.81	.81	.81	.85	.85	.85	.83	.83	.83
	宏	**.84**	.80	.81	.83	**.82**	**.83**	.75	.69	.71	.76	.72	.73	.81	.80	.80	.76	**.82**	.79
测试：Jira 训练：GitHub（跨平台）	-NEG	.63	.75	.69	.66	.74	.70	.51	.71	.60	.61	.64	.62	.69	.60	.64	.68	.73	.71
	中性	.93	.79	.85	.93	.80	.86	.92	.58	.71	.89	.78	.83	.90	.85	.87	.92	.82	.87
	+POS	.66	.91	.77	.65	.91	.76	.45	.92	.61	.62	.86	.72	.69	.89	.78	.68	.92	.78
	微型	.81	.81	.81	.81	.81	.81	.66	.66	.66	.78	.78	.78	.82	.82	.82	**.83**	**.83**	**.83**
	宏	.74	**.82**	.77	.75	**.82**	.77	.63	.74	.64	.71	.76	.73	**.76**	.78	.76	**.76**	**.82**	**.79**
测试：Jira 训练：SO（交叉-平台）	-NEG	.67	.68	.67	.72	.68	.70	.58	.61	.59	.50	.29	.37	.36	.06	.10	.68	.73	.71
	中性	.92	.81	.86	.92	.82	.87	.88	.77	.82	.82	.81	.81	.75	.94	.83	.92	.82	.87
	+POS	.67	.93	.78	.65	.93	.76	.62	.89	.73	.59	.80	.68	.71	.49	.58	.68	.92	.78
	微型	.82	.82	.82	.82	.82	.82	.77	.77	.77	.74	.74	.74	.73	.73	.73	**.83**	**.83**	**.83**
	宏	.75	.81	.77	**.76**	.81	.78	.69	.75	.71	.64	.63	.62	.61	.49	.50	**.76**	**.82**	**.79**

3.1.3 Results

在表3中，第三至第五列（SentiMoji、BERT4SentiSE、RNN4SentiSE）展示了三种用于软件工程的深度学习情感检测工具的性能。该表分为三个部分，各部分之间以一条贯穿整行的

表4：跨平台设置下软件工程特定监督式情感分析工具的性能下降（SO =Stack Overflow）

训练数据集	指标	基于深度学习									基于浅层机器学习					
		SEntiMoji			BERT4SEntiSE			RNN4SentiSE			Senti4SD			SentiCR		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
测试数据集：GitHub																
SO	▽ 微型	-16%	-16%	-16%	-15%	-15%	-15%	-3%	-3%	-3%	-18%	-18%	-18%	-17%	-17%	-17%
	▽ 宏	-11%	-19%	-17%	-12%	-18%	-16%	2%	-8%	-5%	-17%	-20%	-19%	-10%	-21%	-20%
Jira	▽ 微型	-27%	-27%	-27%	-26%	-26%	-26%	-12%	-12%	-12%	-24%	-24%	-24%	-27%	-27%	-27%
	▽ 宏	-11%	-32%	-31%	-13%	-31%	-29%	2%	-21%	-20%	-14%	-29%	-27%	-9%	-34%	-34%
测试数据集：Stack Overflow																
GitHub	▽ 微型	-5%	-5%	-5%	-3%	-3%	-3%	-15%	-15%	-15%	-8%	-8%	-8%	-13%	-13%	-13%
	▽ 宏	-4%	-5%	-5%	-3%	-3%	-3%	-15%	-15%	-15%	-7%	-8%	-8%	-12%	-15%	-15%
Jira	▽ 微型	-23%	-23%	-23%	-13%	-13%	-13%	-26%	-26%	-26%	-22%	-22%	-22%	-27%	-27%	-27%
	▽ 宏	-6%	-27%	-26%	-6%	-16%	-14%	-15%	-30%	-31%	-11%	-27%	-26%	-14%	-31%	-33%
测试数据集：Jira																
GitHub	▽ 微型	-6%	-6%	-6%	-6%	-6%	-6%	-17%	-17%	-17%	-4%	-4%	-4%	-3%	-3%	-3%
	▽ 宏	-11%	3%	-5%	-10%	0%	-7%	-16%	7%	-10%	-7%	7%	-1%	-6%	-2%	-5%
SO	▽ 微型	-5%	-5%	-5%	-5%	-5%	-5%	-4%	-4%	-4%	-9%	-9%	-9%	-14%	-14%	-14%
	▽ 宏	-10%	1%	-5%	-8%	-1%	-6%	-7%	9%	1%	-16%	-12%	-15%	-25%	-38%	-37%

表5：在同平台设置中正确分类、但在跨平台设置中被误分类的句子的百分比（各数值均四舍五入至最接近的整数）

SD Tool	正确率 / 误分类	GitHub 测试		Stack Overflow 测试		Jira 测试	
		训练 Stack Overflow	训练 Jira	训练 GitHub	训练 Jira	训练 GitHub	训练 Stack Overflow
BERT4SentiSE	总体	19	30	8	18	11	10
	严重	3	1	0	0	1	1
	Mild	16	28	7	18	11	10
SentiMoji	总体	19	30	10	29	11	10
	严重	1	1	1	0	0	0
	Mild	18	29	9	29	11	10

横线。例如，在“测试数据集：GitHub”标题下，我们展示各工具在将 GitHub 用作测试数据集时，于平台内和跨平台设置下的表现。这意味着：(1) 首先，我们展示工具在 GitHub 数据集内的表现（即采用训练-测试集划分技术）；(2) 其次，我们展示各工具在使用另外两个数据集进行训练后，在 GitHub 数据集上的表现，即一次使用 Stack Overflow (SO) 训练，另一次使用 Jira 数据集训练。因此，第一种设置属于平台内设置，后两种设置属于跨平台设置。对于给定的设置（即平台内或跨平台），我们报告每种工具在三个极性类别（即正向、负向、中性）上的三项性能指标：精确率 (P)、召回率 (R) 和 F1度量 (F1)。对于每种设置，我们还使用两项指标报告各工具的总体表现：微平均和宏平均。下面讨论重要的观察结果。

每种用于软件工程的深度学习情感检测工具在跨平台设置下的表现都低于其平台内表现。所有深度学习情感检测工具从平台内设置转到跨平台设置时，表现均明显下降。不过，这种下降幅度在六种跨平台设置之间有所不同，在三种深度学习工具之间也有所不同。在表 4中，我们根据两项总体指标（宏平均值和微平均值）展示了每种工具从平台内设置转到跨平台设置时的性能下降百分比。从宏平均 F1 分数来看，RNN4SentiSE 的降幅最大，达到 31%；该工具在 Jira 上训练后，再在

SO 数据集上测试。这意味着，RNN4SentSE 在平台内设置下于 SO 上测试时，宏平均 F1 分数为 0.78；而在 Jira 上训练后，它在 SO 上的宏平均 F1 分数为 0.54。对于 SentiMoji，最大降幅出现在该工具于 Jira 上训练、再在 GitHub 上测试时（31%）。SentiMoji 在 GitHub 上训练并测试时（平台内），宏平均 F1 分数为 0.90；而在 Jira 上训练后再于 GitHub 上测试时，该分数为 0.62。对于 BERT4SentiSE，最大性能降幅出现在该工具于 Jira 上训练、再在 GitHub 上测试时（29%）。BERT4SentiSE 在 GitHub 上训练并测试时（平台内），宏平均 F1 分数为 0.92；而在 Jira 上训练后再于 GitHub 上测试时，该分数为 0.65。因此，我们发现，三种深度学习工具在 Jira 数据集上训练、再在另外两个数据集上测试时，性能降幅最大。

无论是在平台内设置还是跨平台设置中，BERT4SentiSE 都是三种软件工程深度学习情感检测工具中表现最佳的。在 GitHub、SO 和 Jira 这三个数据集的平台内设置中，BERT4SentiSE 的宏平均 F1 分数分别为 0.92、0.88 和 0.83。平台内设置中表现第二好的深度学习情感检测工具是 SentiMoji，其在 GitHub、SO 和 Jira 这三个数据集上的宏平均 F1 分数分别为 0.90、0.87 和 0.81。RNN4SentiSE 在这三个数据集上的宏平均 F1 分数分别为 0.68、0.78 和 0.71。与我们在平台内设置中观察到的 BERT4SentiSE 和 SentiMoji 之间性能差距较小的情况类似，在六种跨平台设置中，这两种工具的性能差距仍然很小。例如，在 SO 上训练并在 GitHub 上测试时，BERT4SentiSE 的宏平均 F1 分数为 0.77，而 SentiMoji 的宏平均 F1 分数为 0.75。相比之下，在相同设置下，RNN4SentiSE 的宏平均 F1 分数为 0.65。
BERT4SentiSE 和 SentiMoji 均基于预训练模型的原理开发，即这两种工具在大量非软件工程数据集（如 Twitter 和 Wikipedia）上进行训练，以学习通用的词嵌入和表示。RNN4SentiSE 的架构不支持这种预训练嵌入。预训练的 Transformer 模型（如 BERT）在其他自然语言处理任务中表现更优 [55]。因此，BERT4SentiSE 和 SentiMoji 的性能下降幅度小于另一种深度学习情感检测工具 RNN4SentiSE，这一点并不意外。

深度学习情感检测工具在跨平台设置中的性能下降，主要是因为它们将中性句子错误标注为非中性句子，或反之。在表 5中，我们展示了表现最好的两种深度学习情感检测工具（BERT4SentiSE 和 SentiMoji）如何将在平台内设置中正确分类的句子，在跨平台设置中误分类。例如，在平台内设置下，BERT4SentiSE 在 GitHub 数据集上正确分类的句子中，有 30% 的句子在使用 Jira 数据进行训练、然后在同一 GitHub 数据集上测试时被误分类。其中只有 1% 属于严重误分类，即正确标签为负向/正向时，却被错误预测为正向/负向。其余 29% 的误分类属于轻微错误，即正确标签为非中性/中性时，却被错误预测为中性/非中性。例如，BERT4SentiSE 在平台内设置中正确预测了“that’s because it’s old API. :((GitHub, oracle: negative)”；但在使用 Stack Overflow 进行训练时，在跨平台设置中却将其错误预测为中性。同样，SentiMoji 在平台内设置中正确预测了“sorry for the delay (Jira, oracle: negative)”；但在使用 Stack Overflow 进行训练时，在跨平台设置中却将其错误预测为中性。

上述观察表明，深度学习情感检测工具在跨平台设置中的表现不佳，主要是由于分类中的轻微错误，而不是严重错误。

RQ_1 软件工程的深度学习情感检测工具在平台内设置和跨平台设置中的表现是否同样出色? 我们研究的三个数据集中的每一个深度学习软件工程情感检测工具，其性能都出现了下降；其中，当工具在 Jira 数据集上训练、随后在另外两个数据集上测试时，性能下降最为明显。在这三个情感检测工具中，BERT4SentiSE 的表现最佳，SentiMoji 紧随其后。与平台内设置相比，跨平台设置中的误分类主要是由轻微错误造成的，即工具将中性误分类为非中性，反之亦然。

3.2 RQ_2 深度学习情感检测工具在跨平台设置中的表现是否优于非深度学习情感检测工具?

3.2.1 Motivation

RQ1 的研究结果表明，我们研究中的三个软件工程深度学习情感检测工具（BERT4SentiSE、SentiMoji、RNN4SentiSE）在跨平台设置中的性能均有所下降。这一发现与 Novielli 等人的研究结果一致， [38] 他们观察到，在跨平台设置中，浅层机器学习情感检测工具（Senti4SD、SentiCR）的性能也有所下降。鉴于此前的研究发现，在平台内设置中，深度学习情感检测工具的表现优于非深度学习情感检测工具，因此有必要了解：尽管深度学习情感检测工具在跨平台设置中的性能有所下降，它们是否仍能在跨平台设置中胜过非深度学习工具。

3.2.2 Approach

为了比较这三种深度学习情感检测工具与非深度学习情感检测工具的性能，我们选取了 Novielli 等人在跨平台设置下表现最佳的三种非深度学习情感检测工具：Senti4SD、SentiCR 和 SentiStrengthSE。在非深度学习工具中，Senti4SD 和 SentiCR 基于浅层机器学习，而 SentiStrengthSE 基于规则。我们按照 RQ 中评估三种深度学习工具的相同流程，计算两种浅层学习情感检测工具在我们的数据集上的性能$_1$。该方法在第3.1.2节中进行了描述，其中分为平台内和跨平台设置。SentiStrengthSE 基于规则，因此无需在任何数据集上对其进行训练。为此，我们只需在每个数据集上运行一次，并计算性能指标。我们使用第3.1 节中介绍的相同性能指标来报告性能。

3.2.3 Results

在表3中，我们展示了两种浅层学习情感检测工具（Senti4SD、SentiCR）和基于规则的工具 SentiStrengthSE 在最后三个

表6：工具与人工标注之间的一致性指标。

训练数据集	分类器	K	完美一致性	不一致	
				严重	Mild
测试数据集：GitHub					
GitHub（平台内）	SEntiMoji	0.84	90%	1%	9%
	BERT4SentiSE	0.87	92%	1%	7%
	RNN4SentiSE	0.52	68%	7%	25%
	Senti4SD	0.86	91%	1%	8%
	SentiCR	0.73	83%	2%	15%
	SentiStrengthSE	0.67	79%	2%	19%
Stack Overflow（跨平台）	SEntiMoji	0.61	75%	2%	23%
	BERT4SentiSE	0.65	78%	3%	19%
	RNN4SentiSE	0.47	66%	4%	30%
	Senti4SD	0.60	74%	4%	22%
	SentiCR	0.49	68%	2%	29%
	SentiStrengthSE	0.67	79%	2%	19%
Jira（跨平台）	SEntiMoji	0.44	66%	1%	33%
	BERT4SentiSE	0.48	68%	2%	30%
	RNN4SentiSE	0.35	60%	4%	36%
	Senti4SD	0.50	69%	2%	28%
	SentiCR	0.34	60%	1%	39%
	SentiStrengthSE	0.67	79%	2%	19%
测试数据集：Stack Overflow					
Stack Overflow（平台内）	SentiMoji	0.80	87%	1%	12%
	BERT4SentiSE	0.82	88%	1%	11%
	RNN4SentiSE	0.67	78%	3%	19%
	Senti4SD	0.77	85%	1%	14%
	SentiCR	0.73	82%	3%	15%
	SentiStrengthSE	0.68	79%	1%	20%
GitHub（跨平台）	SEntiMoji	0.73	83%	1%	16%
	BERT4SentiSE	0.78	85%	1%	14%
	RNN4SentiSE	0.49	67%	4%	29%
	Senti4SD	0.67	78%	2%	20%
	SentiCR	0.56	71%	5%	23%
	SentiStrengthSE	0.68	79%	1%	20%
Jira（跨平台）	SEntiMoji	0.48	67%	0%	33%
	BERT4SentiSE	0.63	76%	1%	23%
	RNN4SentiSE	0.34	58%	4%	38%
	Senti4SD	0.47	66%	1%	33%
	SentiCR	0.37	60%	2%	38%
	SentiStrengthSE	0.68	79%	1%	20%
测试数据集：Jira					
Jira（平台内）	SEntiMoji	0.71	86%	0%	14%
	BERT4SentiSE	0.72	86%	0%	13%
	RNN4SentiSE	0.57	80%	1%	20%
	Senti4SD	0.59	81%	0%	19%
	SentiCR	0.69	85%	0%	15%
	SentiStrengthSE	0.67	83%	0%	17%
GitHub（跨平台）	SEntiMoji	0.65	81%	1%	18%
	BERT4SentiSE	0.65	81%	1%	18%
	RNN4SentiSE	0.45	66%	2%	32%
	Senti4SD	0.58	78%	1%	21%
	SentiCR	0.65	82%	1%	17%
	SentiStrengthSE	0.67	83%	0%	17%
Stack Overflow（跨平台）	SEntiMoji	0.65	82%	1%	18%
	BERT4SentiSE	0.66	82%	1%	17%
	RNN4SentiSE	0.57	77%	1%	22%
	Senti4SD	0.46	74%	1%	25%
	SentiCR	0.34	73%	1%	26%
	SentiStrengthSE	0.67	83%	0%	17%

列。在表 4中，最后两列显示了基于浅层机器学习的工具在跨平台设置下相较于平台内设置的性能下降百分比。由于 SentiStrengthSE 无需重新训练，因此其在跨平台设置下不会出现性能下降，故未列入表4。

在本研究涉及的所有软件工程情感检测工具中，SentiStrengthSE 在六种跨平台设置中的五种设置下表现最佳，而 BERT4SentiSE 在一种跨平台设置下表现最佳。虽然在平台内设置下，BERT4SentiSE $considerably$ 的表现优于 SentiStrengthSE，但在六种跨平台设置中的五种设置下，SentiStrengthSE $marginally$ 的表现优于 BERT4SentiSE。当 BERT4SentiSE 在 SO 上训练、再在 GitHub 上测试时，SentiStrengthSE 的宏平均 F1 分数为 0.78，而 BERT4SentiSE 为 0.77（F1 分数差异为 $= 0.01$）。总体而言，在 SentiStrengthSE 表现优于 BERT4SentiSE 的五种跨平台设置中，两者的宏平均 F1 分数差异介于 0.01–0.02 之间；但当 BERT4SentiSE 在 Jira 上训练、在 GitHub 上测试时，差异为 0.13。BERT4SentiSE 表现优于 SentiStrengthSE 的唯一一种跨平台设置，是在 GitHub 上训练并在 SO 上测试，此时宏平均 F1 分数的差异为 0.06。因此，在跨平台设置下，SentiStrengthSE 与 BERT4SentiSE 的性能差异最多为 0.06，但 BERT4SentiSE 在 Jira 以外的数据集上训练时除外。因此，像 BERT4SentiSE 这样的监督式工具的性能，以及它在跨平台设置下能否胜过像 SentiStrengthSE 这样的基于规则的工具，可能取决于训练数据集的类型。

在软件工程领域的浅层学习和深度学习情感检测工具中，BERT4SentiSE 在六种跨平台设置中的五种表现最佳，而 Senti4SD 在一种跨平台设置中表现最佳。在 BERT4SentiSE 表现优于 Senti4SD 的五种跨平台设置中，两者的宏平均 F1 分数差异介于 0.04 至 0.16 之间。在一种跨平台设置中，Senti4SD 的表现优于 BERT4SentiSE：当工具使用 Jira 数据训练并在 GitHub 上测试时，其宏平均 F1 分数为 0.67，而 BERT4SentiSE 为 0.65，即差异为 $= 0.02$。BERT4SentiSE 在全部三种平台内设置中的表现也都优于 Senti4SD，且优于其他工具。研究结果证实，与文本分类的其他领域类似，在软件工程领域，BERT4SentiSE 这类预训练 Transformer 工具也优于其他有监督情感检测工具。BERT4SentiSE 的表现略逊于基于规则的工具 SentiStrengthSE，因此无论是在平台内设置还是跨平台设置中，它都可能是一个合理的选择。

在同平台和跨平台设置下，RNN4SentiSE 在有监督情感检测工具中出现的严重错误和轻微错误最多。在表 6 中，我们展示了本研究中的五种有监督情感检测工具在同平台和跨平台设置下与人工标注的一致性。我们还报告了基于规则的 SentiStrengthSE 与人工标注的一致性；但由于它无需重新训练，其跨平台一致性不会发生变化。我们使用两种指标计算一致性：Cohen κ [14] 和百分比一致率。我们将工具的预测类别与人工标注之间的不一致分为严重和轻微两类：当人工标注为正向的句子被预测为负向，反之亦然时，属于严重不一致；当人工标注为中性标签的句子被预测为非中性，反之亦然时，属于轻微不一致。在这些情感检测工具中，两种深度学习情感检测工具（BERT4SentiSE 和 SentiMoji）与人工标注之间的严重和轻微不一致最少。

在表6 中，我们展示了同平台和跨平台设置下的一致性与不一致情况。总体而言，对于任何给定的测试数据集，工具在跨平台设置下的严重和轻微不一致数量都比同平台设置下更多。不过，与 RQ1 中的观察类似，我们指出 BERT4SentiSE 和 SentiMoji 在跨平台设置下主要出现轻微错误；表6 中所有有监督情感检测工具（RNN4SentiSE 除外）与人工标注呈现出相似的模式。因此，在跨平台设置下进行分类时，有监督情感检测工具可以通过修复轻微错误获得更大收益。

表现最佳的两种深度学习情感检测工具（BERT4SentiSE 和 SentiMoji），在有监督情感检测工具中，与人工标注的一致性最高，严重不一致的比例最低。在表6中，我们报告了工具预测结果与人工标注的配对比较。在跨平台设置下，当使用 Stack Overflow 数据集训练并使用 GitHub 数据集测试时，BERT4SentiSE 和 SentiMoji 与人工标注的严重不一致比例最高（分别为 3% 和 2%），轻微不一致比例也最高（分别为 30% 和 33%）。我们发现，大多数严重不一致（61%）是由于标注中的主观性，或示例缺少恰当的语境。例如，GitHub 数据集中的一个示例（如“这么做真是很糟糕的做法 :)”），BERT4SentiSE 和 SentiMoji 都能正确理解该句子的底层语境，即“:)”这个表情符号并未使深度学习工具产生误判，二者都将该示例预测为负向情感。然而，对于这个具体示例，标准答案是“正向”，也就是说，人工标注者可能掌握了更多语境，促使其作出这一标注；或者这也可能是标注者之间存在主观性的结果（关于标注中的主观性，我们将在 RQ_4中进一步讨论）。

研究问题$_2$ 在跨平台设置中，深度学习情感检测工具的表现是否优于非深度学习情感检测工具? 在本研究涵盖的所有软件工程情感检测工具中，在我们研究的三个数据集所涉及的六种跨平台设置中，SentiStrengthSE 在其中五种设置下表现最佳，但仅略胜 BERT4SentiSE；在另一种跨平台设置下，BERT4SentiSE 表现最佳。在浅层学习和深度学习软件工程情感检测工具中，BERT4SentiSE 在六种跨平台设置中的五种设置下表现最佳，而 Senti4SD 在一种设置下表现最佳。

3.3 研究问题$_3$ 跨平台设置下，这些工具的错误类别有哪些?

3.3.1 Motivation

RQ_1 和 RQ_2 的研究发现表明，所有用于软件工程的有监督情感检测工具在跨平台设置下都会出现性能下降。对于 RQ_3，我们发现，两个数据集越相似，在跨平台设置下预期的性能下降就越小。然而，这并不能完全解释性能下降的原因，也无法解释为什么某个情感检测工具会错误地预测给定句子的情感，而另一个情感检测工具却能正确预测。了解这类句子的错误类型，可以进一步为未来改进这些工具在跨平台设置下的性能提供见解。

3.3.2 Approach

我们手动分析了跨平台设置下误分类句子的错误类别。我们分三个步骤进行分析。首先，从 4,188 个 BERT4SentiSE 和 SentiMoji 均分类错误的句子中随机抽取 400 个。该样本量在 95% 的置信水平和五个区间下具有统计显著性。随后，我们根据 Novielli 等人为每个句子手动标注错误类别。 [38],他们在研究浅层软件工程专用情感检测工具在跨平台设置下的有效性时提出了这些类别。本研究采用了 Novielli 等人使用的相同七种错误类别，但将其中一个类别的名称从“极性事实”改为“领域特定”，因为作者所认为的极性事实实际上是软件领域专用术语，这些术语具有某种内在的正面或负面极性（例如“bug”“问题”“已解决”）。最后两位作者参与了错误分析。他们讨论了将错误归入不同类别的指南。作者通过多轮联合讨论，确定样本中 400 个句子的错误类别。在联合讨论期间，两位作者共同标注每个句子，而不是分别标注。这样的安排确保两位作者针对每个句子进行共同讨论，并在标注过程中统一或澄清对错误代码的理解。此外，在标注过程中，如果需要全体作者共同澄清，两位作者还会咨询第一作者。由于采用了联合讨论的方式，作者无需记录彼此之间的一致或不一致程度。其次，我们将自己的错误类别与 Novielli 等人的类别进行比较。 [38], 他们分析了三个用于跨平台设置的数据集中的两种浅层学习情感检测工具（Senti4SD 和 SentiCR）的错误类别，其中两种工具均分类错误。第三，我们分析了 70 个随机抽取的句子，这些句子来自三个数据集，且基于规则的工具 SentiStrengthSE 在这些句子上的分类有误。步骤 2 和步骤 3 的分析旨在确定错误类别是否会因工具类型（即深度学习、浅层学习或基于规则）而异。

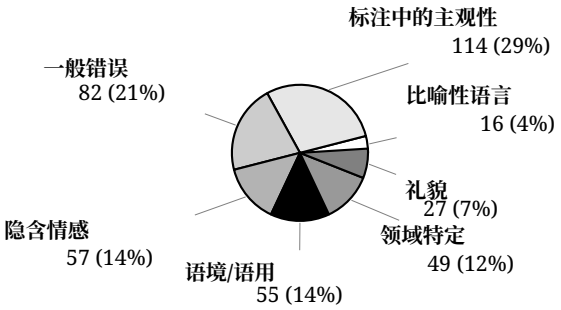


图1：我们的400个样本中，两种深度学习工具（BERT4SentiSE 和 SentiMoji）的错误类别分布

3.3.3 Results

在我们的样本中，BERT4SentiSE 和 SentiMoji 均判断错误的案例里，大多数（29%）错误源于“标注中的主观性”，即人工标注不一致。在图1中，我们用饼图展示了在六种跨平台设置下，BERT4SentiSE 和 SentiMoji 均判断错误的 400 个句子样本中观察到的七类错误所占的百分比分布。大多数误分类（29%）源于“标注中的主观性”。第二常见的类别是“一般错误”（21%）。这一分布与 Novielli 等人报告的分布略有不同 [38]：在其针对基于浅层机器学习的软件缺陷检测工具的研究中，“一般错误”最为常见（68%）。正如我们将在下文讨论的那样，一般错误是由于工具无法处理某些句子（例如，包含表情符号的句子）而导致的。我们研究中的一种深度学习工具 SentiMoji 专为更好地处理表情符号而设计。因此，在我们的样本中，与“一般错误”相关的误分类比例要低得多。下面我们将结合示例讨论各类错误。

标注中的主观性（29%）。误分类最大的成因是标注中的主观性。即使有针对数据集标注的标准指南，人工标注对人工标注者而言始终具有主观性 [38] *“+1 。例如，在同一个 Jira 数据集中，某条内容被标注为中性，而另一条则被标注为正向。这是因为保守型评审者有时会将带有轻微情绪的句子标注为中性 [38]。尤其是带有礼貌表达的句子，主观性相当普遍。例如，这句话被人工标注为中性，但两种工具（BERT4SentiSE 和 SentiMoji）都将其标注为正向，* *“Thanks you for quick response”*。

一般性错误（21%）。工具无法识别语言线索，可能导致极性标注出现误分类。如上所述，与 Novielli 等人相比，我们的样本中观察到此错误类别的情况要少得多 [38]。两种深度学习工具在此类别中的表现相对较好，因为二者都是预训练的，即使句子未被妥善处理，也有助于泛化句子的语境。此外，SEngMoji 基于 DeepMoji [16], ，擅长识别“:)”和“:smiley:”等表情符号。然而，我们仍在数据集中发现了误分类案例。例如，我们在 GitHub 数据集中发现了此类错误：*“Ok, this can be removed :-)”*。该错误发生在两种工具都使用 Jira 进行训练、并在 GitHub 上进行测试时。使用 SO 数据集进行训练时，两种工具都能提供正确的极性标签。我们发现，Jira 数据集中有类似这样的句子 *“Vadim you’re keeping us busy :).”* ，或这样的句子 *“Oh my stupid:)”*，又或这样的句子 *“Stupid user error :)”*。这三个句子被人工标注为非正向，但两种工具都将其标注为正向。因此，尽管我们将此类误分类归入一般性错误类别，但也可以将其另归为数据集标注不一致。

隐含情感（14%）。隐含情感的存在可能会使任何情感检测工具感到困惑，尤其是在分析单位是词数很少（即语境不多）的句子时。这两种深度学习情感检测工具最常将此类情况误分类为中性，因为句子中不含任何词汇线索。例如，以下两个句子经人工标注为负向，但两种情感检测工具都将它们标注为中性：1. *“This is not what I build a framework for.”* 2。 *“25hours gone.”*

语境/语用（14%）。句子缺乏恰当的语境或语用信息，尤其是短句，会让工具难以确定正确的极性标签。如果人工标注过程也参考了某个句子周围的句子，这同样可能造成问题。正如 Uddin 等人 [63] 所示，如果在 SO 中对某个句子的人工标注是通过查看该句子所在的整篇帖子完成的，那么缺乏语境可能会给基于 SO 的数据集带来问题。例如，以下句子被工具标注为中性，但在基准数据集中被标注为负向：1. *“huh ... i thought i didresolve this.”*（在 Jira 中）2. *“How about fixing hashcode too?”*（在 GitHub 中）。

领域特异性（12%）错误的产生，是因为句子中出现了情感检测工具在训练期间未曾见过的软件工程领域特定行话。有些软件工程领域特定行话本身就带有令人满意或不满意的情绪，例如“bug”具有负向极性，而“patch accepted”具有正向极性。这个句子被人工标注为负向，但工具将其预测为中性：*“My patch wouldn’t compile”*。另一个例子是 *“‘killed that process and restarted, and it worked!”*。这里，人工标注的极性为正向，因为在软件工程语境中，重启后问题得到修复，这是一个令人满意的结果。更糟的是，这些软件工程领域特定行话在不同数据集中标注不一致，例如，Jira 数据集中的 bug 相关句子总是被标注为正向或负向极性 [38, 40, 39]。然而，在另外两个数据集中，bug 相关讨论被标注为中性，除非其中明确表达了情绪。以下句子被人工标注为负向，但被两个工具都预测为中性：*“The visibility is missing, and the default value seems weird”*。

礼貌（7%）。类型错误有两个原因：（1）在不同数据集中，表达礼貌的句子所获标注不一致。例如，在 SO 和 GitHub 数据集中，礼貌被视为中性（例如，*“Thank you”*）[38]。另一方面，在 Jira 数据集中，致谢表达被标注为正向极性（例如，*“Thanks Ashish For committing the patch.”*）；（2）含有礼貌表达的句子最容易受到主观标注的影响，尤其是在跨平台数据集中（如上文主观标注类别中所述）。

比喻性语言（4%）。错误源于句子中存在讽刺或粗话。例如，在 GitHub 数据集中，以下句子被标注为正向，但在跨平台设置下，两个工具都将其预测为中性：*“There is a backdoor in this commit, i’ll give a cookie to the one who finds it first:P”*。以下句子在基准数据集中被标注为中性，但工具将它们预测为中性：1 2 *“I 。 used to be the latter but then I went to law school and got a degree in this shit.” “because hack shit methods must die.”*（GitHub），。，（Jira）。

两种深度学习工具的错误类别分布略有不同于跨平台设置下浅层情感检测工具所观察到的分布。图2 展示了两个基于浅层学习的情感检测工具（Senti4SD 和 SentiCR）都判断错误的 320 个句子中的错误类别。我们报告了 Novielli 等人的结果。[38]，也就是说，我们没有分析这 320 个句子。两个浅层学习情感检测工具中最常见的类别是“一般错误”，这是图1中第二常见的类别，即两个深度学习情感检测工具所观察到的错误类别。图1中两个深度学习情感检测工具最常见的错误类别是“标注中的主观性”，这是图2中两个浅层学习情感检测工具的第二常见类别。浅层学习与深度学习情感检测工具的错误类别分布之所以不同，原因有二：（1）一般性错误主要是由于工具无法分析有助于检测极性的非文本内容（例如表情符号）而产生的。

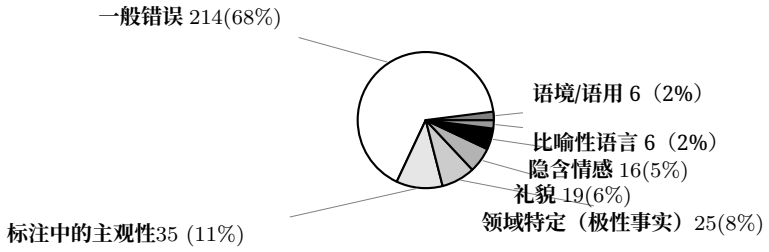


图 2：Novielli 等人报告的浅层情感分析工具（SentiCR 和 Senti4SD）的错误类别分布 [38]

两个深度学习情感检测工具（尤其是 SentiMoji）基于 Twitter 上的大量表情符号语料库进行预训练。因此，在跨平台设置下进行极性标注时，两个深度学习情感检测工具更能抵御此类一般性错误。（2）两个深度学习情感检测工具，尤其是 BERT4SentiSE，基于 BERT。BERT 由于采用了新颖的 Transformer 架构和注意力层，在从句子中推断语境方面表现更佳。因此，BERT4SentiSE 能够从训练数据中识别底层语境；而对于训练数据集与测试数据集中相似、却因人工标注的主观性而被赋予不同标签的句子，这会造成问题。因此，深度学习情感检测工具更容易受到这种不一致性的影响。

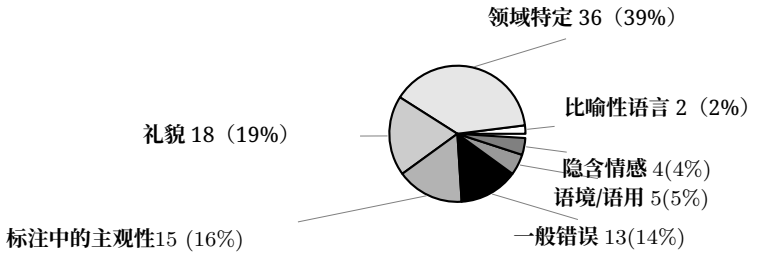


图 3：基于规则的工具（SentiStrengthSE）在我们的 93 个样本中的错误类别分布

对语境和底层领域特异性的理解不足，是基于规则的情感检测工具 SentiStrengthSE 出现最多错误的原因。在我们的三个数据集中，SentiStrengthSE 将 3,470 个句子误分类。我们从中随机抽取 93 个句子，对错误类别进行人工分析。该样本在 95% 的置信水平下具有统计显著性，置信区间为 10。图 3 展示了 SentiStrengthSE 误分类错误的概况。通过错误分析，我们发现，基于词典的工具 SentiStrengthSE 误分类的主要原因是无法理解领域特定语境（39%）。基于预训练模型和深度学习的情感检测工具在这一类别中的表现相当不错（错误仅占 12%）。以下是 SentiStrengthSE 因“领域特异性”而产生错误的一些示例

：“Confirm, this breaks compile. :((GitHub)”、“ROT13. It’s super efficient too! (Stack Overflow)”。标注指南不一致是第二大错误原因（19%）：SentiStrengthSE 总是将“Thanks”“Thank you, some name”等礼貌表达预测为正向极性；但在 Jira 数据集中，除非有明确的情绪表达，否则礼貌不会被标注为正向。第三类错误是主观标注（16%）。第四类错误是一般类别（14%），即无法理解语言线索（例如，“+1 Patch looks nice.”“:+1: for speed!”“ya +1 for clearer code comment :D”）。SentiStrengthSE 无法检测出这类语言线索。

当表现最佳的两款深度学习情感检测工具（BERT4SentiSE 和 SentiMoji）都与人工标注存在严重不一致时，大多数情况下，这是由于人工标注具有主观性。从表6 中我们可以看到，在跨平台设置下，使用 Stack Overflow 进行训练、使用 GitHub 数据集进行测试时，预训练的深度学习情感检测工具 SentiMoji 和 BERT4SentiSE 与人工标注的严重不一致率最高（分别为 2% 和 3%）。为了更好地了解这些问题，我们进一步进行了调查。我们在 GitHub 数据集中发现了 74 个样本，其中 BERT4SentiSE 和 SentiMoji 都与人工标注存在严重不一致；也就是说，如果某个句子的人工标注为正向，那么这两款情感检测工具都将其预测为负向，反之亦然。将这 74 个样本人工归入七种错误类别后，我们发现其中 45 个样本（即 61%）实际上属于主观标注。这意味着我们认同这些样本的人工标注，但我们认为工具的预测才是正确的。例如，“天哪，我真不擅长这个 :)”和“这样做确实是个坏习惯 :)”被标注为正向，但两款工具都将其预测为负向，而我们也认同这一判断。遵循标准标注指南即可修正这类错误。我们还发现，在 16 个案例（即 22%）中，两款工具都未能理解句子中的隐含情感。例如：“谢谢你，不过我这边的情况马上就要失控了。:(（GitHub，负向）”和“是啊，真蠢，我居然没检查其他提交……抱歉 : (（GitHub，正向）”。我们发现有 10 个案例（即 14%）是由于工具未能正确理解软件领域概念而导致错误。例如：“太好了，客户端又卡死了……”两款工具都将该样本预测为正向，而正确的预测应为负向。我们的分析表明，这些工具未能理解诸如“修复”“卡死”“难以使用库”等软件工程领域专用术语。

在两个深度学习情感检测工具（BERT4SentiSE、SentiMoji）均出错的误分类句子中，约有 45% 可由非深度学习情感检测工具纠正。图 10 展示了我们从图 1 所示样本中选取的 400 个句子中，两个深度学习情感检测工具（BERT4SentiSE、SentiMoji）均出错的句子是否能被本研究中的另外三个非深度学习情感检测工具纠正，以及如何纠正。两个深度学习情感检测工具造成的大多数一般性错误都可以由非深度学习情感检测工具纠正。我们发现，这是因为存在数据预处理错误，例如，深度学习情感检测工具内置的分词器模块在某些情况下无法正确处理表情符号。浅层学习情感检测工具 SentiCR 使用预定义的表情符号词典，可以弥补许多此类错误。总体而言，在所有七种错误类别中，深度学习情感检测工具都可以从非深度学习情感检测工具的正确分类中受益。

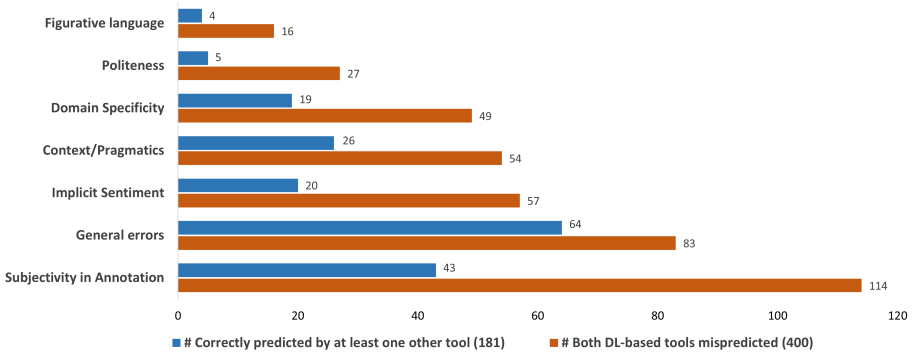


图 4：图中 BERT4SentiSE 和 SentiMoji 均误分类、但至少一种非深度学习情感检测工具能够纠正的案例 1

研究问题$_3$ 跨平台设置下这些工具的错误类别有哪些? 在跨平台设置下，BERT4SentiSE 和 SentiMoji 均出错的 400 个案例样本中，标注中的主观性是造成错误最多的原因。我们还观察到数据集间存在不一致案例（即相似句子在不同数据集中被赋予不同标签）。对于两个浅层学习情感检测工具（Senti4SD、SentiCR）均出错的句子，一般性错误最为常见。对于基于规则的工具 SentistrengthSE，缺乏对底层语境/领域的理解是造成错误最多的原因。

3.4 RQ$_4$ 在多大程度上，深度学习和非深度学习情感检测工具能够在跨平台设置下相互补充?

3.4.1 Motivation

我们从 RQ1 和 RQ2 得出的发现表明，有监督情感检测工具在跨平台设置下的性能会下降。我们在 RQ4 中对这些工具的误分类案例进行人工分析后发现，错误可分为七种类型，其中标注中的主观性、一般性错误和领域特异性是各工具中最常见的错误类别。情感检测工具采用不同的技术，并基于不同的平台进行设计。例如，SentiStrengthSE 主要通过分析 Jira 中的代码评审而设计，而 SentiMoji 使用从 Twitter 等论坛收集的表情符号。直观而言，鉴于这些工具的设计方式各不相同，它们可以相互补充。事实上，RQ4 中的图 10显示，非深度学习情感检测工具能够纠正两个表现最佳的深度学习情感检测工具（BERT4SentiSE 和 SentiMoji）共同误分类的许多案例，从而补充深度学习情感检测工具。因此，值得探究的是，能否通过让这些工具相互补充，降低情感检测工具在跨平台设置下的性能下降，例如，根据多个情感检测工具给出的极性标签，确定某个句子的最终极性判定。

Table 7a: Agreement between tools. Test in GitHub (G). Mji = SentiMoji Brt = BERT4SentiSE K = Cohen κ

Data	T1 ↔ T2	Agreement		Disagree	
		κ	P	S	M
G (W)	Mji vs Brt	0.86	91%	1%	8%
	Mji vs Rnn	0.52	68%	7%	25%
	Mji vs S4D	0.83	89%	1%	10%
	Mji vs SCr	0.76	85%	2%	13%
	Mji vs SSt	0.68	79%	2%	18%
	Brt vs Rnn	0.53	69%	7%	24%
	Brt vs S4D	0.85	90%	2%	8%
	Brt vs SCr	0.74	83%	2%	14%
	Brt vs SSt	0.68	80%	2%	18%
	Rnn vs S4D	0.52	69%	7%	25%
	Rnn vs SCr	0.43	63%	7%	31%
	Rnn vs SSt	0.44	63%	6%	31%
	S4D vs SCr	0.74	83%	2%	14%
	S4D vs SSt	0.67	79%	3%	18%
	SCr vs SSt	0.63	77%	3%	21%
S(C)	Mji vs Brt	0.72	83%	1%	15%
	Mji vs Rnn	0.58	76%	2%	22%
	Mji vs S4D	0.51	70%	3%	27%
	Mji vs SCr	0.40	68%	2%	30%
	Mji vs SSt	0.57	74%	2%	25%
	Brt vs Rnn	0.59	76%	2%	22%
	Brt vs S4D	0.52	70%	4%	26%
	Brt vs SCr	0.39	66%	3%	31%
	Brt vs SSt	0.61	76%	2%	22%
	Rnn vs S4D	0.43	65%	4%	31%
	Rnn vs SCr	0.30	62%	3%	35%
	Rnn vs SSt	0.48	68%	3%	29%
	S4D vs SCr	0.45	68%	2%	30%
	S4D vs SSt	0.55	72%	3%	25%
	SCr vs SSt	0.45	68%	2%	31%
J(C)	Mji vs Brt	0.78	90%	0%	10%
	Mji vs Rnn	0.54	81%	1%	19%
	Mji vs S4D	0.60	82%	1%	18%
	Mji vs SCr	0.65	87%	0%	13%
	Mji vs SSt	0.46	69%	0%	30%
	Brt vs Rnn	0.56	81%	1%	18%
	Brt vs S4D	0.60	81%	1%	18%
	Brt vs SCr	0.67	87%	0%	13%
	Brt vs SSt	0.51	71%	1%	28%
	Rnn vs S4D	0.44	74%	2%	24%
	Rnn vs SCr	0.45	79%	1%	20%
	Rnn vs SSt	0.36	63%	2%	34%
	S4D vs SCr	0.52	79%	1%	20%
	S4D vs SSt	0.48	70%	2%	29%
	SCr vs SSt	0.39	66%	0%	34%

Table 7b: Agreement between tools. Test in SO (S). S4D = Senti4SD, Scr = SentiCR, P = Percent Agreement

Data	T1 ↔ T2	Agreement		Disagree	
		κ	P	S	M
S (W)	Mji vs Brt	0.86	91%	1%	8%
	Mji vs Rnn	0.70	80%	3%	17%
	Mji vs S4D	0.81	88%	1%	12%
	Mji vs SCr	0.78	85%	2%	12%
	Mji vs SSt	0.72	81%	1%	17%
	Brt vs Rnn	0.71	81%	2%	17%
	Brt vs S4D	0.83	88%	1%	11%
	Brt vs SCr	0.78	85%	3%	12%
	Brt vs SSt	0.72	81%	1%	18%
	Rnn vs S4D	0.70	80%	3%	17%
	Rnn vs SCr	0.65	77%	4%	19%
	Rnn vs SSt	0.63	75%	3%	21%
	S4D vs SCr	0.78	85%	3%	12%
	S4D vs SSt	0.73	82%	1%	17%
	SCr vs SSt	0.70	81%	2%	17%
G (C)	Mji vs Brt	0.79	86%	1%	13%
	Mji vs Rnn	0.53	70%	4%	27%
	Mji vs S4D	0.70	80%	3%	17%
	Mji vs SCr	0.62	75%	5%	20%
	Mji vs SSt	0.67	79%	1%	20%
	Brt vs Rnn	0.53	69%	5%	26%
	Brt vs S4D	0.73	82%	2%	16%
	Brt vs SCr	0.60	74%	5%	21%
	Brt vs SSt	0.69	79%	1%	20%
	Rnn vs S4D	0.54	69%	4%	26%
	Rnn vs SCr	0.40	61%	6%	33%
	Rnn vs SSt	0.48	66%	4%	30%
	S4D vs SCr	0.58	73%	4%	23%
	S4D vs SSt	0.65	77%	1%	22%
	SCr vs SSt	0.56	71%	4%	25%
J (C)	Mji vs Brt	0.66	82%	0%	18%
	Mji vs Rnn	0.47	75%	1%	23%
	Mji vs S4D	0.60	81%	1%	19%
	Mji vs SCr	0.56	80%	1%	19%
	Mji vs SSt	0.49	68%	0%	31%
	Brt vs Rnn	0.43	70%	2%	28%
	Brt vs S4D	0.57	77%	1%	22%
	Brt vs SCr	0.52	75%	1%	25%
	Brt vs SSt	0.60	75%	0%	25%
	Rnn vs S4D	0.42	72%	2%	27%
	Rnn vs SCr	0.33	70%	1%	29%
	Rnn vs SSt	0.36	60%	3%	36%
	S4D vs SCr	0.51	77%	1%	22%
	S4D vs SSt	0.46	66%	1%	33%
	SCr vs SSt	0.40	63%	1%	36%

Table 7c: Agreement between tools. Test in Jira (J). SSt = SentistrengthSE, C, W = Cross, Within-Platform

Data	T1 ↔ T2	Agreement		Disagree	
		κ	P	S	M
J (W)	Mji vs Brt	0.85	93%	0%	7%
	Mji vs Rnn	0.68	86%	0%	14%
	Mji vs S4D	0.72	87%	0%	13%
	Mji vs SCr	0.81	91%	0%	9%
	Mji vs SSt	0.75	87%	0%	13%
	Brt vs Rnn	0.68	85%	1%	14%
	Brt vs S4D	0.72	87%	0%	13%
	Brt vs SCr	0.79	90%	0%	10%
	Brt vs SSt	0.75	87%	0%	13%
	Rnn vs S4D	0.61	83%	0%	17%
	Rnn vs SCr	0.70	86%	1%	14%
	Rnn vs SSt	0.63	81%	1%	18%
	S4D vs SCr	0.70	86%	0%	13%
	S4D vs SSt	0.67	83%	1%	16%
	SCr vs SSt	0.77	88%	1%	11%
G (C)	Mji vs Brt	0.81	89%	1%	10%
	Mji vs Rnn	0.56	72%	3%	25%
	Mji vs S4D	0.72	84%	1%	14%
	Mji vs SCr	0.75	86%	1%	13%
	Mji vs SSt	0.75	86%	1%	13%
	Brt vs Rnn	0.57	73%	3%	24%
	Brt vs S4D	0.73	85%	1%	14%
	Brt vs SCr	0.75	87%	1%	13%
	Brt vs SSt	0.75	86%	1%	13%
	Rnn vs S4D	0.52	70%	3%	27%
	Rnn vs SCr	0.50	69%	2%	28%
	Rnn vs SSt	0.54	71%	2%	27%
	S4D vs SCr	0.68	83%	1%	16%
	S4D vs SSt	0.70	83%	1%	16%
	SCr vs SSt	0.76	87%	1%	12%
S (C)	Mji vs Brt	0.80	89%	1%	10%
	Mji vs Rnn	0.70	84%	1%	15%
	Mji vs S4D	0.61	79%	1%	19%
	Mji vs SCr	0.35	70%	1%	29%
	Mji vs SSt	0.77	87%	1%	12%
	Brt vs Rnn	0.72	84%	1%	14%
	Brt vs S4D	0.60	79%	2%	19%
	Brt vs SCr	0.36	71%	1%	28%
	Brt vs SSt	0.77	87%	1%	12%
	Rnn vs S4D	0.56	77%	2%	22%
	Rnn vs SCr	0.30	67%	1%	32%
	Rnn vs SSt	0.70	83%	1%	16%
	S4D vs SCr	0.41	76%	1%	23%
	S4D vs SSt	0.59	78%	2%	20%
	SCr vs SSt	0.34	70%	1%	29%

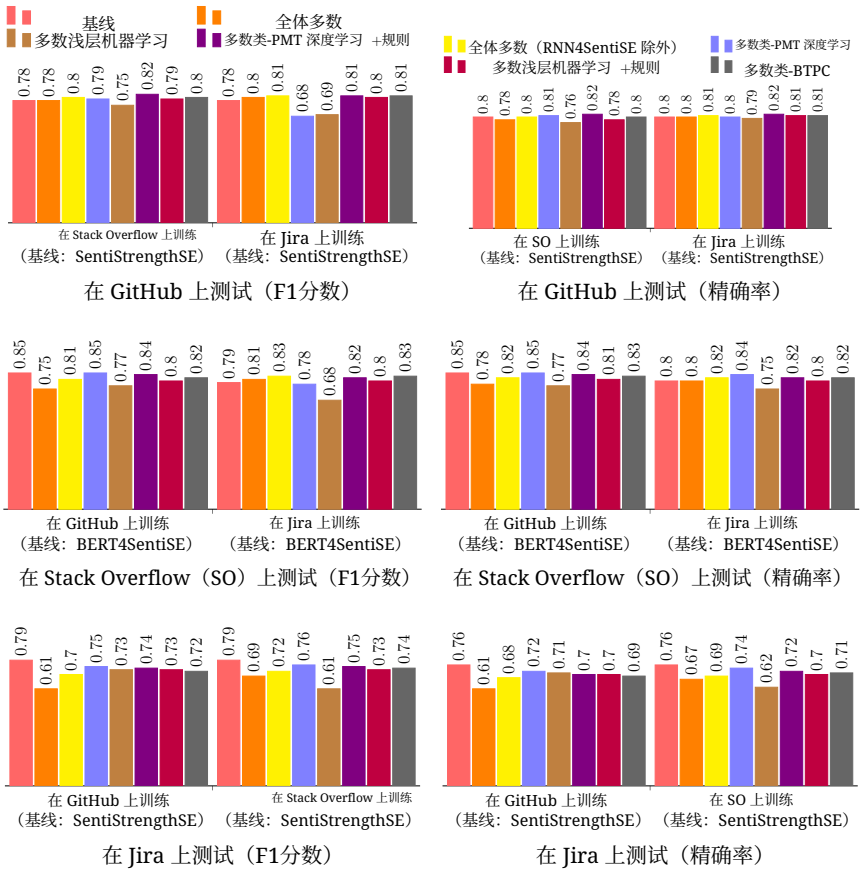


图 5：多数投票集成工具与情感检测工具的性能比较（All——除基线 DL-RNN4SEntiSE 外的所有情感检测工具；BTPC——PMT 中最佳的基于深度学习的情感检测工具（BERT4SentiSE）、基于浅层机器学习的情感检测工具（Senti4SD）和基于规则的情感检测工具（SentiStrengthSE））

3.4.2 Approach

我们采用基于多数投票的技术，研究工具的不同组合，以了解这些工具是否能够以及如何相互补充。我们的目的是探究此类组合能否降低工具在跨平台设置中的性能下降。这意味着，如果我们将三个工具（例如，全部三个深度学习情感检测工具）组合起来，首先要获取这三个工具对给定句子的极性标签，然后将出现次数最多的极性标签（即至少由 2 个工具给出的标签）赋予该句子。为了了解工具的不同组合，我们针对所研究的三个数据集，分别计算工具在同平台和跨平台设置下的一致性与不一致性。我们使用 Cohen κ和百分比一致率，计算给定设置和数据集下两个工具之间的一致性。我们使用两项指标计算不一致性：严重

和轻微。相关指标详见 RQ2（表6）。根据分析结果，我们在多数投票技术中研究了工具的七种不同组合：

–全体多数：基于 SentiMoji、BERT4SentiSE、RNN4SentiSE、Senti4SD、SentiCR 和 SentiStrengthSE 进行多数投票。– 全体多数，但不包括 RNN4SentiSE。– 多数类-PMT 深度学习：表现最好的两种深度学习情感检测工具（SentiMoji 和 BERT4SentiSE）。–多数浅层机器学习：表现最好的两种浅层学习情感检测工具（Senti4SD 和 SentiCR）。–多数类-PMT 深度学习 ＋ 规则：表现最好的两种深度学习情感检测工具加上基于规则的情感检测工具：（SentiMoji、BERT4SentiSE 和 SentiStrengthSE）。– 多数浅层机器学习 ＋ 规则：两种浅层学习情感检测工具加上基于规则的情感检测工具（Senti4SD、SentiCR 和 SentiStrengthSE）。– 多数类-BTPC：表现最好的深度学习、浅层学习和基于规则的情感检测工具：（BERT4SentiSE、Senti4SD 和 SentiStrengthSE）。

我们计算并报告了六种跨平台设置下每种组合的精确率、召回率和 F1 分数。

3.4.3 Results

在本研究的六种软件工程专用情感检测工具中，表现最好的两种深度学习情感检测工具（BERT4SentiSE、Senti- Moji）彼此之间的一致程度最高，不一致程度最低。在表7a、7b和7c中，我们报告了各分类器对之间的一致性成对比较。结果表明，SentiMoji 和 BERT4SentiSE 彼此之间的一致性最高。它们的完全一致百分比也最高，严重不一致百分比则最低。SentiMoji 和 BERT4SentiSE 都是预训练模型，已经具备语言理解能力。这些工具经过重新训练，用于特定任务。它们在正确预测极性方面的表现都很出色。相反，RNN4SentiSE 和 SentiCR 彼此之间的一致性最低。

基于多数投票的集成，结合表现最佳的两种深度学习情感检测工具（BERT4SentiSE ＋ SentiMoji），或结合表现最佳的两种深度学习情感检测工具（BERT4SentiSE ＋ SentiMoji） ＋在跨平台设置中的表现略优于独立情感检测工具。基于规则的工具（SentistrengthSE）也比独立情感检测工具表现稍好。图5 展示了六种跨平台设置中七种多数投票组合的表现。对于每种设置，我们报告七种组合（下文称为集成器）以及表现最佳的独立情感检测工具（在图5中标记为基线）的表现。对于每种设置，我们报告两项性能指标：精确率和宏平均 F1 分数。例如，图5最上方一行展示了基线和七种集成器在训练数据为 SO 或 Jira、测试数据为 GitHub 时的宏平均 F1 分数和精确率。最左侧的八个柱形展示了基线的宏平均 F1 分数 ＋ ，以及在训练数据为 SO、测试数据为 GitHub 时七种集成器的宏平均 F1 分数。我们还在这八个柱形下方标注了基线名称，在此例中为 SentistrengthSE。在

此情况下，我们看到，由两种表现最佳的深度学习情感检测工具（BERT4SentiSE、SentiMoji）和 SentistrengthSE 组成的集成，在基线和各集成器中取得了最高的宏平均 F1 分数 0.82。总体而言，按宏平均 F1 分数衡量，基于集成的工具在三种跨平台设置中表现最佳，而独立情感检测工具（即基线）在另外三种跨平台设置中表现最佳。在基线表现最佳的设置中，SentistrengthSE 在两种设置中胜出（在 SO 或 GitHub 上训练后于 Jira 上测试），BERT4SentiSE 则在一种设置中胜出（在 GitHub 上训练、在 SO 上测试）。SentistrengthSE 在 Jira 数据集上表现优异的原因之一，是 SentiStrengthSE 的规则是在研究 Jira 平台后开发的 [43]。因此，在我们的 Jira 数据集中进行测试时，该工具可能比其他工具具有一定优势。在多数投票集成器取得最高 F1 分数的三种设置中，最佳组合始终是两种表现最佳的深度学习情感检测工具（BERT4SentiSE、SentiMoji）与 SentistrengthSE 的组合。

在每种跨平台设置下，表现最佳的集成工具与基线（即表现最佳的独立工具）之间的性能差异几乎可以忽略不计。根据图5，基于多数投票的集成工具在三种跨平台设置下表现最佳：在 SO 上训练后于 GitHub 上测试（最佳集成工具的宏平均 F1 分数为 0.82），或在 Jira 上训练后于 GitHub 上测试（$F1 = 0.81$）；以及在 Jira 上训练后于 SO 上测试（$F1 = 0.82$）。在这三种设置下，表现最佳的基线分别是前两种情况下的 SentiStrengthSE（$F1 = 0.78$）和最后一种情况下的 BERT4SentiSE（$F1 = 0.79$）。因此，在这三种设置下，相较于基线，使用集成工具所能带来的最大性能提升为 0.03。因此，在跨平台设置下对六种工具进行多数投票并不实用，因为宏平均 F1 分数仅提升了 0.03。

研究问题$_4$ 深度学习和非深度学习情感检测工具在多大程度上能够在跨平台设置中互为补充? 在我们的研究中，表现最佳的两种深度学习情感检测工具（BERT4SentiSE、SentiMoji）在所有六种软件工程专用情感检测工具中彼此的一致程度最高、不一致程度最低。基于两种表现最佳的深度学习情感检测工具（BERT4SentiSE + SentiMoji）进行多数投票的集成，或两种表现最佳的深度学习情感检测工具（BERT4SentiSE + SentiMoji） +基于规则的工具（SentistrengthSE），在跨平台设置中的表现略优于单独使用情感检测工具。每种跨平台设置中，表现最佳的集成与基线（即表现最佳的单独工具）之间的性能差异几乎可以忽略不计。

4 讨论

本节将讨论我们在研究过程中发现的主要主题。

4.1 对所研究数据集进行相似性分析，能否解释深度学习情感检测工具在跨平台设置下的数据集中性能下降的原因?

在我们研究的三个数据集中，所有用于软件工程的有监督情感检测工具在跨平台设置下都出现了性能下降。尽管两种深度学习情感检测工具（BERT4SentiSE 和 SentiMoji）在大多数跨平台设置下的表现优于其他有监督情感检测工具，但它们的性能仍因数据集而异。例如，正如我们在 RQ_1, 中指出的，当工具在 Jira 数据集上训练、再在另外两个数据集上测试时，性能下降最为严重。直观而言，两个数据集的文本内容越相似，我们预期工具在这两个数据集之间进行跨平台测试时的性能下降就越小。因此，了解所研究数据集的内容相似度与所研究工具在跨平台设置下的性能是否相关十分重要。

4.1.1 Approach

在这项分析中，我们重点关注RQ_1, 中表现最好的两种用于软件工程的深度学习情感检测工具，即 BERT4SentiSE 和 SentiMoji。对于每种工具，以及涉及两个数据集的每种跨平台设置（即一个数据集用于训练，另一个用于测试），我们分析该工具在跨平台设置下的性能与这两个数据集之间的文本相似度是否存在相关性。我们使用两种标准指标来计算两个数据集之间的相似度：余弦相似度和詹森-香农散度。下文将讨论如何使用这两种指标计算给定两个数据集之间的相似度。

(1) **余弦相似度** [33] 有助于衡量两篇文档之间的文本相似度，而不受文档大小的影响。对于每个数据集，我们在移除停用词后，选取排名前 10,000 的一元词（停用词取自 Python 的 'nltk' 库）。随后，我们将这些词合并，构建所有数据集的统一词汇表（V）。我们使用 V 根据数据集中每个句子中词语的出现频率，为每个数据集创建向量化表示。对于任意两个数据集（例如 A、B），我们将它们得到的向量代入公式 2，计算两个数据集之间的相似度。我们使用 Scikit-learn 库中的余弦相似度方法[3]。

$$Cosine_{Similarity}(A,B) = \frac{A.B}{|A||B|} \tag{2}$$

(2) **詹森-香农散度** [17] (D_{JS}) 是库尔贝克-莱布勒散度的一种对称且平滑的形式 D_{KL}, $D_{KL}(A||B) = \sum_i^N a_i log\frac{a_i}{b_i}$。 D_{KL} 是 A 和 B 的概率分布。当任一元素 $a_i \in A$的概率为零时， D_{KL} 没有定义。而 D_{JS} 是词汇表 V 中每个词的概率分布。该指标最初用于计算两个数据集 A 和 B 之间的相似度，公式如下:

$$D_{JS}(A,B) = \frac{1}{2}[D_{KL}(A||B) + D_{KL}(B||A)] \tag{3}$$

[3] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html

表 8：跨平台设置下数据集相似度值与宏平均 F1 分数之间的相关性

	BERT4SentiSE		SentiMoji	
	p 值	效应量	p 值	效应量
詹森-香农	0.04*	大（0.6）	0.003**	大（1.0）
余弦	0.002**	大（1.0）	0.003**	大（1.0）

显著性代码：***$p < 0.001$，**$p < 0.01$，*$p < 0.05$

根据公式 4，如果值为 $D_{JS} = 0$，则表示这两个数据集最相似；值为 1.0 则表示它们完全不同。鉴于余弦相似度对相似度采用相反的定义（即 1 表示完全相似），我们将公式 4修改如下：

$$D_{JS}(A, B) = 1 - \frac{1}{2}[D_{KL}(A||B) + D_{KL}(B||A)] \tag{4}$$

因此，在我们的案例中，当 $D_{JS} = 1.0$ 时，表示数据集最相似；当 $D_{JS} = 0.0$ 时，表示数据集彼此完全不同。我们按如下方式计算每对数据集（A、B）的指标。与余弦相似度方法类似，我们将每个数据集表示为向量。对于任意两个数据集（例如 A、B），我们将其生成的向量代入公式 4，以计算这两个数据集之间的 D_{JS} 相似度。

我们使用曼-惠特尼 U 检验 [35]计算两个数据集之间的相似度与工具在跨平台设置中的性能之间的相关性。MW 检验是一种非参数检验，适用于数据不服从正态分布的情况。我们使用 p 值报告相关性的显著性，并使用 Cliff δ 值报告效应量。我们按照 δ Romano 等人 [46]的标准进行解释：

$$\text{Effect Size} = \begin{cases} negligible\ (N), & \text{if } |\delta| \leq 0.147 \\ small\ (S), & \text{if } 0.147 < |\delta| \leq 0.33 \\ medium\ (M), & \text{if } 0.33 < |\delta| \leq 0.474 \\ large\ (L), & \text{if } 0.474 < |\delta| \leq 1 \end{cases} \tag{5}$$

效应量类别（可忽略、小、中或大）量化了这些分布之间的差异。

4.1.2 Results

在跨平台设置中，数据集的内容相似度与其宏平均 F1 分数之间存在统计学上显著的相关性。在表8中，我们使用 MW U 检验报告数据集相似度与工具性能值之间相关性分析的 p 值，并展示效应量。在图6中，我们展示数据集之间的相似度得分，并比较 RQ_1 中表现最好的两种深度学习情感检测工具（即 BERT4SentiSE 和 SentiMoji）在不同数据集上的宏平均 F1 分数。例如，在第一个区块（即在 GitHub 上测试）中，我们有两组柱状图：GitHub 与 SO 数据集之间的相似度得分，以及 BERT4SentiSE 和 SentiMoji 在跨平台设置下的宏平均 F1 分数（即在 SO 上训练并在 GitHub 上测试）。Jansen

数据集之间的相似度得分至少为 0.87，余弦相似度至少为 0.93。因此，尽管数据集来自不同的平台，它们仍包含相当多的相似词语。所以，两个数据集越相似，我们在跨平台设置中预期的性能下降就越小。

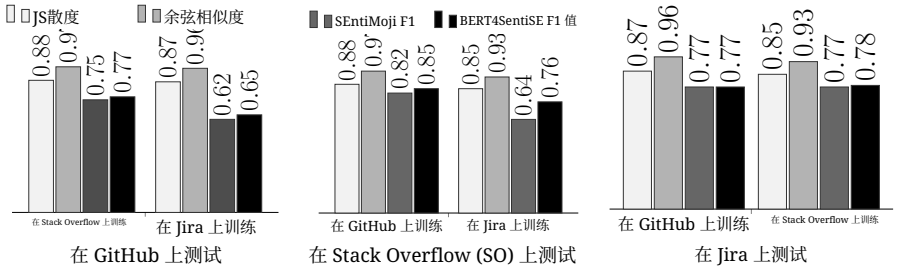


图 6：跨平台设置下，各数据集之间的相似度得分与各工具（BERT4SentiSE 和 SentiMoji）的宏平均 F1 分数。图中包含三组柱状图，每组分别对应三个数据集中的一个。对于第一组，“在 GitHub 上测试”表示工具 BERT4SentiSE 在另外两个软件工程数据集以及非软件工程数据集（即 Twitter）上训练，然后在 GitHub 上测试。“在 GitHub 上测试”对应的图例包括：在 SO（即 Stack Overflow）上训练、在 Jira 上训练，以及在 Twitter 上训练（为简洁起见，后两项省略了“在……上训练”）。对于每个“在¡数据集¿上训练”，都有三根柱子：第一根（白色柱）表示训练数据集与测试数据集之间的 JS 散度分数（此处为 SO 与 GitHub 之间），第二根（灰色柱）表示训练数据集与测试数据集之间的余弦相似度分数（即 SO 与 GitHub 之间），第三根（黑色柱）表示 BERT4SentiSE 在此类跨平台设置下的 F1 分数（即在 SO 上训练，然后在 GitHub 上测试）。

表 9：与平台内设置相比，BERT4SentiSE 和 SentiMoji 在跨平台设置下的数据集内容相似度与性能下降（宏平均 F1 分数）的分布

Test	训练	相似度得分		性能下降百分比	
数据集	数据集	JS散度	余弦距离	SentiMoji	BERT4SentiSE
GitHub	SO	0.88	0.97	17	16
	Jira	0.87	0.96	31	29
SO	GitHub	0.88	0.97	5	3
	Jira	0.85	0.93	26	14
Jira	GitHub	0.87	0.96	5	7
	SO	0.85	0.93	5	6

显著性分析无法解释跨平台设置中 Jira 数据集与非 Jira 数据集之间性能表现差异的程度。我们在 RQ_1 中的发现表明，当这两种深度学习情感检测工具使用 Jira 数据集训练、再使用另外两个数据集测试时，性能下降最大。

表10：跨平台设置下误判为中性和误判为非中性的预测分布。（误判为中性 =预测是将非中性判为中性，误判为非中性 = 预测是将中性句子判为非中性）

SD	正确率	在 GitHub 上训练		在 SO 上训练		在 Jira 上训练	
Tool	误分类	测试 Stack Overflow	测试 Jira	测试 GitHub	测试 Jira	测试 GitHub	测试 SO
SentiMoji	误判为中性	61	22	75	28	**94**	**97**
	误判为非中性	34	74	15	69	**2**	**3**
BERT4SentiSE	误判为中性	41	23	69	26	**91**	**88**
	误判为非中性	52	72	17	69	**3**	**10**

表 8 中的显著性分析表明，这种性能下降可能归因于 Jira 数据集与另外两个数据集之间的相似性（或缺乏相似性）。然而，图6 显示，Jira 数据集与 SO 数据集和 GitHub 数据集之间的内容相似度得分，几乎与 SO 数据集和 GitHub 数据集彼此之间的得分相同。此外，只有在使用 Jira 数据集训练工具时，工具的性能才会大幅下降。例如，BERT4SentiSE 使用 Jira 数据集训练、使用 GitHub 数据集测试时，宏平均 F1 分数为 0.65；而使用 GitHub 数据集训练、再使用 Jira 数据集测试时，F1 分数为 0.77。表 9中展示了各跨平台设置下宏平均 F1 分数相较于平台内设置的下降百分比。例如，表9前两行显示，使用 SO 数据集训练、再使用 GitHub 数据集测试时，BERT4SentiSE 的性能相较于其在 GitHub 数据集上训练并测试时下降了 16%。然而，使用 Jira 数据集训练 BERT4SentiSE、再使用 GitHub 数据集测试时，性能下降了 29%。GitHub 与 SO 之间的余弦相似度为 0.97，GitHub 与 Jira 之间的余弦相似度为 0.96。工具使用 GitHub 数据集训练、使用 SO 数据集测试时，性能下降了 3%；而使用 Jira 数据集训练、使用 SO 数据集测试时，性能下降了 14%。SO 与 GitHub 之间的余弦相似度为 0.97，SO 与 Jira 之间的余弦相似度为 0.93。因此，使用 Jira 数据集时出现的性能下降，可能是由内容相似度以外的因素造成的。

相反，Jira 数据集中极性类别的分布或许能提供另一种解释（见表 1）。GitHub 和 SO 的三种极性类别分布都较为均衡（每种极性类别约占句子的 27–35%），而 Jira 数据集则存在类别不平衡，正向类和负向类分别仅占 19% 和 14%。这意味着，与使用另外两个数据集训练时相比，使用 Jira 数据集训练的工具无法获得足够的两种极性类别信息。Jira 数据集中的不平衡也可能导致极性标注出现偏差（例如，更倾向于中性类），从而造成表 6中轻微不一致的比例较高。尽管每种工具在模型训练时都经过专门设计，以处理类别不平衡问题，这种情况仍然发生了。

在类别不平衡的 Jira 数据集（67% 为中性）上训练时，最主要的错误情况（约占 90%）是所有监督式工具都将非中性句子预测为中性极性。 从表 3可以看出，在 Jira 数据集上训练、并在非 Jira 数据集上测试时，SentiMoji 和 BERT4SentiSE 对两个非中性类别的精确率都较高，而召回率较低。同时，这些工具对中性类的召回率较高，而精确率较低。例如，在 Jira 数据集上训练并在

GitHub 数据集上，BERT4SentiSE 对负类的精确率为 0.90，召回率为 0.43。该工具对正类的精确率为 0.91，召回率为 0.51。同时，报告显示，该工具对中性类的召回率较高（0.97），精确率较低（0.59）。这一观察结果与跨平台设置下的 SentiMoji 一致。表 10 列出了监督式情感检测工具在跨平台设置下训练时性能下降的详细情况。约 95% 至 99% 的误分类属于轻微错误（即“误判为中性”和“误判为非中性”之和）。约 88% 至 97% 的错误是由“误判为中性”导致的，也就是说，人工标注为非中性，但工具预测为中性。例如，BERT4SentiSE 在 Jira 数据集上训练、在 GitHub 上测试时，宏 F1 值下降约 29%；其所有错误中，94% 属于轻微错误，6% 属于严重错误。我们的分析表明，在这些设置下，91% 的错误属于 “ 误判为中性 ”，只有 3% 属于“误判为非中性”。Novelli 等人 [38] 也报告了监督式工具在跨平台设置下出现类似的性能下降；而在平台内设置下，即使使用类别不平衡的 Jira 数据集，基于深度学习的情感检测工具的表现也优于所有非深度学习情感检测工具。如第 2.1节所述，本研究采用了原作者在各自研究中报告的、针对所研究监督式工具的最佳模型配置、超参数和设置。

4.2 跨平台相似度与准确率

在第4.1节中，我们观察到，本研究使用的三个软件工程数据集在文本内容方面具有相当高的相似度（至少为 85%），而且，两个数据集之间的相似度与表现最佳的深度学习情感检测工具 BERT4SentiSE 在这两个数据集上的跨平台性能之间存在统计显著的正相关。为了更好地了解这一观察是否合理，我们还检查了三个软件工程工具与一个非软件工程数据集之间的相似度。该数据集名为“tweeter-airline-sentiment”，可在 Kaggle[4]上获取。该数据集包含消费者发布的 14,640 条推文（以及相应的情感极性标签），内容涉及美国六家主要航空公司。我们使用这个 Twitter 数据集训练 BER4SentiSE，然后在我们的三个软件工程数据集上测试该模型。我们还使用在 RQ_2, 中介绍的两种度量，计算了 Twitter 数据集与我们的三个软件工程数据集之间的相似度，即 Jensen-Shannon 散度和余弦相似度。在图7中，我们展示了三组柱状图。第一组使用六根柱（两种相似度度量各两根）展示 GitHub 数据集与其他三个数据集（Twitter、GitHub、Jira）的相似度。第一组还展示了 BERT4SentiSE 在其他三个数据集上训练后，于 GitHub 数据集上测试时的宏平均 F1 分数。第二组展示了“在 SO 数据集上测试”的类似数据，第三组则展示了“在 Jira 上测试”的数据。在所有三个测试数据集中，Twitter 数据集与其他数据集相比，相似度得分最低（最高仅为 0.74）。在 Twitter 数据集上训练时，模型在三个软件工程数据集上的跨平台 F1 分数也明显偏低（在 Twitter 上训练时为 0.51–0.65，而在软件工程数据集上训练时至少为 0.77）。这些结果证实，软件工程数据集

[4] https://www.kaggle.com/crowdflower/twitter-airline-sentiment

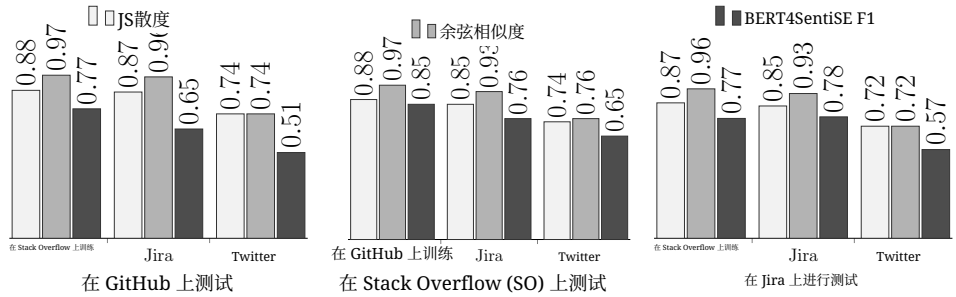


图 7：Twitter 与其他数据集之间的相似度得分与 BERT4SentiSE 在跨平台设置下的宏平均 F1 分数。图中有三组柱状图，分别对应三个数据集中的每一个数据集。对于第一组，“在 GitHub 上进行测试”表示工具 BERT4SentiSE 使用另外两个软件工程数据集以及非软件工程数据集（即 Twitter）进行训练，然后在 GitHub 上进行测试。“在 GitHub 上进行测试”的图例包括：在 SO（即 Stack Overflow）上训练、在 Jira 上训练，以及在 Twitter 上训练（为简洁起见，后两项省略了“在.....上训练”这一表述）。对于每个“在¡dataset¿上训练”，都有三根柱：第一根柱表示“在 SO 上训练”（即白色柱），代表训练数据集与测试数据集之间的 JS 散度分数（本例中为 SO 与 GitHub 之间的 JS 散度分数）；第二根柱（即灰色柱）表示训练数据集与测试数据集之间的余弦相似度分数（即 SO 与 GitHub 之间的余弦相似度分数）；第三根柱（即黑色柱）表示 BERT4SentiSE 在此类跨平台设置下的 F1 分数（即在 SO 上训练，然后在 GitHub 上进行测试）。

彼此之间的相似度确实高于它们与 Twitter 这类非软件工程数据集之间的相似度。相似度的提高确实会影响工具 BERT4SentiSE 在软件工程数据集跨平台设置下的性能。

4.3 工具的学习曲线

我们对这些工具的性能分析发现，在平台内设置下，深度学习情感检测工具 BERT4SentiSE 在三个数据集上均表现最佳。在跨平台设置下，它也是有监督情感检测工具中表现最佳的工具。另一个深度学习情感检测工具 SentiMoji 在各数据集的平台内设置下表现第二好。SentiMoji 在大多数跨平台设置下的表现也优于浅层学习情感检测工具。这两个深度学习情感检测工具适合部署于范围广泛的数据集，尤其是在能够针对每个数据集重新训练工具的情况下。因此，了解它们需要多少训练数据才能提供相当好的性能非常重要。总体而言，要取得更好的性能，深度学习工具比浅层学习工具需要更多数据。不过，鉴于这两个工具都使用预训练嵌入，它们可能不需要大量训练数据。在图8中，我们展示了三个图表，分别对应三个数据集。对于每个图表，我们展示了随着训练数据规模增加，各工具的表现如何变化。为了生成学习曲线，我们在训练每个工具时逐步增加训练样本的数量，然后使用相同的、由 30% 样本组成的留出测试集对其进行测试。首先，我们抽取

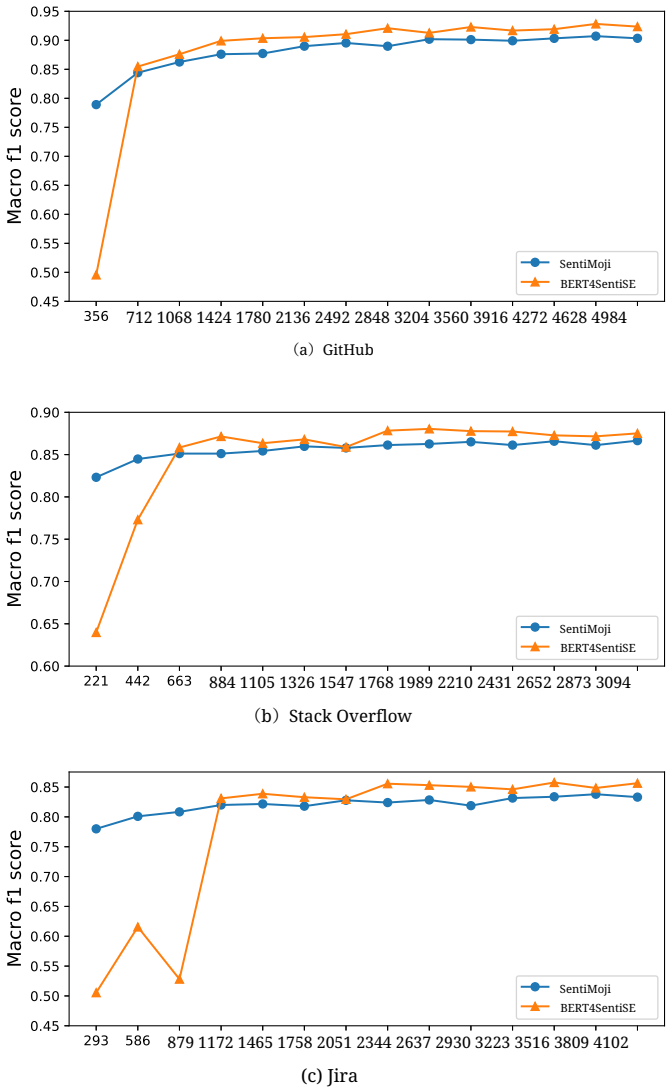


图 8：SentiMoji 和 BERT4SentiSE 在平台内设置下于 (a) GitHub、(b) Stack Overflow 和 (c) Jira 上的学习曲线

训练集的 5%，采用分层抽样，以保持类别分布不变；随后每次将训练集增加 5%，并使用之前留出的测试集对模型进行测试。Novielli 等人也采用了类似的方法 [38]，以绘制 Senti4SD 和 SentiCR 在同三个数据集上的学习曲线。

在这三个数据集上，使用 5%–10% 的训练数据时，SentiMoji 的表现（宏平均 F1 分数）更好。然而，此后，BERT4SentiSE 在两个数据集（GitHub 和 SO）上的表现开始超过 SentiMoji。对于另一个数据集（Jira），BERT4SentiSE 在使用约 20%–25% 的

表 11：跨平台设置下，工具 BERT4SentiSE 和 SentiMoji 在三个数据集的 400 条人工分析样本中的错误类别分布

类别	GitHub	SO	Jira
标注中的主观性	59	76	44
一般错误	11	34	67
隐含情感	4	13	49
语境/语用	23	16	29
领域特定	14	16	38
礼貌	18	23	8
比喻性语言	9	6	11
Sum	138	184	246

训练数据后开始超过 SentiMoji。使用 20%–25% 的训练数据时，这两种工具的宏平均 F1 分数均超过 0.80–0.85。此后，这两种工具的性能提升速度有所放缓。在使用 30%–40% 的数据进行训练后，这两种工具在 GitHub 上的 F1 分数最高可超过 0.9，在 SO 和 Jira 上则可达到 0.85–0.88；此后性能提升微乎其微。这意味着，在无法创建大规模基准数据、但创建规模较小的数据（例如 500–1500 个标注句子）并非难事的现实场景中，这两种工具都是可行的选择。

4.4 错误类别分布

在表 11中，我们展示了 RQ_3 中报告的六种错误类别在三个数据集（GitHub、SO 和 Jira）中的分布。请注意，样本量为 400 个句子，这些句子均被表现最佳的两个深度学习情感检测工具（BERT4SentiSE 和 SentiMoji）误判。这是一个随机样本。根据各数据集中实际类别的分布，我们发现各数据集中的误分类案例比例几乎相同。总体而言，大多数误分类案例来自 Jira 数据集，这也是这两个工具表现最差的数据集。在七种错误类别中，工具使用 GitHub 数据集进行训练时，最常见的一类是标注中的主观性。在跨平台设置下，这种错误在 Jira 数据集中最为常见，即 Jira 数据集的人工标注存在不一致（RQ_3）。工具使用 SO 数据集进行训练时，最常见的一类是礼貌。这是因为我们发现 SO 数据集中与礼貌相关的句子较少。因此，尽管 GitHub 和 Jira 都包含更多带有礼貌表达的非中性句子，这些工具仍未能充分学习与礼貌相关的情感极性。

当工具使用 Jira 数据集进行训练时，另外七类错误中出现频率最高的类别为：1. 一般错误，2. 隐含情感，3. 语境/语用，4. 领域特异性，以及 5. 比喻性语言。错误类别集中度较高有几个原因。首先，与另外两个数据集相比，Jira 数据集中的表情符号句子较少，这意味着工具无法正确学习与表情符号相关的词语中的符号表示。其次，正如我们在 RQ_3,中指出的，Jira 数据集对表情符号的标注也存在不一致之处，这可能会让工具在学习和后续预测时产生混淆

在跨平台设置中。第三，Jira 中的大多数句子都比另外两个数据集中的句子短，因此在工具学习过程中提供的语境更少。在图 9中，我们展示了各数据集中不同词数范围的句子所占百分比。例如，Jira 数据集中约有 6% 的句子少于五个词，而 SO 数据集中少于五个词的句子约占 2%。这就是为什么使用 Jira 数据集训练的工具在语境/语用方面出现更多错误。因此，Jira 数据集可以通过更一致的标注和更丰富的数据点（例如更长的句子）获益。

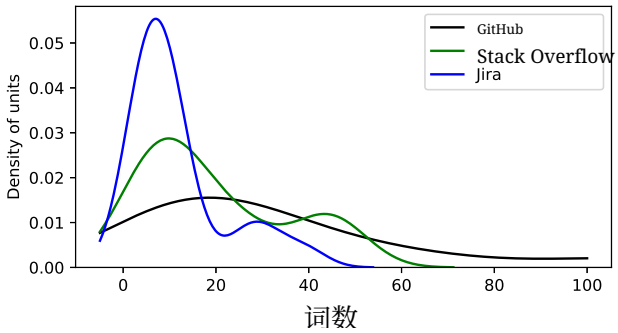


图 9：每个句子的词数分布

4.5 礼貌极性方面的 *Issues*

正如我们在 RQ$_3$，中指出的，包含礼貌表达的句子在各数据集中并未得到一致标注。例如，在 Jira 数据集中，我们发现相似的句子在不同位置被标注为中性或正向。礼貌表达似乎使三个数据集的人工编码员感到困惑，导致监督式工具在预测含有礼貌表达的句子的标签时容易出错。为探究这一问题的严重程度，我们进行了一项简单实验：首先选取含有礼貌表达的句子样本，然后确定在跨平台设置下，这些句子中有多少被误分类。首先，我们通过搜索句子中的三个关键词——“thanks”（谢谢）、“thank you”（谢谢你）和“sorry”（抱歉）——来找出含有礼貌表达的句子样本。在表12中，我们展示了各数据集中此类句子的分布。三个数据集中，包含礼貌表达的句子约有 2,700 条（即三个数据集中所有句子的 15.5%）。在这三个数据集中，Jira 中包含礼貌表达的句子数量最多，约为 1,386 条（24%）；其次是 GitHub 的 928 条（13%）和 SO 的 430 条（10%）。表12 的最后三列显示了在本研究中使用表现最佳的深度学习情感检测工具 BERT4SentiSE 时，各数据集中含有礼貌表达的句子在跨平台设置下被误分类的百分比。例如，当工具在 Jira 上训练并测试时（即平台内设置），37.5% 的含礼貌表达的句子被误分类。然而，当工具在 Jira 上训练、但在

表12：在各数据集的跨平台设置中，BERT4SEntiSE 和 SEntiMoji 对具有礼貌的句子及其误分类的分布

训练 on	礼貌		在以下测试时的误分类百分比		
	#	%	Jira	GitHub	SO
Jira	1,386	24%	37.5%	4.1%	0.8%
GitHub	928	13%	36.2%	9.3%	1.5%
SO	430	10%	36.3%	6.0%	0.8%

GitHub 上测试时，GitHub 中含有礼貌表达的句子仅有 4.1% 被误分类；SO 数据集中的误分类比例则仅为 0.8%。事实上，Jira 数据集中误分类案例的比例与其他数据集相比差异很大。当 BERT4SentiSE 在 GitHub 上训练并在 Jira 上测试时，Jira 中含有礼貌表达的句子仍有 36.2% 被误分类。当工具在 SO 上训练并在 Jira 上测试时，仍有 36.3% 的句子被误分类。分析结果证实了我们先前的观察：要提升工具在所研究的三个数据集上的跨平台表现，我们可以通过减少 Jira 数据集中礼貌表达标注的不一致性来获益。

4.6 SentiStrengthSE 在跨平台设置中的优势

在 RQ2（见第3.2节）中，我们发现，在六种跨平台设置中的五种设置下，面向 SE 的基于规则的情感检测工具 SentiStrengthSE 优于其他所有面向 SE 的有监督浅层工具和深度学习工具；在其中五种设置中的四种设置下，它还以微弱优势胜过深度学习情感检测工具 BERT4SentiSE。这一发现促使我们在 RQ3（见第 3.3节）中对这些工具进行误分类分析。在图 1中，我们展示了 BERT4SentiSE 和 SentiMoji 均判断错误的 400 个句子的分布。在图3中，我们展示了 SentiStrengthSE 判断错误的 93 个句子的分布。对于深度学习情感检测工具，最常见的错误类别是“标注中的主观性”（占 29%）。对于 SentiStrengthSE，最常见的错误类别是“领域特异性”（占 39%）。直观而言，这一发现是合理的，因为像 SentiStrengthSE 这样的基于规则的工具无法从数据集中学习特定领域的洞见，因为它无法在该数据集上重新训练。而像 BERT4SentiSE 这样的有监督情感检测工具，在数据集上训练后可以学习领域特异性。鉴于我们的跨平台数据集相当相似（见第 4.1节），与基于规则的情感检测工具相比，有监督情感检测工具在跨平台和平台内设置下都更适合处理此类领域特异性。例如，考虑 Jira 数据集中的这个句子 *“Bug Flavio more? Seriously”*。其人工标注为“负向”；鉴于底层语境涉及系统中存在“更多缺陷”，这一标注是正确的。BERT4SentiSE 等监督式工具正确地将该句子标注为“负向”。然而，SentiStrengthSE 将其标注为中性，因为它无法理解底层语境。然而，尽管此类训练能够帮助有监督情感检测工具更好地处理领域特异性，但如果数据集的人工标注存在问题，进而导致“标注中的主观性”错误，那么这类训练也可能成为有监督工具的问题。数据集中的人工标注不一致会使有监督情感检测工具感到困惑，

因为它在给定句子上训练时学到的内容（例如“谢谢你”）可能不适用于测试数据集，前提是相同或相似的句子在训练数据集和测试数据集中的标注不同。例如，在第4.5节中，我们提供了包含礼貌表达的句子示例，并展示了这些句子在不同数据集中的标注不一致的证据。对于像 SentiStrengthSE 这样的基于规则的情感检测工具，当人工标注偏离规则时，此类不一致也可能导致一些误分类。例如，工具始终认为“谢谢你”应标注为正向，而人工编码者有时将其标注为正向、有时将其标注为中性，就属于这种情况。然而，标注中的主观性可以超出礼貌表达的范畴，因此对有监督情感检测工具的危害可能更大，因为这些工具在数据集上进行训练和测试，因而更容易受到底层细微差异的影响。

5 研究发现的启示

本研究的发现可为软件工程（SE）领域的以下利益相关者提供指导：**(1)软件工程数据科学家** 可据此判断，在多种软件工程数据仓库中，使用情感检测工具分析软件工程制品是否可行；**(2) 软件工程管理者** 可据此确定最合适的情感检测工具，以了解软件工程数据仓库中反映的组织情感；**(3) 软件工程架构师** 可据此设计能够纳入软件工程专用情感检测工具、用于分析软件工程数据仓库的软件工程流水线；**(4) 软件工程研究人员** 可据此提升软件工程专用情感检测工具在多种软件工程数据源（即跨平台设置）上的性能。下文将结合第 3 节和第 4节的研究结果讨论其启示。

• **软件工程数据科学家**。微软等科技公司如今几乎在每个软件工程团队中都配备了数据科学家 [4]。数据科学家可以开发情感检测工具，分析组织内多种软件工程数据仓库。事实上，利用情感检测进行软件分析，已成为支持软件库选择、缺陷修复等多种开发任务的重要领域 [63, 59]。软件工程领域的数据科学家可以借助本研究的发现，判断深度学习情感检测工具能否以及如何应用于跨平台设置。具体而言，数据科学家可以从本研究中获得以下见解：

(1) 数据集相似度与类别不平衡问题。 我们在 RQ3 中的分析发现，在跨平台设置下，我们研究中的任意两个数据集之间的相似度，与两种表现最佳的深度学习情感检测工具（BERT4SentiSE 和 SentiMoji）在这两个数据集上的性能之间，存在统计显著的正相关。我们在第 4.2 节中进一步验证了这一假设。结果显示，对于 Twitter 这类非软件工程数据集，当我们使用该数据集训练工具、再用其测试软件工程数据集时，数据集相似度和工具性能都会下降。然而，正如我们在 RQ_3,中指出的那样，这一观察无法解释工具在 Jira 数据集上训练、再在另外两个数据集上测试时性能下降的现象。正如我们在 RQ3 中指出的，与另外两个数据集相比，Jira 数据集的两个极性类别（正向、负向）明显不平衡。因此，软件工程团队中的数据科学家可以依据上述两点采取以下做法：(a) 让情感检测数据集在三个极性类别之间更加均衡；(b) 两个软件工程数据集越相似，跨平台设置下这两种工具（BERT4SentiSE、SentiMoji）的性能下降幅度就越小

越相似，我们预计在跨平台设置下这两种工具（BERT4SentiSE、SentiMoji）的性能下降就越小。

(2) 领域特异性与跨平台性能下降。 基于规则的工具 SentistrengthSE 在六种跨平台设置中的五种设置下，优于表现最佳的有监督情感检测工具 BERT4SentiSE。其中四种设置涉及 Jira 数据集，例如，使用 Jira 训练工具，再在另外两个数据集上测试；或者使用另外两个数据集训练工具，再在 Jira 数据集上测试。如上所述，Jira 数据集存在类别不平衡问题，这意味着在 Jira 数据集上训练时，有监督工具可能无法从类别不平衡的数据中获得足够的信息。我们对数据集的分析显示，Jira 数据集中大约有不到 5% 的样本少于五个词。这些短句有时无法提供足够的语境信息。Jira 数据集中的这两个缺点（即类别不平衡和短句）导致两种有监督情感检测工具在 Jira 上训练、在其他数据集上测试时表现不佳。因此，正如第（4.1）节所讨论的，尽管按照作者针对平台内设置所述的最佳配置（模型设置、超参数、预处理指南）配置了情感检测工具，但当这些工具在 Jira 上训练、在非 Jira 数据集上测试时，它们约 94% 至 99% 的错误都源于将非中性句子预测为中性。此外，在所有数据集中，对于两种深度学习工具 BERT4SentiSE 和 SentiMoji，Jira 数据集涉及“一般错误”的错误数量也是最多的（见第 4.4 节）。这意味着在测试该数据集时，这两种工具未能正确处理 Jira 数据集。基于 Ortu 等人创建的 Jira 数据集制定规则，开发出了基于规则的工具 SentistrengthSE。 [43]因此，SentistrengthSE 在跨平台设置下也可能无法理解底层语境。事实上，我们在研究问题4中的错误分析发现，SentistrengthSE 在三个数据集上的错误中，数量最多的是由于该工具无法确定数据中的底层语境（即领域特异性）所致。因此，软件工程领域的数据科学家需要进一步研究软件工程专用情感检测工具，然后才能断言像 SentistrengthSE 这样的基于规则的工具在跨平台设置下具有优越性。

(3) 深度学习情感检测工具中的预训练 ***Transformer*** 与循环神经网络架构。在本研究中，我们考察了三种用于软件工程的深度学习情感检测工具：BERT4SentiSE [6], SentiMoji [13],和 RNN4SentiSE [7]。选择这三种工具，是因为在我们分析期间，它们被认为是表现最佳的用于软件工程的深度学习情感检测工具。在我们的三个数据集中，无论是平台内还是跨平台设置，BERT4SentiSE 都是有监督情感检测工具中表现最佳的；而另一种深度学习情感检测工具 RNN4SentiSE，则是所有工具中表现最差的。我们对数据和 RNN4SentiSE 的分析表明，这并非源于工具本身某些明显的缺陷，而是由工具的设计和使用方式所致。例如，BERT4SentiSE 使用 BERT 的预训练 Transformer 架构。SentiMoji 使用双向长短期记忆网络架构，并且也在大规模的 Twitter 及其他表情符号数据语料库上进行了预训练。RNN4SentiSE 使用简单的循环神经网络架构，而且也没有经过预训练。事实上，先前针对平台内设置的研究发现，BERT4SentiSE 的表现优于 RNN4SentiSE [6]。因此，BERT4SentiSE 在跨平台设置中的表现也优于 RNN4SentiSE，这一点合乎直觉；如果结果相反，就会引发对底层模型、数据乃至

研究方法的质疑。事实上，预训练模型在其他领域表现出色的情况，已在多种文本分类任务中得到报告。正如我们在 RQ2 中指出的那样，本研究中表现最佳的有监督情感检测工具 BERT4SentiSE，与表现最佳的基于规则的工具 SentiStrengthSE 相比，性能差异也几乎可以忽略（0.01 - 0.03）；但在平台内设置中，BERT4SentiSE 的表现远胜于 SentiStrengthSE。因此，软件工程数据科学家可以专注于改进 BERT4SentiSE 等情感检测工具，例如增加隐藏层数量、评估不同的损失/优化参数等。

• **软件工程架构师**。鉴于软件工程领域的架构师负责设计软件系统，他们可以利用我们的研究结果，决定将何种工具或工具组合纳入机器学习（ML）软件流水线，以支持分析不同软件工程数据集中的情感极性。具体而言，以下研究发现有助于他们进行决策。

(1) 在平台内和跨平台设置中，性能最佳的软件工程专用情感检测工具与表现尚可的工具之间的权衡。由于需要不断更新模型以及用于训练和测试模型的数据，将机器学习流水线嵌入软件开发生命周期可能颇具挑战且复杂 [3]。对于负责将软件工程专用情感检测工具嵌入软件系统的软件工程架构师而言，软件仓库中的数据种类繁多，而且正如我们的研究（RQ1-RQ2）所观察到的，该工具在跨平台设置中的性能会下降，因此这项工作颇具挑战性。这意味着软件工程架构师必须根据数据类型考虑工具的不同组合。当每种工具都需要用新数据更新，以确保工具不会随时间推移而发生漂移、性能退化时，这样的配置会更加复杂。在这种情况下，能够适用于所有平台且表现尚可的工具或许更为理想，尤其是当该工具在跨平台设置中的性能不会大幅下降时。在平台内设置中，表现最佳的深度学习情感检测工具 BERT4SentiSE 优于其他工具。在跨平台设置中，该工具的表现也位居第二，仅以微弱差距落后于基于规则的工具 SentiStrengthSE。因此，对于需要在生产环境中分析不同软件工程数据集情感极性的软件工程架构师而言，BERT4SentiSE 可能是嵌入软件开发生命周期的最佳选择。

(2) 学习曲线与数据需求。有监督情感检测工具的成效很大程度上取决于数据的数量和质量。直观而言，数据越多、越具代表性，我们就越有理由期待有监督情感检测工具能够更好地识别情感极性标签。只要不同平台也能从领域中立数据中学习词汇/模式，将迁移学习原理应用于预训练深度学习情感检测工具，就能降低对大量数据的依赖。事实上，正如我们在第 4.3 节中发现的那样，本研究中表现最好的两种预训练深度学习情感检测工具（BERT4SentiSE、SentiMoji）仅用全部训练数据的 30% 进行训练，就能在平台内设置中取得最佳表现。考虑到数据收集和维护会极大地增加软件开发生命周期的复杂性，软件工程架构师可以在软件工程专用软件产品中采用 BERT4SentiSE 和 SentiMoji 等工具，从而获益。

• **软件工程研究人员**。从 2016年还没有软件工程专用情感检测工具，到如今已有多种性能出色的软件工程专用情感检测工具，能够在多种软件工程数据集上胜过跨领域情感检测工具，我们已经取得了长足进展。然而，软件工程专用情感检测工具在跨平台设置中的性能下降，说明我们需要进一步研究这一领域，以提升其在跨平台设置中的表现。

具体而言，我们的研究发现可以为从事这一领域工作的软件工程研究人员提供以下帮助。

(1) 需要制定标准指南来创建情感极性基准数据集。 在本研究中，我们使用了与 Novielli 等人相同的三个极性基准数据集（即数据集）。 [38, 39] 他们在跨平台设置下对浅层学习情感检测工具进行研究时使用了这三个数据集。我们决定使用相同的三个基准数据集，以便将本研究的发现与 Novielli 等人的研究结果进行比较。 [38, 39]除 Novielli 等人的研究外， [38, 39] 迄今为止，这三个数据集还被用于软件工程领域的多项研究和已发表论文中 [6, 7, 40]。我们对跨平台设置下表现最佳的两种深度学习情感检测工具所误分类案例中的错误类别进行了人工分析和比较，结果表明，人工标注中的主观性是这两种工具发生误分类的主要原因。这是因为在同一数据集中，人工编码员对相似句子的标注不一致。这意味着人工编码员没有得到具体的标注指南。不同数据集中的相似句子被人工编码员标注为不同类别时，也会出现标注中的主观性。这意味着两个数据集是依据两套不同的情感极性标注指南创建的。因此，某个工具在一个数据集上训练后，依据训练实例对相似句子进行标注时，可能会在另一个数据集中出错。因此，未来软件工程领域的情感检测研究应着重于实现标注指南的标准化。基于我们的发现，我们还呼吁修复当前的数据集，使其都遵循相似的标注指南。

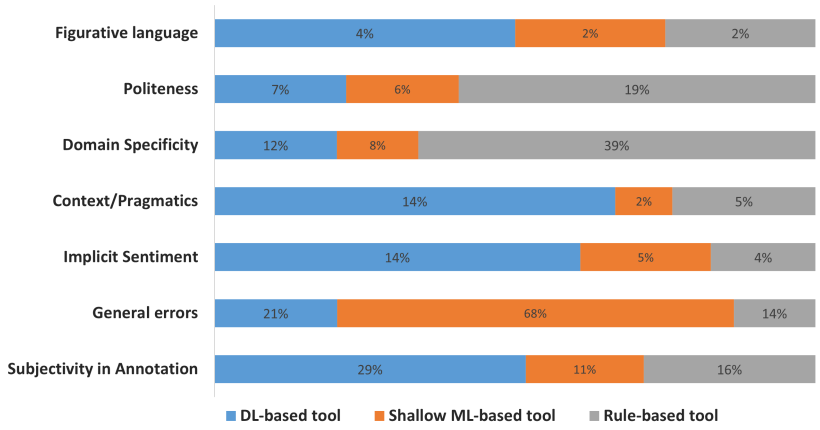


图10：基于深度学习、基于浅层机器学习和基于规则的情感检测工具的误预测类别比较。

(2) 找到合适的集成工具，以发挥不同情感检测工具的优势。我们在研究问题4中的错误分析表明，错误类别的分布会因情感检测工具的类型（即深度学习、浅层学习或基于规则）而异。在图10中，我们汇总了研究问题4中三类工具的错误类别分布（以百分比表示）。

这些工具在我们研究的六种跨平台设置中的表现。该汇总基于图1、2和3中的三个饼图数据生成。深度学习情感检测工具（BERT4SentiSE、SentiMoji）的错误大多源于人工标注中的主观性；浅层学习情感检测工具（Senti4SD、SentiCR）的错误大多属于一般性错误；而基于规则的情感检测工具（SentistrengthSE）最常见的错误则是缺乏对特定领域语境的理解。直观而言，这三类工具因此可以针对给定数据集相互补充。这正是我们开展研究问题5的动机：我们尝试了七种基于多数投票原则的工具集成。然而，在六种跨平台设置中，基于多数投票的集成工具仅在三种设置下优于表现最佳的独立情感检测工具，而且优势也很小。因此，未来的相关研究可以探讨优于简单多数投票的集成技术。

(3) 对“更多”迁移学习的需求。从我们的实验中可以看出，预训练深度学习情感检测工具 BERT4SentiSE 的表现优于基于浅层机器学习的工具 SentiCR 和 Senti4SD。基于规则的工具 SentiStrengthSE 则略胜一筹。BERT4SentiSE采用BERT架构，该架构基于迁移学习原理。这意味着，BERT4SentiSE得益于BERT从大型语料库数据（例如Wikipedia、英文小说等）中学习到的领域无关词嵌入。事实上，迁移学习已展现出良好的前景，能够使监督分类模型跨多个领域进行泛化，并使这些模型只需少量来自目标领域的训练数据，就能更有效地适用于该领域。事实上，在第4.3节中，我们发现，只需使用某个数据集全部训练数据的约30%，BERT4SentiSE便可在该数据集上达到最佳性能水平。迁移学习是自然语言处理领域中不断演进的范式，新技术正在迅速发展（例如RoBERTa[29], XLNet[69],多语言LASER模型等）。因此，软件工程领域对情感检测工具的研究可以通过开发更多基于迁移学习技术的软件工程情感检测工具而受益。

- **软件工程管理者**。软件工程团队和组织中的管理者需要评估同事的生产力和心理健康。对软件工程代码库中员工表达的情感进行自动分析，有助于管理者实时、全面地了解员工的情况，并掌握整个组织的状况。本研究的发现可以为软件工程管理者使用软件工程专用情感检测工具进行此类分析提供指导，具体如下。

(1) 在跨平台设置下使用情感分析评估开发者生产力。团队的生产力可能取决于开发者在各类开发活动中的情感 [42, 21, 45, 34]。因此，组织中的软件工程管理者可以利用组织各类软件工程代码库中的情感极性分析，了解组织的心理健康、生产力和团队凝聚力。然而，正如本研究所见，软件工程数据集可能存在差异，在一个数据集上训练的情感检测工具在另一个数据集上测试时，性能会下降。因此，软件工程管理者应针对特定平台训练和测试其软件工程情感检测工具，而不是在跨平台设置下应用该工具。否则，得出的结果可能无法准确反映整个组织的情感状况。

(2) 多数投票与独立工具之间的权衡。 我们在研究问题4中的错误分析以及图 10 表明，情感检测工具在跨平台设置中的错误各不相同

具体取决于工具类型（例如，深度学习、浅层学习或基于规则）。然而，在50%的跨平台设置中，基于多数投票的集成无法胜过表现最佳的独立情感检测工具。因此，软件工程管理者或许应避免使用基于多数投票的集成工具，从软件工程代码库中挖掘情感极性。

6 效度威胁

在本节中，我们遵循实证研究指南，讨论本研究的效度威胁 [68]。

构念效度 威胁涉及研究方法中可能存在的错误。本研究使用了 Novielli 等人以及其他类似研究所用的同三个基准数据集[38] [13, 26]。在对深度学习情感检测工具进行错误分析时，我们遵循了相关研究所采用的类似指南 [38, 62]。我们还观察到了 Novielli 等人先前报告过的相同错误类别 [38]。另一个问题是，报告的性能指标可能不够规范。为减轻这一威胁，我们遵循该领域的标准报告惯例，同时报告宏平均和微平均分数以及科恩卡帕一致性值。

内部效度 威胁涉及作者在开展研究时可能存在的偏见。这指的是由于未对情感检测工具使用最优配置参数而可能导致的错误。对于研究中的每种工具，我们都使用作者提供的代码来运行工具并微调其性能（例如，调整超参数）。在平台内设置中，我们采用了10折交叉验证设置，并对每种工具进行训练和测试各十次，以减轻模型预测的方差。为减少编码错误，我们使用了广泛应用的开源库（例如，SciKit-learn [48]）来计算本研究中报告的指标。另一种偏见可能来自人工对情感检测工具误分类结果进行归类时产生的偏差。两位作者共同讨论并参与错误分类过程，以减少人为偏差。我们还在复现实验包中分享了人工分类结果。

外部效度威胁指研究结果的可推广性。在本研究中，我们使用了三个数据集，它们代表了软件工程领域情感检测的所有可用数据集，并曾用于类似研究。这些数据集包含约 1.7 万条数据，来自三个广受欢迎、使用广泛的协作平台：问答平台（Stack Overflow）、协作开发平台（GitHub）和问题跟踪平台（Jira）。将结果推广到另一个新的软件工程平台，可能会为这些工具的跨平台应用带来新的见解。总之，与其他工具相比，本研究表明，预训练模型深度学习工具在情感检测任务中表现良好，因为它们能够有效理解并泛化软件工程领域的语境。

7 相关工作

本文的研究动机源自 Novielli 等人近期开展的研究 [38, 39]。Novielli 等人 [38, 39] 研究了浅层学习情感检测

工具在跨平台设置中的有效性。我们通过分析三种最新的软件工程深度学习情感检测工具在跨平台设置中的有效性，扩展了软件工程领域的这部分知识。与 Novielli 等人 [38, 39], 不同，我们还考察了在跨平台设置中集成不同软件工程情感检测工具的益处。在第 1节中，我们总结指出，鉴于以下原因，有必要对深度学习情感检测工具在软件工程跨平台设置中的有效性开展深入且系统的研究：1. 性能优势的一致性。我们希望了解，在跨平台设置中，深度学习情感检测工具是否也像在平台内设置中那样优于浅层学习情感检测工具。2. 模型的预训练设置。我们假设，BERT4SentiSE 等深度学习模型的预训练设置表明，该模型在跨平台设置中也能像在平台内设置中一样优于所有其他模型。我们的总体目标是找到一个能够在跨平台和平台内设置中都提供可靠性能的模型，这样我们就不必针对每个平台训练模型，也能获得可靠的情感标签。Novielli 等人 [38] 的研究发现也进一步促成了这一目标：浅层学习情感检测工具在跨平台设置中的性能会大幅下降，因此其在跨平台设置中的可靠性不如在平台内设置中的可靠性。因此，与 Novielli 等人 [38], 类似，我们使用相同的三个基准数据集。此外，我们选取了 Novielli 等人 [38]研究中表现最佳的浅层学习情感检测工具和无监督情感检测工具，与我们的三种软件工程深度学习情感检测工具进行比较。通过回答四个研究问题，我们得出的发现为 Novielli 等人 [38] 的研究结果提供了新的见解，具体如下。

– 与浅层学习情感检测工具相比，BERT4SentiSE 等深度学习情感检测工具在跨平台设置中的性能下降幅度略小。因此，当无法在新平台上训练工具时，深度学习情感检测工具在跨平台设置中可能比浅层学习情感检测工具更可靠。

– 总体而言，与浅层学习情感检测工具相比，深度学习情感检测工具 BERT4SentiSE 在六种跨平台设置中的五种设置下表现最佳。然而，在六种跨平台设置中，该工具仅在两种设置下表现次于无监督情感检测工具 SentiStrengthSE。这一发现与 Novielli 等人 [38],的结论一致，他们也报告称 SentiStrengthSE 优于浅层学习情感检测工具。不过，在他们的研究中，SentiStrengthSE 在全部六种跨平台设置中都优于浅层学习情感检测工具。

– 我们对工具误分类案例的标注发现，深度学习工具的性能下降主要受数据集标注中的主观性影响，也就是说，问题在于创建数据集时情感极性的标注不一致。这一发现不同于 Novielli 等人对浅层学习情感检测工具的误分类分析。Novielli 等人 [38], 报告称，浅层学习情感检测工具的大多数误分类案例都与一般性错误有关，例如工具无法理解数据集中的特定句法元素（如表情符号）。这一发现促使我们探究深度学习情感检测工具能否在集成设置中补充浅层学习情感检测工具，例如，对所有可用工具给出的某个句子的情感极性进行多数投票，其表现是否优于单个工具。

– 在研究问题4中，我们报告称，在跨平台设置下，简单的基于多数投票的工具集成，其表现并不优于单个工具。Novielli 等人 [38] 并未研究跨平台设置下单个情感检测工具集成的有效性。

– 与 Novielli 等人不同， [38]，我们还检验了两个数据集之间的相似性是否可用于理解软件工程情感检测工具在跨平台设置下何时／如何会更有用或更无用。我们在第 4.1 节对此进行了系统分析，并给出了详细结果。我们发现，软件工程跨平台数据集之间的相似度确实很高，但情感检测工具在跨平台设置下仍会出现性能下降。为了进一步了解性能下降是否与数据集之间的相似性相关，我们在第 4.2 节进一步检验了三个软件工程工具与一个非软件工程数据集之间的相似性，发现软件工程数据集之间确实比它们与非软件工程数据集更为相似。这些发现使我们得出结论：在情感极性的标注方式上，各软件工程数据集彼此并不一致，进而导致情感检测工具在跨平台设置下受到混淆（因为这些工具无法可靠地将从一个数据集中获得的知识应用到另一个数据集中）。

– 在第 4.4 节中，我们展示了在跨平台设置下，情感检测工具在一个数据集（例如 Jira）上训练时可能比在其他数据集上训练时出现更多问题。在第 4.5 节中，我们展示了具体证据，说明由于文本中存在礼貌表达，情感检测标签的不一致性可能会受到影响（例如，“谢谢”在某些数据集中被标记为中性，而在其他数据集中则被标记为正向，等等）。未来的研究可以进一步调查这些发现，以确定如何修复或改进此类软件工程数据集，从而使情感检测工具能够更好地应用于跨平台设置。

软件工程研究中有关情感分析的其他相关工作大致可以分为两类：**研究** 不同软件工程代码库中的情感，以及针对不同软件工程数据集开发特定于软件工程的情感检测**工具**。

研究。大量研究致力于分析情感在软件工程代码库和开发场景中的普遍性及影响，例如 IT工单 [11]，分布式协作团队通信 [20]，Twitter 信息流 [19]，提交日志 [52]，GitHub 中与安全相关的讨论 [45]，应用程序接口评审 [63, 59, 61]，和代码评审 [5]。研究还将情感分析与其他技术相结合，以改进领域特定分类，例如对应用程序评审进行分类 [31, 44]。情感分析关注极性，而情绪分析则关注更细粒度的表达，例如愤怒、爱等。情绪分析可用于确定团队凝聚力，例如霸凌者对软件工程团队的影响 [42],VAD（效价、唤醒度、支配度）评分 [67] 与软件工程团队生产力和职业倦怠之间的关系 [34]，以及利用表达出的情绪来确定改进事项的优先级或提高团队意识 [21, 18]。总体而言，情感和情绪分析正日益成为软件工程领域中备受关注的研究方向 [37, 41]

工具。由于其他领域现成的情感检测工具准确度不高 [24, 25]，因此需要软件工程专用情感分析工具。随后，研究人员为软件工程开发了多种情感与情绪检测器以及基准数据集 [23, 22, 26, 63, 8, 1, 36, 43, 13, 23, 9, 32]。总体而言，软件工程领域的情感检测工作可分为三类：基于规则、基于浅层学习和基于深度学习。本文研究了最近推出的三种

跨平台设置下的深度学习情感检测工具，并将其性能与软件工程领域表现最佳的基于浅层学习和基于规则的情感检测工具进行了比较。

在我们最近发表的论文中 [58]，，我们报告了一项关于软件工程情感检测工具集成的实证研究。我们研究了浅层学习和深度学习情感检测工具。对于给定的数据集，我们考察了工具集成是否能够提供更好的性能。在本文中，作为研究问题4的一部分（见第 3.4 节），我们还考察了软件工程情感检测工具集成是否能够提供更好的性能。然而，在本文中，我们考察了这些工具在跨平台设置下的性能，而在已发表的论文中，我们考察的是这些工具在平台内设置下的性能。此外，本文研究的三个深度学习情感检测工具（BERT4SentiSE、RNN4SentiSE 和 SentiMoji）并未在我们之前发表的论文中进行研究 [58]。

8 结论

软件工程平台的多样性使得在所有平台上部署软件工程专用的有监督情感检测工具变得颇具挑战，因为必须在每个平台上分别对其进行训练。鉴于在每个软件工程平台上训练情感检测工具成本高昂，有时甚至无法实现，因此，找到一种在平台内设置和跨平台设置下都能表现良好的情感检测工具是有益的。我们研究了三种近期提出的、面向软件工程的深度学习情感检测工具，这些工具应用于来自三个软件工程平台的三个软件工程基准数据集。我们发现，BERT4SentiSE 在平台内设置下表现最佳，同时在跨平台设置下也是软件工程有监督情感检测工具中表现最佳的工具。在跨平台设置下，基于词典的工具 SentiStrengthSE 的表现略优于 BERT4SentiSE；但在平台内设置下，BERT4SentiSE 的表现以较大幅度超过 SentiStrengthSE。基于预训练深度学习的工具展现出从小规模训练数据集中理解软件工程语境的潜力。然而，不一致的标注以及基准数据集缺乏标准标注指南，正阻碍着该领域未来的改进。未来，我们将重点改进用于情感检测的软件工程数据集（例如，修复极性标注中的不一致问题）。

参考文献

1. T. Ahmed、A. Bosu、A. Iqbal 和 S. Rahimi，SentiCR：一款用于代码审查交互的定制情感分析工具。载于 *Automated Software Engineering* *Proceedings of the 32nd International Conference on* 106 111, 2017 ，第 – 页。

2. T. Ahmed、A. Bosu、A. Iqbal 和 S. Rahimi，SentiCR：一款用于代码审查交互的定制情感分析工具。载于 *Automated Software Engineering (ASE)* *2017 32nd IEEE/ACM International Conference on* 106 111, 2017 ，第 – 页。

3. M. A. A. Alamin 和 G. Uddin。软件开发生命周期各阶段机器学习软件应用的质量保证挑战。载于 *Conference on Autonomous Systems* *IEEE International* 4, 2021 ，第 页。4.

S. Amershi、A. Begel、C. Bird、R. DeLine、H. Gall、E. Kamar、N. Nagappan、B. Nushi 和 T. Zimmermann，机器学习软件工程：一项案例研究，载于 *2019年 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)* 291 300 2019 ，第 – 页。

IEEE，。5. I. E. Asri、N. Kerzazi、G. Uddin、F. Khomh 和 M. J. Idrissi。代码评审中情感的实证研究。*Information and Software Technology*，114:37–54，2019年。

6. E. Biswas、M. E. Karabulut、L. Pollock 和 K. Vijay-Shanker。利用 BERT 在软件工程领域实现可靠的情感分析。载于 *IEEE International Conference on Software Maintenance and Evolution* 162 173, 2020 ，第 – 页。7. E. Biswas、K. Vijay-Shanker 和 L. Pollock。探索词嵌入技术以改进软件工程文本的情感分析。载于 *Proceedings of the 16th International Conference on Mining Software Repositories* 68 78, 2019 ，第 – 页。8. F. Calefato、F. Lanubile、F. Maiorano 和 N. Novielli。软件开发的情感极性检测。 *Empirical Software Engineering*，23(3):1352–1382，2018年。9. F. Calefato、F. Lanubile 和 N. Novielli。EmoTxt：用于从文本中识别情绪的工具包。载于 *Proc. 7th Affective Computing and Intelligent Interaction*，第 2 页，2017年。10. F. Calefato、F. Lanubile、N. Novielli 和 L. Quaranta。Emtk——情绪挖掘工具包。载于 *2019 IEEE/ACM 4th International Workshop on Emotion Awareness in Software Engineering (SEmotion)* 34 37 2019 ，第 – 页。IEEE，。11. C. Castaldi、A. Blaz 和 K. Becker。IT支持工单的情感分析。载于 *IEEE/ACM 13th Working Conference on Mining Software Repositories* 235 246, ，第 – 页。2016
12. N. V. Chawla、K. W. Bowyer、L. O. Hall 和 W. P. Kegelmeyer。SMOTE：合成少数类过采样技术。 *Journal of artificial intelligence research*，16:321–357，2002年。13. Z. Chen、Y. Cao、X. Lu、Q. Mei 和 X. Liu。SentiMoji：一种用于软件工程情感分析的表情符号驱动学习方法。载于 *27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* 841 852, 2019 ，第 – 页。

14. J. Cohen。加权卡帕系数：为尺度化分歧或部分得分提供的名义尺度一致性。 *Psychological bulletin*，70(4):213，1968年。15. J. Devlin、M.-W. Chang、K. Lee 和 K. Toutanova。BERT：用于语言理解的深度双向 Transformer 预训练。技术报告， https://arxiv.org/abs/1810.04805 2018 ，。16. B. Felbo、A. Mislove、A. Søgaard、I. Rahwan 和 S. Lehmann。利用数百万次表情符号出现记录，学习跨领域表示，以检测情感、情绪和讽刺。 *arXiv preprint arXiv:1708.00524*，2017年。17. B. Fuglede 和 F. Topsoe。詹森-香农散度与希尔伯特空间嵌入。载于 *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.*，第 31 页。IEEE，2004年。18. D. Gachechiladze、F. Lanubile、N. Novielli 和 A. Serebrenik。协作式软件开发中的情感及其指向。载于 *39th International Conference on Software Engineering: New Ideas and Emerging Results Track* 11 14, 2017 ，第 – 页。

19. E. Guzman、R. Alkadhi、和 N. Seyf。大海捞针：Twitter 用户如何谈论软件？载于 *24th IEEE International Requirements Engineering Conference*，第 96–105 页，2016年。20. E. Guzman、D. Az´ocar 和 Y. Li。GitHub 提交评论的情感分析：一项实证研究。载于 *Proceedings of the 11th Working Conference on Mining Software Repositories* 352 355, 2014 ，第 – 页。

21. E. Guzman 和 B. Bruegge。迈向软件开发团队的情绪感知。载于 *Proceedings of the 9th Joint Meeting on Foundations of Software Engineering*，第 671–674 页，2013年。22. M. R. Islam 和 M. F. Zibran。在软件工程中运用自动化情感分析。载于 *Proc. 14th International Conference on Mining Software Repositories*，第 203–214 页，2017年。23. M. R. Islam 和 M. F. Zibran。Deva：在软件工程文本的效价—唤醒度空间中感知情绪。载于 *33rd Annual ACM Symposium on Applied Computing*，第 1536–1543 页，2018年。24. R. Jongeling、S. Datta 和 A. Serebrenik。选择你的武器：软件工程研究中的情感分析工具。载于 *Proceedings of the 31st International Conference on Software Maintenance and Evolution* 2015 ，。25. R. Jongeling、P. Sarkar、S. Datta 和 A. Serebrenik。将情感分析工具用于软件工程研究时的负面结果。 *Journal Empirical Software Engineering* 22 5 2543 2584, 2017 ，( ): –。26. B. Lin、F. Zampetti、G. Bavota、M. D. Penta 和 M. Lanza。问答网站中意见的基于模式的挖掘。载于 *Proc. 41st International Conference on Software Engineering*，第 548–559 页，2019年。

27. B. Lin, F. Zampetti, G. Bavota, M. D. Penta, M. Lanza, 和 R. Oliveto。软件工程中的情感分析：我们能走多远? 载于 *Proc. 40th International Conference on* ，页码 –。28. *Software Engineering* 94 104, 2018
B. Liu。 *Sentiment Analysis and Subjectivity*。CRC出版社，泰勒与弗朗西斯集团，佛罗里达州博卡拉顿，第2版，2010年。29. Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, 和 V. Stoyanov。RoBERTa：一种经过稳健优化的 BERT 预训练方法。 *arXiv preprint* *arXiv:1907.11692* 2019 ，。30. X. Lu, Y.
Cao, Z. Chen, 和 X. Liu。初探 GitHub 上表情符号的使用：一项实证研究。 *arXiv preprint arXiv:1812.04863*, 2018年。31. W. Maalej, Z. Kurtanovi´c, H. Nabil, 和 C. Stanik。关于应用评论的自动分类。载于 *International Requirements Engineering Conference*，第311–331页，2016年。32. R. Maipradit, H. Hata, 和 K. Matsumoto。使用 n元语法逆文档频率和自动化机器学习进行情感分类。 *IEEE Software*，36(5):65–70，2019年。33. C. D. Manning, P. Raghavan, 和 H. Sch¨utze。 *An Introduction to Information Retrieval*。剑桥大学出版社，2009年。34. M. M˙antyl˙a, B. Adams, G. Destefanis, D. Graziotin, 和 M. Ortu。挖掘效价、唤醒度和支配度——检测职业倦怠和生产力的可能性? 载于 *Proceedings of the 13th Working Conference on Mining Software Repositories* 247 258, 2016 ，页码 –。35. P. E.
McKnight 和 J. Najab。曼-惠特尼 U 检验。 *The Corsini encyclopedia of psychology* 1 1, 2010 ，页码 –。36. A.
Murgia, P. Tourani, B. Adams, 和 M. Ortu。开发人员会感受到情绪吗? 对软件制品中情绪的探索性分析。载于 *Proceedings of the 11th Working Conference on Mining Software Repositories* 262 271, 2014 ，页码 –。37. N.
Novielli, A. Begel, 和 W. Maalej。软件工程中情感意识特刊导言。 *Journal of Systems and Software*，148:180–182，2019年。38. N. Novielli, F. Calefato, D. Dongiovanni, D. Girardi, 和 F. Lanubile。我们能否在跨平台设置中使用软件工程专用情感分析工具? 载于 *17th International Conference on Mining Software Repositories* 158 168, 2020 ，页码 –。39. N.
Novielli, F. Calefato, F. Lanubile, 和 A. Serebrenik。对现成软件工程专用情感分析工具的评估：一项扩展复现研究。 *Empirical Software Engineering* 26 2 2021 ，( )，。40. N.
Novielli, D. Girardi, 和 F. Lanubile。一项针对软件工程研究中情感分析的基准研究。载于 *Proceedings of the 15th International Conference on Mining Software Repositories* 364 375, 2018 ，页码 –。41. N.
Novielli 和 A. Serebrenik。软件工程中的情感与情绪。 *IEEE Software* 36 5 6 23, 2019 ，( ): –。42. M.
Ortu, B. Adams, G. Destefanis, P. Tourani, M. Marchesi, 和 R. Tonelli。霸凌者的生产力更高吗? 情感效能与问题修复时间的实证研究。载于 *Proceedings of the 12th Working Conference on Mining Software Repositories* 303 313, 2015 ，页码 –。43. M.
Ortu, A. Murgia, G. Destefanis, P. Tourani, R. Tonelli, M. L. Marchesi, 和 B. Adams。Jira 中软件开发人员的情感面向。载于 *Mining Software Repositories* 480 483, 2016 *15th International Conference on* ，页码 –。
44. S. Panichella, A. D. Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, 和 H. C. Gall。如何改进我的应用? 对用户评论进行分类，以支持软件维护与演进。载于 *IEEE International Conf. on Software Maintenance and Evolution*，第281–290页，2015年。45. D. Pletea, B. Vasilescu, 和 A. Serebrenik。安全与情绪：对 GitHub 安全讨论进行情感分析。载于 *Proceedings of the 11th Working Conference on Mining Software Repositories* 348 351, 2014 ，页码 –。46. J.
Romano, J. D. Kromrey, J. Coraggio, J. Skowronek, 和 L. Devine。探索评估 NSSE 及其他调查中群体差异的方法：t检验和 Cohen's d 指数是否为最恰当的选择? 载于 *Annual meeting of the Southern Association for Institutional Research* 51, 2006 ，第 页。47. S.
Ruder, M. E. Peters, S. Swayamdipta, 和 T. Wolf。自然语言处理中的迁移学习。载于 *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials* 15 18, 2019 ，页码 –。48.
SciKit-learn。 *Machine Learning in Python*。 http://scikit-learn.org/stable/。49. F. Sebastiani。自动文本分类中的机器学习。 *Journal of ACM Computing Surveys* 34 1 1 47, 2002 ，( ): –。

50. P. Shaver, J. Schwartz, D. Kirson, 和 C. O'Connor。情绪知识：进一步探索原型方法。 *Journal of personality and social psychology* 52 6 1061, , ( ):。51. P. 1987
Shaver, J. Schwartz, D. Kirson, 和 C. O'Connor。情绪知识：进一步探索原型方法。 *Journal of Personality and Social Psychology*，52(6):1061–1086，1987年。52. V. Sinha, A. Lazar, 和 B. Sharif。分析提交日志中的开发者情感。载于 *13th International Conference on Mining Software Repositories* 281 290, 2016 ，页码 –。53. M. Sokolova 和 G. Lapalme。对分类任务性能度量指标的系统分析。 *Information processing & management*，45(4):427–437，2009年。54. M. Stone。统计预测的交叉验证选择与评估。*royal statistical society: Series B (Methodological)* 36 2 111 133, 1974 *Journal of the* ，( ): –。55. C. Sun, X. Qiu, Y. Xu, 和 X. Huang。如何微调 BERT 以进行文本分类? 载于 *China National Conference on Chinese Computational Linguistics*，第194–206页。施普林格，2019年。56. M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, 和 A. Kappas。非正式文本中的短文本情感强度检测。 *Journal of the American Society for Information Science and Technology* 61 12 2544 2558, 2010 ，( ): –。57. G. Uddin, O. Baysal, L. Guerrouj, 和 F. Khomh。了解开发人员如何以及为何寻找和分析 API 相关观点。 *IEEE Transactions on Software Engineering*，2019年。58. G. Uddin, Y.-G. Gu´eh´eneuc, F. Khomh, 和 C. K. Roy。一项关于独立情感检测工具集成对软件工程数据集有效性的实证研究。 *ACM Transactions on Software Engineering and Methodology (TOSEM)*，第38页，2022年。59. G. Uddin 和 F. Khomh。API评审的自动摘要生成。载于 *The 32nd IEEE/ACM International Conference on Automated Software Engineering*，第159–170页，2017年。60. G. Uddin 和 F. Khomh。挖掘 API 评审中的 API 方面。技术报告，蒙特利尔理工学院，2017年。61. G. Uddin 和 F. Khomh。Opiner：API评审搜索与摘要引擎。载于 *Proc. 32nd IEEE/ACM International Conference on Automated Software Engineering*，第978–983页，2017年。62. G. Uddin 和 F. Khomh。自动挖掘 Stack Overflow 上针对 API 表达的观点。 *IEEE Transactions on Software Engineering*，2019年。63. G. Uddin 和 F. Khomh。自动挖掘 API 评审中的观点。*on Software Engineering* 37, 2019 *IEEE Transactions* ，第 页。64. G. Uddin, F. Khomh, 和 C. K. Roy。从 Stack Overflow 挖掘 API 使用场景。 *Information and Software Technology*，122，2020年。65. G. Uddin, F. Khomh, 和 C. K. Roy。从技术问答网站自动生成 API 使用场景文档。 *ACM Transactions on Software Engineering and Methodology*，30 3( ): 第31号文章，2021年。66. A. J. Viera, J. M. Garrett 等。理解观察者间一致性：卡帕统计量。 *Fam med*，37(5):360–363，2005年。67. A. B. Warriner, V. Kuperman, 和 M. Brysbaert。13,915个英语词元的效价、唤醒度和支配度规范值。 *Behavior Research Methods*，45(4):1191–1207，2013年。68. C. Wohlin, P. Runeson, M. H ¨ ´ *Experimentation in software engineering* 2012
ost, M. C. Ohlsson, B. Regnell, 和 A. Wesslen。. 施普林格科学与商业媒体，。69. Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, 和 Q. V. Le。XLNet：用于语言理解的广义自回归预训练。技术报告，https://arxiv.org/abs/1906.08237，2020年。70. T. Zhang, B. Xu, F. Thung, S. A. Haryono, D. Lo, 和 L. Jiang。软件工程中的情感分析：预训练 Transformer 模型能走多远? 载于 *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)* 70 80 ，页码 –。IEEE，2020年。