从大规模聊天中检测隐藏的功能请求通过深连体网络发送消息

Lin Shi1, §, Mingzhe Xing1,2, §, Mingyang Li1,2, Yawen Wang1,2, Shoubin Li1,2, Qing Wang1,2,3 1 中国科学院软件研究所互联网软件技术实验室,北京,中国科学院大学,中国 2 北京 中国科学院软件研究所计算机科学国家重点实验室,中国北京 3 {shilin,shoubin,wq}@iscas.ac.cn, mingzhe@itechs.iscas.ac.cn, {mingyang,yawen}@nfs.iscas.ac.cn

抽象的

在线聊天越来越受欢迎,并且在软件开发中发挥着越来越重要的作用。在讨论功能时,开发人员可能会向其他开发人员透露他们想要的功能。从大量聊天消息中检索特征请求的自动挖掘技术可以使需求收集过程受益。但是执行这样的技术非常具有挑战性,因为从对话中检测特征请求需要对上下文信息有透彻的理解,并且为学习而标注特征请求对话的成本也非常高。为了弥补这一差距,我们将传统的将单个对话映射到其类别的文本分类任务重新转换为通过结合小样本学习来确定两个对话是否相似的任务。我们提出了一种名为 FRMiner 的新方法,它可以通过深度连体网络从聊天消息中检测特征请求对话。我们设计了一个基于 BiLSTM的对话模型,可以在正向和反向两个方向学习对话的上下文信息。对现实世界项目的评估表明,我们的方法实现了 88.52%、88.50% 和 88.51% 的平均准确率、召回率和 F1-score,这证实了我们的方法可以有效地检测聊天消息中的隐藏特征请求,从而有助于收集综合需求以自动化的方式从人群中。

关键词

特征请求、需求工程、深度学习、连体网络

ACM 参考格式:
Lin Shi1, §, Mingzhe Xing1,2, §, Mingyang Li1,2, Yawen Wang1,2, Shoubin Li1,2, Qing Wang1,2,3. 2020. 通过深连体网络检测来自海量聊天消息的隐藏特征请求。 在第 42 届国际软件工程会议 (ICSE 20) 上,2020 年 5 月 23 日至 29 日,韩国首尔。 ACM,纽约,纽约,美国,13 页。 https://doi.org/10.1145/3377811.3380356

通讯作者。 § 两位作者对这项研究的贡献相同。

ICSE 20,2020 年 5 月 23-29 日,韩国首尔 © 2020 计算机协会。
ACM ISBN 978-1-4503-7121-6/20/05。 . . 15.00 美元 https://doi.org/10.1145/3377811.3380356

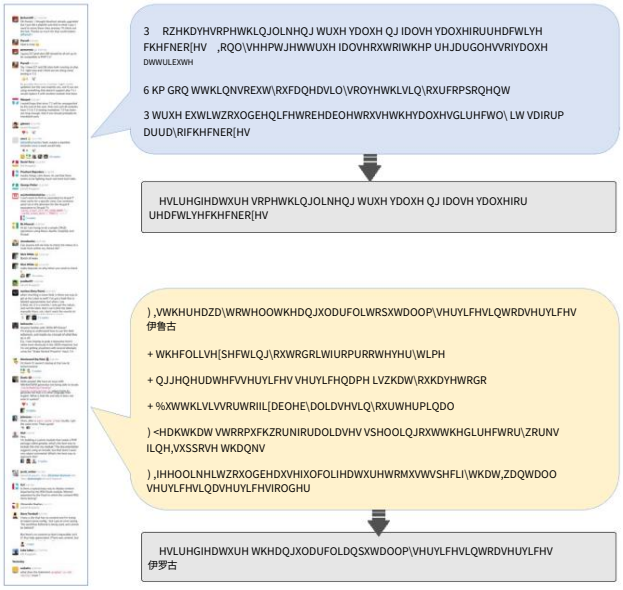

图 1:来自 AngularJS 项目的示例聊天消息,其中对所需功能的请求被隐藏在大量聊天历史中。

1 简介

最近的研究报告称,在线聊天的使用越来越受欢迎,并且在软件开发中发挥着越来越重要的作用,在某些情况下已经取代了电子邮件[35,56,57]。

开发人员正在转向公共工作场所聊天平台,例如Slack、IRC、HipChat、Gitter 和 Freenode,以分享意见和有趣的见解,讨论如何解决缺陷以及未来要实现哪些功能 [9]。

尽管开发人员在与其他开发人员交流时会透露他们想要的功能,但在线聊天的开放性和拥挤性使得这些功能请求对话很快就会被新传入的消息淹没。通常,在线聊天中讨论的功能请求如果没有记录在案,很可能会被隐藏和忽略。以 AngularJS项目的聊天消息为例（图 1）,开发者 P 和 F在在线聊天中发布了他们的问题。一开始,他们的意图是寻求其他开发人员的帮助,以寻求可行的问题解决方案。在与其他开发人员聊天后,他们意识到现有系统无法按照他们想要的方式运行。然后

他们的意图从寻求解决方案转变为要求功能。 P 请求“类似 ng-true-value/ng-false-value
对于反应性复选框”,并且 F 要求“角度 cli 可以放置
我所有的服务都放入服务文件夹”。在这项工作中,我们考虑包含对新功能/增强功能的请求的对话
作为特征请求对话。在实践中,发布团队监控
多种沟通渠道,拥有多种来源
可能与下一个版本相关的信息[51, 61]。如果
发布团队可以确认那些隐藏的功能请求
从聊天消息中,他们的下一个发布计划可能有机会通过考虑最大化利益相关者的满意度
更多功能请求 [50]。

从大量聊天消息中检索有价值信息的自动化挖掘技术对于收集信息来说是非常必要的
来自大量用户的综合功能请求,这
有助于需求获取和发布计划,以及
反过来,促进软件开发的成功[6, 21, 25]。
尽管聊天消息可能很大并且随着时间的推移嵌入功能请求,但挖掘起来非常具有挑战性
由于以下障碍,大量聊天消息。

对话式分析。分析聊天消息中的对话
与常规文本挖掘任务的不同之处在于它需要考虑
理解一个句子时对话范围内的上下文信息。因此,现有的研究

逐句特征请求检测[13, 25, 55]不能直接用于此任务。例如,句子“我们需要
添加垂直导航栏选项”被归类为功能请求
句法技巧。但是当发布在在线聊天中时,后续对话指出现有功能可以实现

以另一种方式提出该请求。此外,在句子方面
由于大量跑题,检测结果会不准确
句子将被识别为聊天消息中的功能请求,
例如,“我真的需要恢复我的编程技能。”,“我会
想喝点咖啡和饼干。”

极其昂贵的注释。聊天消息通常很大。寻找特征请求之间的对话

海量的聊天信息就像大海捞针。
注释来自聊天的功能请求非常昂贵
由于语料库容量大,信息比例低
真实数据。只有少数带标签的聊天消息被归类为功能请求类型。如何最大限度地利用
在少数标记数据中准确分类未标记的聊天
消息成为一个关键问题。

纠缠和嘈杂的数据。聊天消息通常量大,包含涵盖范围广泛的非正式对话
的主题。两个或多个开发者同步交互
彼此的话语在很大程度上纠缠在聊天中
消息。此外,聊天消息中存在重复和离题消息等嘈杂的话语,不提供

任何有价值的信息。纠缠和嘈杂的数据构成了
难以分析和解释交际对话。

在这项工作中,我们迈出了对话式技术的第一步
旨在自动检测发布的隐藏功能请求
在聊天消息中。我们提出了一种新的方法,命名为 FRMiner,
它可以从聊天消息中检测特征请求对话
通过深连体网络。为了更好地理解上下文
对话范围内的信息,我们首先构建一个上下文感知
基于双向 LSTM (BiLSTM) 结构的对话模型
可以深入学习对话的上下文信息
正向和反向。受少数人学习的启发
旨在通过利用不足的标记资源来构建性能预测模型的技术,我们重铸了传统文本
将单个对话映射到其类的分类任务进入任务
确定两个对话属于同一类还是不同类。因此,我们将上下文感知对话模型与
连体网络学习一对对话之间的相似性
而不是特定类的模式。预测结果
可以根据相似度推断特征请求对话
预测和观察到的其伙伴对话的类别。至
评估提出的方法,我们注释了 1,035 个对话
来自三个流行的开源项目。实验结果
表明我们的方法明显优于两个句子分类器和四种传统的文本分类方法

平均准确率、召回率和 F1 分数分别为 88.52%、88.50% 和 88.51%。结果证实,我们的方法可以有效地从聊天消息中检测隐藏的特征请求,从而可以促进
从大量的综合需求中收集
用户以自动化的方式。

本文的主要贡献如下。

- ·我们率先推广检测隐藏功能请求
  来自海量聊天消息,可以全面受益
  需求收集。
- ·我们引入了一种解决方案,通过结合 Siamese Network,可以有效地基于有限的标记数据预测特征请求对话,这显着减轻了
  注释监督数据的负担。
- ·我们评估我们对三个活跃的开源项目的方法,
  实证比较表明,所提出的方法优于现有研究和四种文本分类方法。
- ·可公开访问的数据集和源代码1 ,以促进
  复制我们的研究及其在其他情况下的应用。

2 背景

本节介绍与这项研究相关的三个关键技术:TextCNN、BiLSTM 和少样本学习技术。我们
将它们包括在这里,因为我们的工作基于这些技术。

2.1 文本CNN

聊天消息中的对话是记录文本语句的形式
按照社区讨论的时间顺序
在线交流期间的开发人员。建模句子表示是高级对话分析的基础。在这个
在论文中,我们使用 TextCNN [30] 来表示句子,它有一个
优于在标记数据不足的情况下学习,因为它采用
简洁的网络结构和少量的参数。

TextCNN 是一种经典的句子建模方法
使用浅层卷积神经网络 (CNN) [32]进行建模
句子表示。 CNN是一种深度学习模型
已广泛应用于计算机视觉。它使用几个卷积核来捕获局部信息作为感受野,

1https://github.com/FRMiner/FRMiner

然后用这些局部信息产生全局表示。类似地,在自然语言处理 (NLP) 中,CNN可以聚合 n-gram 信息并对句子表示进行建模。 TextCNN 将预训练或随机生成的词嵌入作为输入。其输出的维度取决于卷积核的数量和大小。一个n长的句子可以表示为一个形状为n×d的矩阵,其中d是词嵌入的维度。每个内核w ∈ Rkd卷积核的大小,应用于k个单词的窗口,以映射成一个新的一维向量。令Xi:i+k表示原句中k-gram单词的拼接,然后对其进行卷积操作。卷积层的输出可以计算为oi = f (w · Xi:i+k + b)其中b是偏置项， f是激活函数。给定一个句子的长度l和卷积核大小k,我们可以得到句子的表示,其大小为l-k + 1。卷积层后面是一个最大轮询层,它可以捕获关键信息最高值。其中k是

为了获得由不同尺度的局部信息集成的更充分的语义信息,将多个不同大小的卷积核应用于句子。因此,对于一个句子,给定n × m 个卷积核,其中n是不同大小的核数， m是每个大小的核数,我们可以得到大小为n × m 的句子表示,即将不同尺度的局部信息编码为句子的全局表示。

2.2 双向 LSTM

分析聊天消息中的对话是一项高级文本挖掘任务,因为它需要在理解单个句子时考虑对话范围内的上下文信息。

在本文中,我们利用双向长短期记忆网络（BiLSTM）,将对话的句子视为顺序项,以捕获上下文信息,其中句子的表示由 TextCNN 嵌入。 BiLSTM由 Graves 等人提出。[20]为序列学习任务学习双向信息。 BiLSTM 堆叠两个具有相反方向的标准长短期记忆网络 (LSTM) [23]层,以分别学习单向表示。

然后它将前向和后向表示组合为双向嵌入。长短期记忆网络(LSTM) 是Hochreiter 等人提出的基于门机制的优化循环神经网络 (RNN) 结构。

[23]。 LSTM 利用门机制来过滤关键信息并将它们传递给长序列。一个 LSTM 单元由输入门、遗忘门、单元状态和输出门组成。 LSTM 单元门的输出可以指定如下：

$$\begin{pmatrix} 它 \\ ft \\ c\sim t \\ 过 \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ 谭 \\ \sigma \end{pmatrix} W \begin{pmatrix} xt \\ ht-1 \end{pmatrix} + b$$

$$ct = c\sim t \quad 它 + ct-1英尺$$

$$ht = ot \tanh(ct)$$

其中xt是输入序列中的第t个标记； W是LSTM单元的权重矩阵； b是偏置项； σ表示逻辑 sigmoid 激活函数， tanh表示双曲正切激活函数;表示逐元素乘法。

因此,BiLSTM 的最终表示可以形成为

$h = [\overrightarrow{h} \oplus \overleftarrow{h}]$,其中$\overrightarrow{h}$和$\overleftarrow{h}$分别表示两个 LSTM层的输出,⊕ 是连接操作。

2.3 小样本学习深度学习在计算机

视觉和自然语言处理领域都取得了显着的成功。但它严重依赖足够的训练数据量,当标记资源不足时很难表现良好。聊天消息中的自动化挖掘也面临标签资源不足的问题。在在线聊天中,庞大的开发者社区会在短时间内创建大量讨论。注释大量对话数据以学习有效模型非常耗时,因为它们不仅很长,而且需要领域知识才能彻底理解。提出了很少的学习方法来克服这些限制[64]。少样本学习方法可以分为以下三类[10]。基于模型的方法旨在通过模型设计学习从少量标记数据到分类的投影仪。基于优化器的方法调整传统的梯度下降优化器方法以拟合数据。基于度量的方法通过学习相似度度量函数对样本进行分类。

在本文中,我们利用了一种名为 Siamese network [8] 的基于度量的小样本学习技术,该技术被广泛用于测量文本或图像之间的语义相似性[45]。传统的分类模型试图学习从单个实例到其类的映射,但是当可用的标记资源数据较少时,它们并不总是能很好地工作。不同于传统的

Siamese 网络将成对的实例作为输入,旨在学习确定两个实例属于同一类还是不同类的关键特征。它由两个相同的子组件组成,它们不仅共享模型结构,而且还具有分别编码实例对的参数。

直观地说,我们更容易确定两个对话是否相似,而不是为每个对话给出确切的类。由于连体网络将对作为输入,因此数据集从元素转换为成对,并且可以通过每个突变进行扩充。

3 方法

图 2 展示了我们方法的总体框架。我们通过解开聊天消息中的对话来构建训练数据集。然后我们为每个对话建立一个分层的上下文感知对话模型。 context-ware 对话模型通过 BiLSTM 结构对对话进行编码,该结构使用基于 TextCNN 的句子嵌入作为输入。之后,我们构建了一个具有两个相同的上下文感知对话模型的连体网络。最后,我们根据连体网络产生的概率和配对实例中黄金对话的实际标签来推断预测类别。

3.1 对话解开聊天频道是开发者社区之间的一种同步文本

交流。聊天中的消息形成流信息,对话经常纠缠在一起,例如单个对话线程与其他对话交错。

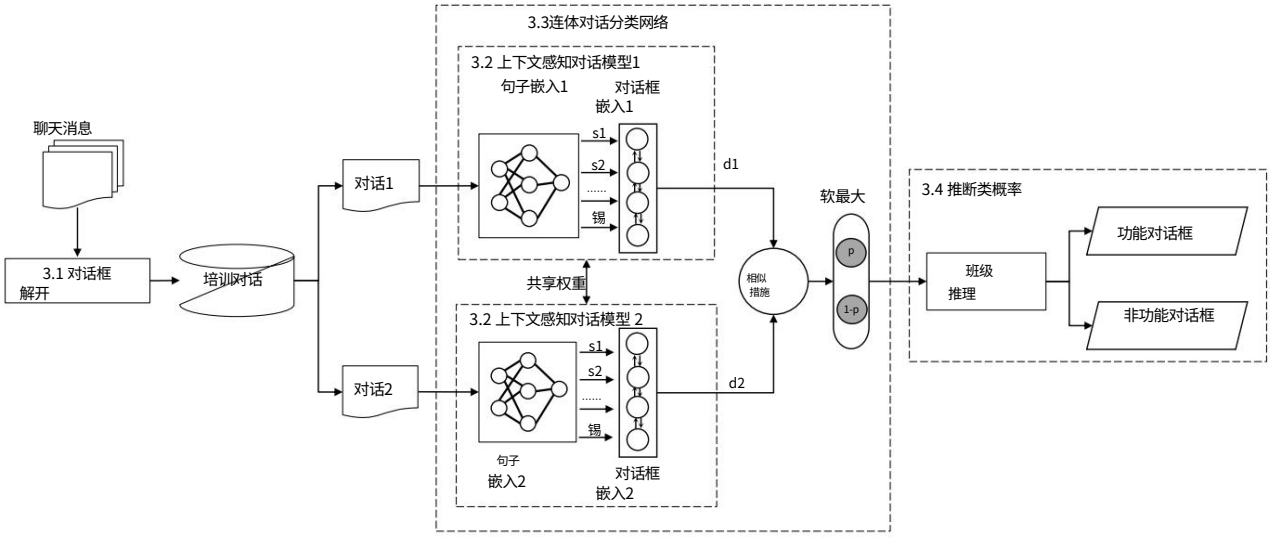

图 2:FRMiner 概述

将聊天消息分成一组不同的对话是一种
任何类型的高级对话分析的必要前提。
我们利用Kummerfeld 等人提出的最先进的对话解开技术。 [33]。他们的模型是
从 77,563 条手动注释的解开的消息中训练
来自在线聊天的对话。它是一个前馈神经网络
具有 2 层、512 维隐藏向量和软符号非线性。模型的输入是一个 77 维向量,其中
每个元素都是从原始元素中提取的数字特征
对话文本,包括与上次聊天的时间间隔
当前用户发布的消息,是否有目标用户在
聊天内容,是否两个聊天文本包含相同的单词等等。

图 3 是解缠结前后的对话演示。该模型可以达到相对较好的性能

准确率 74.9%,召回率 79.7%。

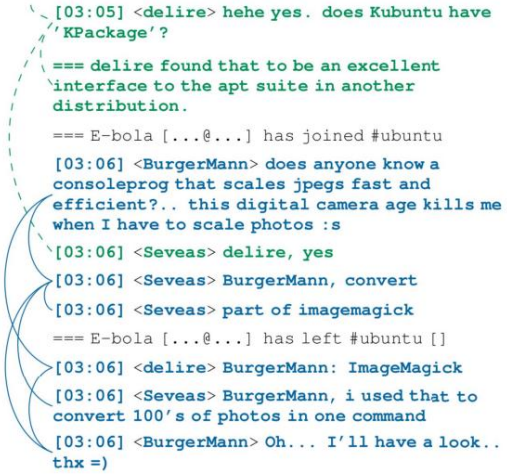

图 3:解开前后的对话示例 [33]。这些不同颜色的曲线代表

解开后不同对话的链接。

3.2 构建上下文感知对话模型

我们设计了一个层次化的上下文对话模型,可以捕获
上下文信息以及每个的语义
对话中的句子。如图 4 所示,上下文感知对话框
模型由四层组成:输入层、句子嵌入层、
对话嵌入层和输出层。

输入层。我们首先将句子标记为标记作为
基本条款。为了获得更好的性能,我们使用了 50维 Glove 词嵌入[49] ,它在 60 亿个上进行了预训练
维基百科和 Gigword 语料库的词作为初始向量
对应的词。此外,受以前作品的启发
[58] [55],我们注意到词性（POS)模式或模板
显然存在于特征请求文本中。直观地说,POS 标签
可以通过引入显式词汇来促进语义理解
信息。因此,我们将 POS 标签信息添加到单词表示中以增强其特征。具体来说,每种类型的 POS 标签
将被初始化为具有均匀分布的随机向量,并且
在训练期间进行优化。因此,每个单词都可以表示
as wi = [wei ⊕ posi],其中wei表示对应的词
embedding 和posi表示嵌入的 POS 标签
单词。

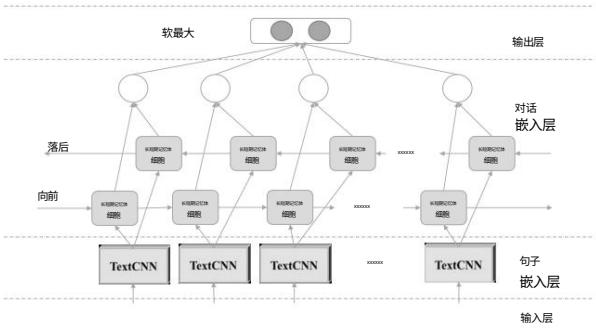

图 4:分层上下文感知对话模型

句子嵌入层。在将原始句子转换为与词嵌入和 POS 标签嵌入堆叠的矩阵后,我们将嵌入矩阵输入 TextCNN 以获得句子表示。 TextCNN表示句子的细节已在 2.1 节中介绍。

对话框嵌入层。在对话嵌入层中,我们使用一系列句子嵌入来表示对话。每个嵌入的句子在根据对话中的序列输入 BiLSTM 编码器时充当一个标记。在对BiLSTM进行编码后,将学习到对话的双向上下文信息。

输出层。在输出层,我们结合了两个方向
表示-→h和←-h由 BiLSTM 编码为对话的输出向量,可以表示为 h = [-→h ⊕ ←-h]。

3.3 构建连体对话分类网络

为了缓解标注数据不足的问题,我们构建了 Siamese 对话分类网络,通过将传统的将单个对话映射到其类的文本分类任务重铸为确定两个对话是否属于相同或不同的班级。为了清楚起见,我们将在本文的其余部分使用特征对话和非特征对话来指代请求特征的对话和不请求特征的对话。

图 2 中“3.3”的虚线框表示了详细的架构。 Siamese 对话分类网络包含两个上下文感知对话模型,它们共享结构和参数,分别将一对对话编码为d1和d2 。我们使用d1和d2的组合形式， [d1 ⊕ d2],作为两个对话之间关系的表示。然后将一对对话之间的关系表示从对话嵌入投影到相似度度量。由于相似性度量的显式方程,例如余弦相似度[24]和欧几里得距离[24],通常用于度量线性空间中向量之间的接近度,因此它们不适用于语义空间中的复杂对话。因此,我们采用在训练神经网络期间学习的相似函数。由于可以获得两个对话的差异或相同的标签,我们可以在神经网络中训练一个相似层。它就像一个黑盒组件。输入是两个带有“相同”或“差异”标签的对话的嵌入表示,输出是它们的相似性。

我们按照以下步骤训练连体对话分类网络。 (1) 我们将数据集随机分为训练数据集和测试数据集。 Train_d 是使用标签为“特征”或“非特征”的对话作为条目的原始数据集。 (2)

通常,标记对话的大小不足以训练有效的基于学习的模型。为了克服这个问题,我们通过从 Train_d 中采样一对带有“相同”或“差异”标签的对话作为 Train_p 的一个条目,将 Train_d 扩充到 Train_p。更具体地说,对于训练数据集中的每个对话,我们从训练数据集 Train_d 中随机选择一个带有“特征”标签的正面合作伙伴和一个带有“非特征”标签的负面合作伙伴。例如,如果两个对话都是特征对话或非特征对话,我们将给该对分配 label- same ,否则为 diff 。由于一正一负采样策略,我们的训练数据可以

自然平衡。此外,假设我们在 Train_d中有 m 个特征对话和n 个非特征对话,我们可以在 Train_p 中将原始数据集扩充为+ m × n的大小。最后,由于每对对话属于“相同”类或“差异”类,相似性度量的输出是一个长度为 2 的向量[score1,score2],表示两个类的分数,其中scorei ∈ R。我们对长度为 2 的向量执行 softmax,可以指定为

$$\text{所以 f tmax(socrei)} = \frac{\text{埃斯科雷}}{\text{escorej j=1}^{2}}$$

然后可以将[score1,score2]归一化为概率 [p, 1－p],其中 p ∈ [0, 1]。

3.4 Infer Class Probability连体对话

分类网络的输出是指示两个对话是相同还是不同的概率能力。

但是我们需要的是指示对话是否是特征对话的概率,因此,我们需要根据概率和配对实例中另一个对话的实际标签来推断对话的标签。例如,我们对< Dialoä1,Dialoä2 > 进行采样,其中Dialoä1是从带有观察到的“非特征对话”标签的训练数据集中采样的黄金对话,而Dialoä2是要预测的未知对话。我们将这对对话输入到连体对话分类网络中,然后做出判断两个对话相同或不同的预测。

假设预测是 diff,那么我们可以推断Dialoä2的类是一个特征对话。如果Dialoä2的实际标签是特征对话,则表明我们的模型做出的这个预测是真阳性。否则,我们会得到假阳性预测。为了获得更可靠的预测结果,我们在预测短语期间采用投票策略。对于每个未知对话,我们通过采样k个不同的黄金对话来构建k个配对实例。在将这些对传递给 FRMiner 之后,我们可以得到l个实例,表明未知的是一个特征对话和 k -l个实例,表明它是一个非特征对话,基于 FRMiner 的预测和

金色对话的标签。如果l大于然后我们为预测对话分配 “特征对话”标签,反之亦然。

3.5 工具实现我们使用 Allennlp [16] 实

现了我们提出的方法 FRMiner,这是一个基于 PyTorch [15] 构建的开源 NLP 库。

实施细节。对于这些超参数,我们使用网格搜索[7]作为参数选择方法以获得最佳性能。词性标签嵌入的维数为 50,与词嵌入相同。然后,我们可以使用s = [w1,w2 ...wn]作为句子的表示,其中wi = [wei ⊕ posi] 。为了获得由不同尺度的局部信息集成的更充分的语义信息,将多个不同大小的卷积核应用于句子。我们设置了 4 种不同的内核大小,分别为 2、3、4、5,每个内核有 25 个特征图。

BiLSTM 的输出维度为 300（每个方向 150）。我们使用线性层作为相似层,将 300 维向量投影到两个值,代表两个类别的概率分数。由于任务可以被视为一个分类问题,我们使用交叉熵作为损失函数。

优化。另外,为了避免过拟合问题,
我们以0.1的丢弃率对输入嵌入应用 dropout [59] ,
这意味着,10% 的神经元细胞将被随机掩盖以减少
每批训练中需要训练的参数。我们
也使用提前停止的策略[52]。如果表现在
测试数据集没有提升 10 个 epoch,训练过程
将被停止。

4 实验设计

4.1 研究问题

我们的评估解决了以下三个研究问题。

RQ1:我们检测隐藏的方法有多有效
功能要求?为了调查我们方法的有效性,
我们对检测特征对话进行 3 折交叉验证
来自三个开源项目的聊天消息。我们也比较
两种分类方法的表现
在线讨论的句子分为功能请求和其他类型。
我们通过预测对话来调整这两种方法来完成我们的任务
包含特征请求句子作为特征对话。除了,
我们检查了四种广泛使用的分类的性能
在有限的标记资源上感知困难的方法
聊天消息中的自动特征请求挖掘。

RQ2:连体网络如何促进特征请求检测?为了检查Siamese Network 引入的性能增强,我们构建了 p-FRMiner,它

是一个普通的 FRMiner,没有包含 Siamese Network 技术。 FRMiner 和 p-FRMiner 之间的详细区别将
在第 4.3 节中描述。然后我们比较性能
由对话实例直接训练的 p-FRMiner,具有
由配对实例训练的 FRMiner 的性能。然后
我们越来越多地将训练对实例的大小扩大到
检查性能增强和之间的关系
数据增强。

RQ3:我们的方法在跨项目验证中是否有效? RQ3 通过以下方式检查我们方法的普遍性
三个开源项目的跨项目验证。我们反复使用两个项目进行训练和保留一个用于测试。

我们还对基线方法进行跨项目验证。

4.2 数据准备

我们的实验数据由 Scrapy [53] 从三个开源项目中抓取:AngularJS [17]、Bootstrap [60]和 Chromium
[18]。我们选择这三个项目的原因如下。第一的,
它们正在积极开发中。第二,大社区
围绕这些项目形成。三、这些项目的开发者
积极使用在线聊天来分享意见,有趣的见解,
并讨论将来要实现的功能。例如,
在过去三年中,平均每人发出 2,823 次话语
AnguluarJS 社区的一周。此外,他们的历史聊天
消息都记录在案并可公开访问[1],其中
为挖掘有价值的信息提供丰富的资源。我们的数据
通过以下步骤收集:

第 1 步:预处理。我们首先对非 ascii 字符进行归一化,例如
表情符号到标准 ascii 字符串。一些低频令牌不能
有助于分类结果,例如 URL、电子邮件地址、
聊天消息中的代码、HTML 标签和版本号。我们更换

表 1:标记对话的统计量

	海量聊天消息			样本		
	时长 406553	#dialog	#sentence	#dialog	#sentence	#FR
AngularJS 2016.5-2019.4 38266 3[illegible]9						36
2014.7-2019.5 10358 锛 2015.5-2019.7 16804			58871	379	2371	76
			118890	340	4465	27
全部的		65428	584314	1035	16056	139

它们带有特定标记 <URL>、<EMAIL>、<HTML>、<CODE> 和
<ID> 分别。我们利用 Spacy [2]将句子标记为
条款。为了减轻词形的影响,我们然后
使用 Spacy 执行词形还原和小写。

第 2 步:对话采样。解压后有
大量识别的聊天对话。为了观察整个对话群体的特征,我们随机抽样

来自三个项目的 400 次对话。然后我们排除了不可读
对话: 1) 用非英语语言编写的对话;
2) 包含过多代码或堆栈跟踪的对话; 3)低质量的对话,例如有很多错别字和语法的对话
错误。 4) 涉及频道机器人的对话。

第 3 步:真实标签。使用标记的对话
作为方法定义和性能的真实数据集
评估。为保证标注结果的正确性,我们
建立了一个由两名高级研究人员组成的检查组
有四名博士候选人。他们都是流利的英语人士,
并在软件开发方面进行了深入的研究工作
开放或一直在积极为开源项目做出贡献。
我们将团队分成两组。每个小组由一名领导者组成
(高级研究员)和两名成员。领导培训成员
在此过程中如何标记和提供咨询。这
成员的标注结果由领导审核
而领导的结果由其他领导审查。我们
仅当对话在各组之间获得完全同意时,才接受并将对话纳入我们的数据集。当一个对话框
收到不同的标签结果,我们与所有人进行了讨论
六人通过投票决定。

我们总共从三个开源项目中收集了 65,428 个对话
项目,并花费 720 工时注释 1,035 (1.6%)
对话。标记对话的详细特征如表 1 所示。最后一列“#FR”表示对话的数量

特色对话。

4.3 实验设置

我们对收集的数据集进行 3 折交叉验证[31]
来自三个开源项目。我们随机划分数据集
分成3部分。我们使用其中 2 个部分进行训练并保留一个部分
用于检测。我们每次保留时重复此过程 3 次
用于测试的不同部分。实验环境是
配备 NVIDIA 1060 GPU、英特尔核心的台式电脑
i7 CPU,16GB RAM,在 Ubuntu 操作系统上运行。

实验 I (RQ1)为了证明我们方法的有效性,
我们选择了两个高级的句子方法和四个文本
分类方法作为基线。有关的详细信息
基线将在 4.4 节中介绍。对于这两个句子

明智的做法,我们使用提供的代码和模型
出版物。对于四种文本分类方法,我们使用
由官方发布的软件包提供的代码[19] [14]。我们申请
随机过采样[36]来解决不平衡数据集。
我们提取词频和逆文档频率

(TFIDF) [27]作为每个对话的特征向量。我们通过网格搜索对四种文本分类方法进行训练和微调超参数,以实现其最佳性能。

实验 II (RQ2)在这个实验中,我们将 FRMiner与 p-FRMiner 进行比较,并研究数据增强 im 是如何提高性能的。请注意,p-FRMiner 与 FR Miner的不同之处在于 p-FRMiner 是单个对话的分类模型,它基于上下文感知对话模型（在第 3.2 节中介绍）,并带有一个附加的分类层。 FRMiner的架构可以通过以下步骤从 p-FRMiner 推导出来： 1)去除 p FRMiner 的 top-top-classification 层； 2) 连接两个共享权重的 p-FRMiner 的输出； 3)添加相似层。首先,我们使用相同大小的数据来训练 FRMiner 和 p-FRMiner,并观察性能提升。然后我们将 FRMiner 的训练数据集扩充 5、10、20和 30 次以研究性能变化。

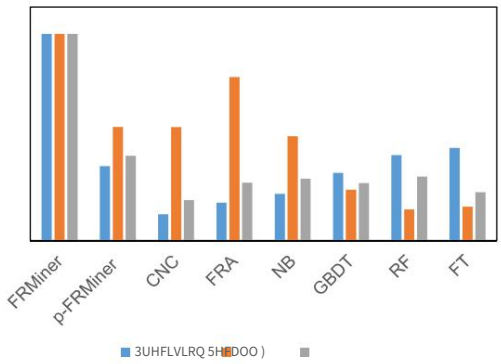

图 5:3 倍验证的平均性能

由于 FRMiner 可以通过应用连体网络来解决不平衡的数据集问题,而 p-FRMiner 不能,因此我们在训练p-FRMiner时通过应用随机过采样[36]来平衡样本。为了确保我们实验的正确性,FRMiner 和 p-FRMiner 使用相同的超参数进行训练,包括每层的维度、网络的深度和学习

速度。

实验 III (RQ3)为了验证我们的方法是否可推广到不合适的项目,我们在两个项目上训练了 FRMiner并在第三个项目上进行评估。我们还使用与实验 II相同的超参数评估跨项目数据集上的其他基线。

4.4 基线

为了展示 FRMiner 的优势,我们将 FRMiner与两种先进的句子方法作为我们的基线进行了比较。

基于 CNN 的分类器 (CNC) [25]。这是最先进的学习技术,用于对从在线问题报告中获取的评论中的句子进行分类。他们提出了一种基于卷积神经网络 (CNN) 的方法,将句子分为七类意图:信息给予、信息寻求、特征请求、解决方案建议、问题发现、方面评估和无意义。我们利用特征请求类别来预测包含特征请求句子的对话作为特征请求对话。

基于规则的分类器（FRA） [55]。它是最先进的基于规则的技术,用于对来自在线问题跟踪系统的特征请求中的句子进行分类。他们提出了 81 条模糊规则,将句子分为 6 种类型。我们认为包含Intent 类型句子的对话被预测为特征对话,不包含 Intent 类型句子的对话被预测为非特征对话。

基于机器学习的分类器。朴素贝叶斯 (NB) [38]是一种基于词袋假设和贝叶斯规则的简单文本分类生成模型。它通过模型和训练数据学习到的先验概率和条件概率来进行句子的联合概率。然后,给定一个句子,它可以推断出所有分类的概率。

随机森林（RF） [34]是一种集成机器学习方法,由多棵树构成,每棵树都可以贡献

到最终的分类结果。 Gradient Boosting Decision Tree (GBDT) [29]是另一种集成方法,与 RF 的不同之处在于它的树是由之前的树带来的残差决定的。在训练 GBDT 时,我们设置初始学习为1.0 ,树的最大深度为 1。我们为FT 训练了 100 个 epoch,设置初始学习率为1.0,输入n-gram的窗口大小为 2。) [28]是最先进的文本分类方法,具有类似于word2vec [42] 架构的浅层神经网络。在训练 RF 时,我们将树的最大深度设置为 2。

4.5 绩效指标

在评估 FRMiner 检测聊天消息中特征请求的有效性时,我们使用了以下指标： (1) 精度,指特征对话的正确预测数与特征对话预测总数的比率； (2)召回率,指特征对话的正确预测次数与黄金测试集中特征对话总数的比值； (3) F1-Score,准确率和召回率的调和平均值。

5 结果与分析

5.1 回答 RQ1图 5 展示了不同

方法在 3 折交叉验证中取得的平均性能,表 2 展示了每个项目的详细性能。精度、召回率和 F1 分数的最佳结果以粗体突出显示。我们可以看到,FR Miner 在三个项目中都取得了最好的成绩,平均分别为 88.52%、88.50% 和 88.51% 的准确率、召回率和 F1-分数。我们还注意到 p-FRMiner 的性能优于所有基线方法,这表明在 BiLSTM 对话模型中记忆上下文信息可以使聊天消息中的文本分类任务受益。我们在 5.2 节进一步评估和分析了 FRMiner 相对于 p-FRMiner 的改进。

对于这两种句子方法,基于 CNN 的分类器平均只能达到 17.33% 的 F1-score,主要是因为CNC 模型是通过来自问题评论领域的足够数据而不是特征对话来训练的。然而,它仍然平均达到 48.55% 的召回率,这表明两个领域之间可能存在共同的模式。迁移学习技术可能有助于迁移相关的常识

表 2:在项目内验证中每个项目通过不同方法取得的绩效

方法 \ 表现		AngularJS			引导程序			铬		
		精确召回		F1	精确召回		F1	精确召回		F1
我们的方法	FRMiner	90.28% 89.73% 90.00%			86.28% 88.78% 87.52% 89.00% 87.00% 88.00%					
	p-FRMiner	31.71%	54.17% 40.00%		50.00%	47.80% 48.98%		14.00% 44.00% 20.00%		
现有研究	数控	7.70%	44.44% 13.13%		16.38%	34.21% 22.13%		9.56%	67.00% 16.73%	
	法兰克福机场	13.67%	80.33% 23.35%		23.00%	48.67% 31.00%		12.00%	81.00% 20.00%	
文本分类	注意	20.00%	27.67% 22.33%		25.67%	62.00% 36.00%		14.33% 44.33% 21.00%		
	GBDT	36.00%	22.33% 27.33% 41.67%			35.67% 38.33%		9.33%	7.33%	8.00%
	射频	52.67%	11.00% 16.33%		57.00%	29.00% 38.33%		0.00%	0.00% 不适用	
	英尺	23.33%	5.33%	8.67%	57.67%	29.00% 38.33%		38.00%	9.10% 15.00%	

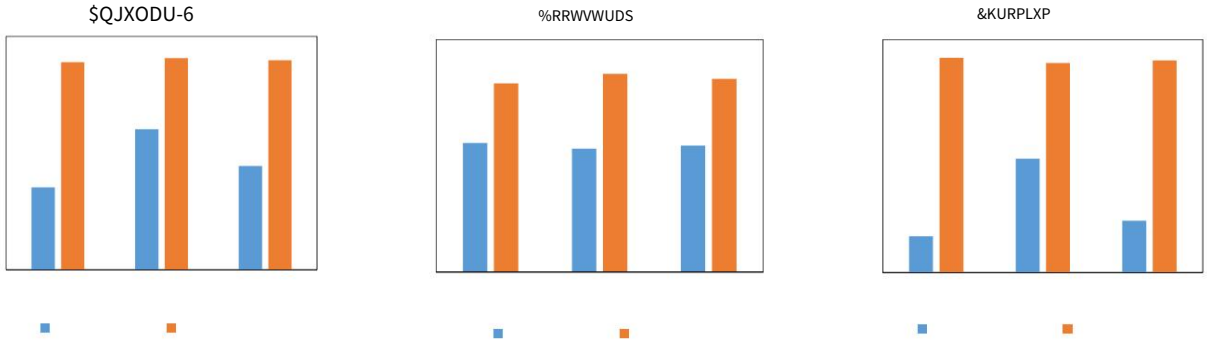

图 6:p-FRMiner 和 FRMiner 在相同数量的原始训练数据下的比较性能。

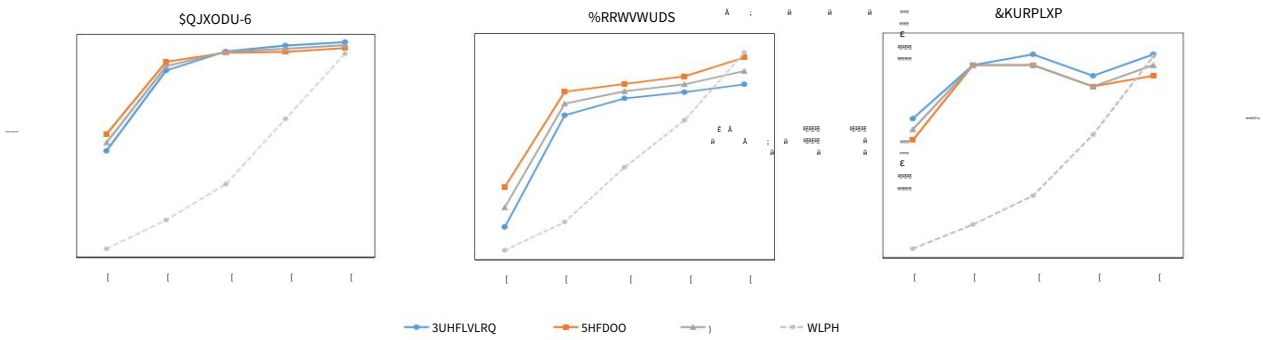

图 7:FRMiner 在生成不同数量对时的性能。

表 3:跨项目验证中每个项目的不同方法取得的绩效

方法 \ 表现		AngularJS			引导程序			铬		
		精确召回		F1	精确召回		F1	精确召回		F1
我们的方法	FRMiner p-	85.23% 86.56% 85.89%			86.84% 85.89% 86.37% 85.87% 86.81% 86.34% 16.00%					
	FRMiner	31.03%	50.00% 38.30%		27.56%	69.08% 39.40%			50.00% 24.24%	
现有研究	数控	7.70%	44.44% 13.13%		16.38%	34.21% 22.13%		9.56%	67.00% 16.73%	
	法兰克福机场	13.67%	80.33% 23.35%		23.00%	48.67% 31.00%		12.00%	81.00% 20.00%	
文本分类	注意	16.00%	75.00% 26.00%		27.00%	36.00% 31.00%		7.00%	26.00% 12.00%	
	GBDT	18.00%	14.00% 16.00%		30.00%	11.00% 16.00%		20.00%	19.00% 19.00%	
	射频	28.00%	14.00% 19.00% 37.00%		19.00%	9.00% 15.00%		12.00%	26.00% 16.00%	
	英尺	32.00%	24.00% 43.00%			13.00% 20.00%		19.00%	11.00% 14.00%	

通过参数传递从问题评论到聊天消息和
微调[47]。同时,基于规则的分类器 FRA 实现了
六种基线方法中最高的召回率。平均值
召回率为 70.00%,在AngularJS 和 Chromium 项目上分别实现了 80.33%
和 81.00% 的召回率。虽然精度是
低,FRA 的预测结果包含大部分实际特征

3UHFLVLRQ 5HFDOO)
3UHFLVLRQ 5HFDOO)
3UHFLVLRQ 5HFDOO)

对话,这意味着聊天消息中的特征请求语句也符合 FRA 规则。由于 FRA 使用规则而不是监督学习,因此在挖掘海量聊天消息时应用起来更容易、更省时。同时,我们可以利用 FRA 的高召回率优势来解决一开始没有提供注释资源的冷启动问题。

对于文本分类方法,NB 似乎是所有文本分类基线中最好的分类器。尽管 RF 在四种方法中平均达到了最高的 27.33% F1 分数,但它在 Chromium 项目上遇到了欠拟合问题。

它既不能对训练数据建模,也不能泛化到新数据,主要是由于 Chromium 数据不够大,RF 模型无法学习相关模式。 FRMiner 明显优于四种传统文本分类模型的原因是:(1)与传统文本分类相比,神经模型具有更大的容量,可以实现更好的性能,特别是对于复杂的对话建模任务。 (2) 模型更容易识别两个对话是否属于同一类,而不是对每个对话进行分类。 (3) 文本分类算法没有从小对话数据集中得到充分的训练,而 FRMiner 是一种成对方法,可以显着增加原始数据集,从而确保训练足够。

总结： FRMiner 明显优于两个句子基准和四种传统文本分类方法。
由于这两个句子基准可以直接应用于聊天消息并实现相对较好的召回率,因此在冷启动情况下它们具有天然优势。

5.2 回答 RQ2图 6 展示了单对话

实例的 p-FRMiner 训练性能和相同大小的配对对话实例的 FRMiner 训练性能。蓝色列表示 p-FRMiner 的训练数据集的大小,即 2 折数据的大小。橙色列表示由 Siamese 网络生成的对实例的大小。我们可以看到,当训练数据集的大小相同时,FRMiner 可以实现比 p-FRMiner 更高的性能。与 p-FRMiner 相比,FRMiner 的 Precision、Recall、F1 分数平均提高了46.79 %、31.13%、42.90%。

图 7 说明了性能增强与训练对实例数量之间的关系,以及训练阶段的时间成本。初始卷 (1x) 是图 6 中所示的训练数据的原始大小,分别为 379、316 和 340对。我们可以看到,扩大训练对实例的大小可以适度提高模型性能。将训练对实例的大小从 1 倍扩大到 30 倍时,准确率、召回率和 F1 分数平均增加 9.82%、8.72% 和 9%。

我们观察到将训练数据集放大 5 倍时性能急剧增加,并且在所有项目上从 5 倍缓慢增加到30 倍。 Chromium 项目的性能在放大 20 倍时甚至略有下降。而训练阶段的时间成本一直在线性增加。

因此,我们认为将训练数据集扩大 5 倍可能是有效性和效率之间的权衡选择。
总结： FRMiner通过显着提高Precision、Recall、F1-score 平均提高了 46.79%、31.13%、42.90%。结果证实,当标记的对话很少时,模型更容易识别两个对话是否属于同一类,而不是直接对确切的类进行分类。我们认为 5 倍是有效性和效率之间的权衡选择,因为在将训练数据集扩大 5 倍后,性能缓慢提高但时间成本大幅上升。

5.3 回答 RQ3图 8 显示了在跨项

目验证设置中不同方法实现的平均性能,表 3 显示了每个项目的详细性能。精度、召回率和 F1 分数的最佳结果以粗体突出显示。

请注意,CNC 和 FRA 的性能与 3 倍项目内验证相同,因为这两种方法没有通过特征对话进行训练。为了比较和分析的目的,我们在表 2 中重复了他们的结果。

我们可以看到,FRMiner 在跨项目设置中也能有很好的表现。与项目内验证的结果相比,性能仅比平均 F1 分数小幅下降 2.27% 。我们认为,表达特征请求的对话在通常与特定领域概念不相关的领域之间共享共同的语言模式。结果表明,FRMiner 可以学习这些常见模式并推广到其他项目。

这表明即使在不同的社区和项目中,开发者表达特性请求的方式也相似,并且不同项目的特性对话具有相似的模式。我们注意到p-FRMiner 的性能不如项目内验证。

平均 F1 分数下降 10.51%。
对于文本分类方法,NB 的 F1 得分最高,为 23%,与项目内验证相比,平均仅略微下降 3.44%。由于用于跨项目验证的训练数据集的大小较大,因此没有一种文本分类方法遇到欠拟合问题。此外,我们注意到大多数文本分类方法在跨项目评估中比在Angular 和 Chromium 的项目内评估中表现更好。主要是由于两个原因:(1)跨项目训练数据集涉及两个项目数据,而项目内只有一个项目的2/3数据。使用更大的数据集进行训练会产生更健壮的分类器。 (2) 跨项目评估导入两个项目进行训练,而项目内评估只有一个项目。由于不同项目引入的有偏见的知识,更广泛的训练数据集将增加分类器的泛化性。

总结： FRMiner在unfitted项目上也能实现高性能,说明FRMiner对其他项目具有泛化性。我们还观察到,NB 是所有文本分类基线中挖掘聊天消息的最佳分类器。

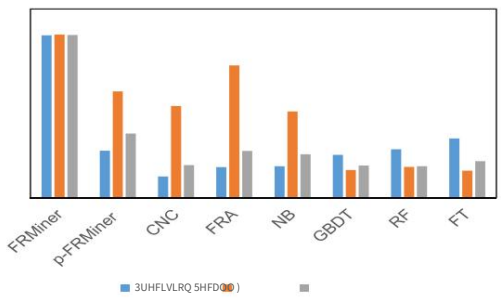

图 8:跨项目验证的平均性能

6 讨论

适用性。我们的工作可以通过方便地将 FRMiner 集成到发布团队成员的工作流程中来收集人群需求。首先,发布团队可以构建聊天消息监控器或爬虫,定期从组织的聊天平台收集文本对话。然后应用自动脚本[33]来预处理和解开原始聊天文本。之后,通过对解开的对话执行 3.4 节中提到的推理过程,FRMiner可以记录所有可能请求特征的对话。发布团队还可以将监控结果订阅为 RSS 提要,以定期接收隐藏功能请求。此外,由于人们以相对一致的方式表达特征对话,FRMiner 用户不需要非常频繁地重新训练模型。重新训练主要需要在数据集的数量或质量发生异常变化时进行。

可扩展性。我们注意到人们以相对一致的方式表达特征对话,例如,除了聊天消息之外,人们还使用类似的表达方式,例如“需要执行……”。在下一个版本中”和“某事。将是一个解决方案/改进”,以指示其他开源平台中的功能请求,包括Github 问题和开发电子邮件。因此,我们认为我们的方法可以扩展到其他数据源。此外, FRMiner 可以应用于其他语言,因为我们的深度上下文对话模型具有很强的语义模式捕获能力。在切换到其他语言时,FRMiner 用户需要将预训练的词嵌入模型适配到特定的语言。他们还需要考虑根据特定语言应用额外的预处理,例如,对中文语料库应用分词。 FRMiner 在语言切换方面的另一个限制与预训练的对话解开模型有关。需要对新的数据集进行注释以重新训练模型,这可能涉及相对较高的成本。

7 对有效性的威胁

外部效度。外部威胁与所提出方法的普遍性有关。这项工作中检查的所有三个系统都是开源项目,可能不代表闭源项目。也有可能我们不小心选择了比平均跨项目特征请求检测性能更好或更差的系统。然而,跨项目评估结果表明,我们的方法在三个研究项目上是可推广的,这在很大程度上减轻了威胁。

内部效度。内部威胁与实验错误和偏差有关。对内部有效性的威胁可能来自对话解开的结果。解开缠结对话的准确性会影响我们的结果。为了减少威胁,当在聊天消息流中分离单个对话时,我们采用了Kummerfeld 等人提出的最先进的技术。 [33],它通过实现 73.5% 的 F1 分数和 91.5 VI2优于以前的研究。

构造效度。构造威胁与评估指标的适用性有关。我们利用精度和召回来评估性能,其中我们使用手动标记的对话作为计算性能指标时的真实情况。威胁可能来自手动检查和标记的过程。我们知道这样的过程可能会出错。

为了减少这种威胁,我们建立了两个检查小组,以就不同的选择达成协议。

8 相关工作

我们的工作与以前的研究有关,这些研究侧重于 (1)特征请求的检测; (2)挖掘开发通信工件。我们简要回顾了每个类别的近期作品。

8.1 特征请求的检测在过去几年中,从人群中收集和分析信息以得出经过验证的用户需求/特征请求的研究量显着增加。

迪索博等人。 [58]提出了一种意图分类法,将开发人员邮件列表中的句子分为六类:功能请求、意见询问、问题发现、解决方案建议、信息搜索和信息提供。尽管该分类已被证明在分析开发电子邮件和来自应用评论的用户反馈方面是有效的[48],但Huang 等人。 [25]发现它不能推广到问题跟踪系统中的讨论,他们通过提出基于卷积神经网络 (CNN) 的方法解决了 Di Sorbo 等人分类法的缺陷。

艾莉亚等人。 [6]通过对 15 个问题讨论线程的定量内容分析,确定了 16 种信息类型,包括潜在的新功能请求。他们还通过使用具有 14 个会话特征的随机森林提供了一种监督分类解决方案,可以以 0.66 F1 分数对表达新特征请求的句子进行分类。莫拉莱斯-拉米雷斯等人。 [43, 44]使用NLP 和语言解析技术支持的 20 条言语行为规则,在 OSS 问题讨论中识别出与需求相关的信息。默滕等人。 [40]研究了自然语言处理和机器学习功能,以检测问题跟踪系统中的软件功能请求。他们的结果表明,软件特征请求检测可以在问题和数据字段的层面上进行,结果令人满意。默滕等人。 [41]还通过手动审查 200 个问题来研究问题跟踪系统中的需求如何传达。他们对文本进行了分类,并报告了问题类型和信息类型的分布。

他们的结果表明,关于优先级和调度的信息可以在自然语言数据中找到。赫齐格等人。 [22]手动检查了 7,000 多个问题报告,并讨论了五个开源项目的错误数据库中错误分类错误的影响。他们的结果表明,39% 的文件被标记为缺陷实际上是新功能、文档更新或内部重构。作者建议,在处理发布的问题时,应该始终让人类参与进来。

安东尼奥等人。 [5]调查了错误跟踪系统中发布的问题文本是否足以将它们分类为纠正性维护和其他类型的活动。他们交替使用各种机器学习方法,例如决策树、朴素贝叶斯分类器和逻辑回归,以区分增强与系统中发布的其他问题。石等人。 [55]提出了 81 条模糊规则,可以将问题中的句子分为六类:意图、利益、缺点、示例、解释和

2 信息变化 (VI) 是衡量从一个聚类到另一个聚类时获得或丢失的信息

琐事。他们的工作旨在帮助理解和分析特征请求的真实意图,这也有助于检测特征请求。罗德盖罗等人。 [39]提出了一种自动化技术,该技术从开发人员与客户的口语对话记录中提取有用信息,以构建用户故事。

他们使用机器学习分类器来确定对话是否包含用户故事信息。 Maalej 和 Nabil [37]利用概率技术以及文本分类、自然语言处理和情感分析技术将应用评论分类为错误报告、功能请求、用户体验和评级。他们的结果表明,分类可以达到 70-95% 的准确率,召回 80-90% 的实际结果。已经发现其他研究也可以自动从应用评论中捕获用户需求[13, 26, 46, 62]。 Vlas 和 Robinson [63]提出了一种基于语法的软件自动化设计,用于发现和分类开源项目存储库中的自然语言需求。克莱德兰-黄等人。 [12]设计了一个自动论坛管理（AFM)系统,用于自动检测已经发布在问题跟踪系统中的重复功能请求。石等人。 [54]提出了一种自动识别特征请求的初始方法,这些请求请求已经通过应用特征树模型实现的特征。总而言之,以前的方法与我们的工作不同:从开发电子邮件中识别出功能请求[25, 58];识别来自问题跟踪系统的功能请求[5、6、22、40、41、43、44、55] ;从口语对话中识别出用户故事[39];从应用评论中识别出功能请求[13、37、46、48、62] ;从项目存储库 [63] 中识别出的功能请求检测到重复的功能请求 [12] [54]。

我们的工作与现有研究的不同之处在于,我们专注于从聊天消息中检测隐藏的特征请求,这些请求会发布不同的挑战,因为聊天消息是非正式的、非结构化的、嘈杂的,并且通常比之前分析的文档没有足够的标记数据。此外,我们的工作补充了现有的关于自动特征请求检测的研究。

8.2 挖矿开发交流神器

先前对开发通信工件的研究表明,在线聊天的使用在软件开发中发挥着越来越重要的作用,而聊天消息是有关软件系统的有价值信息的丰富来源。林等人。 [35]对开发人员如何使用Slack（一种流行的工作场所聊天应用程序)以及他们如何从中受益进行了探索性研究。他们的研究表明,开发人员将 Slack 用于个人、团队和社区范围的目的,而 Slack在软件开发中发挥着越来越重要的作用,在某些情况下取代了电子邮件。希哈布等人。 [56, 57]从几个维度分析了两个大型开源项目的开发者 IRC 会议频道的使用情况:会议内容、会议参与者、他们的贡献和会议风格。他们的结果表明， IRC 会议在开源开发人员中越来越受欢迎,并强调了可以从开发人员聊天消息中获得的丰富信息。于等人。 [66]分析了全球软件开发项目中两种通信机制的使用,即同步（IRC)和异步（邮件列表）。他们的结果表明,开发人员以互补的方式积极使用这两种通信机制。查特吉等人。 [9]进行了一项探索性研究,以调查挖掘开发人员对话在支持软件维护和发展方面的有用性和挑战。他们观察到,开发人员可能会通过即时对话分享关于工具使用、最佳实践和各种技术的观点和有趣的见解。

他们还报告说,通过调整技术和训练集,在解开对话中实现高精度是可行的。阿尔卡迪等人。 [3, 4]从从三个学生项目收集的聊天消息中确定了五个基本原理元素,即问题、替代、支持论点、反对论点和决定。他们开发了两个监督分类器来自动检测手动标记数据上的基本原理元素。伍德等人。 [65]在 bug 修复期间在聊天对话中发现了26 种语音行为类型,并训练了一个监督分类器来自动检测这些语音行为。 Chowdhury 和 Hindle [11]实施了机器学习技术,通过将 StackOverflow 讨论作为正面示例以及将 YouTube 视频评论作为离题讨论示例来过滤编程 IRC频道中的离题讨论。

先前工作的发现激发了本文中提出的工作。我们的研究与之前的工作不同,因为我们专注于检测隐藏在大量聊天消息中的功能请求,这对于 OSS 开发人员增强他们的软件来说是重要且有价值的信息。

9 结论和未来工作

在本文中,我们提出了一种名为 FRMiner 的新方法,它可以通过深度连体网络从聊天消息中检测特征对话。在 FRMiner 中,我们将两个基于 BiLSTM 的对话模型与 Siamese 网络结合起来,以学习一对对话之间的相似性,而不是特定对话的类别。我们在从三个流行的开源项目的大量聊天消息中提取的 1,035 个对话的小样本上评估了 FRMiner 。

实验结果表明,我们的方法显着优于两个句子分类器和四个传统文本分类方法,平均准确率、召回率和 F1分数分别为 88.52%、88.50%。和 88.51%。 FRMiner 还可以在不适合的项目上实现高性能,这表明 FRMiner 可以推广到其他项目。实验结果证实,我们的方法可以有效地检测聊天消息中的隐藏特征请求。我们还观察到,NB 似乎是四个文本分类基线中聊天消息的最佳分类器。未来,我们计划将 NLP 摘要技术与我们的方法一起使用,为开发人员提取一个简短的摘要,这可以减少阅读特征对话的工作量。

此外,我们计划扩展这项工作,不仅对聊天消息进行分类,而且还以精心设计和结构化的格式记录功能请求。

10 致谢

这项工作得到了国家重点研发计划项目No.2018YFB1403400、国家自然科学基金项目No.61802374、 No.61432001、No.61602450的支持。

参考文献 [1] 2019.

echelog。 https://echelog.com/。
[2] 爆炸人工智能。 2019. 太空。 https://spacy.io/。
[3] Rana Alkadhi、Teodora Lata、Emitza Guzmany 和 Bernd Bruegge。 2017.开发聊天消息的基本原理：一项探索性研究。挖掘软件存储库 (2017)，436–446。

[4] Rana Alkadhi、Manuel Nonnenmacher、Emitza Guzman 和 Bernd Bruegge。 2018 年。开发人员如何讨论理由?在 2018 年 IEEE 第 25 届软件分析、进化和再造国际会议 (SANER)。 IEEE，357–369。
[5] Giuliano Antoniol、Kamel Ayari、Massimiliano Di Penta、Foutse Khomh 和 Yann-Gaël Guéhéneuc。 2008. 是错误还是增强？：基于文本的变更请求分类方法。在 2008年合作研究高级研究中心会议论文集：思想会议。 ACM，

23.
[6] Deeksha Arya, Wenting Wang, Jin LC Guo, and Jinghui Cheng. 2019.开源软件问题讨论信息类型分析与检测。
在第 41 届国际软件工程会议论文集上，ICSE 2019，蒙特利尔，QC，加拿大，2019 年 5 月 25-31 日。454-464。
[7]詹姆斯·伯格斯特拉和约书亚·本吉奥。 2012. 随机搜索超参数优化。机器学习研究杂志 13, 1 (2012), 281–305。
[8] Jane Bromley、Isabelle Guyon、Yann LeCun、Eduard Säckinger 和 Roopak Shah。 1994. 使用“连体”时间延迟神经网络进行签名验证。在神经信息处理系统的进展中。 737–744。
[9] Preetha Chatterjee、Kostadin Damevski、Lori Pollock、Vinay Augustine 和Nicholas A Kraft。 2019. Slack Q&A 聊天作为软件工程工具挖掘资源的探索性研究。在第 16 届国际采矿软件存储库会议论文集上。 IEEE 出版社，490–501。

[10] Wei-Yu Chen、Yen-Cheng Liu、Zsolt Kira、Yu-Chiang Wang 和 Jia-Bin Huang。 2019. 仔细观察小样本分类。在国际学习代表大会上。

[11] Shaiful Alam Chowdhury 和 Abram Hindle。 2015. 挖掘 StackOverflow 以过滤掉离题的 IRC 讨论。 (2015), 422–425。
[12] Jane Cleland-Huang、Horatiu Dumitru、Cuan Duan 和 Carlos Castro-Herrera。 2009. 自动支持在开放论坛中管理功能请求。
交流。 ACM 52, 10 (2009), 68–74。
[13] Andrea Di Sorbo、Sebastiano Panichella、Carol V. Alexandru、Junji Shimagaki、Cor rado A. Visaggio、Gerardo Canfora 和 Harald C. Gall。 2016. 用户在我的应用中会发生什么变化?总结应用程序评论以推荐软件更改。在 2016 年第 24 届 ACM SIGSOFT软件工程基础国际研讨会论文集上。 499-510。

[14] 脸书。 2019. 快文。 https://fasttext.cc/。
[15] 脸书。 2019. PyTorch。 https://pytorch.org/。
[16]艾伦人工智能研究所。 2019.艾伦NLP。 https://allennlp.org/。
[17] 谷歌。 2019. AngularJS。 https://angularjs.org/。
[18] 谷歌。 2019. 铬。 https://www.chromium.org/。
[19] 谷歌。 2019. Scikit-学习。 https://scikit-learn.org/。
[20]亚历克斯·格雷夫斯、阿卜杜勒-拉赫曼·穆罕默德和杰弗里·辛顿。 2013. 使用深度循环神经网络进行语音识别。 2013年IEEE国际声学、语音和信号处理会议。 IEEE，6645-6649。

[21] Eduard C Groen、Norbert Seyff、Raian Ali、Fabiano Dalpiaz、Joerg Doerr、Emitza Guzman、Mahmood Hosseini、Jordi Marco、Marc Oriol、Anna Perini 等。 2017 年。
需求工程中的人群：前景和挑战。 IEEE软件 34, 2 (2017), 44–52。

[22]金赫齐格、萨沙贾斯特和安德烈亚斯泽勒。 2013. 这不是错误，而是一个特征：错误分类如何影响错误预测。在 2013 年国际软件工程会议论文集上。 IEEE 出版社，392–401。

[23] Sepp Hochreiter 和 Jürgen Schmidhuber。 1997. 长短期记忆。神经计算 9, 8 (1997), 1735–1780。
[24]黄安娜。 2008. 文本文档聚类的相似性度量。在第六届新西兰计算机科学研究生会议 (NZCSRSC2008) 的会议记录中，新西兰基督城，卷。 4. 9-56。

[25]黄乔、夏心、罗大卫和盖尔 C. 墨菲。 2018. 自动化意图挖掘。 IEEE 软件工程汇刊 PP，99（2018 年），1-1。
[26] Nishant Jha 和阿纳斯·马哈茂德。 2017.使用框架语义从应用商店评论中挖掘用户需求。 (2017), 273–287。
[27]托尔斯滕·约阿希姆斯。 1996. 使用TFIDF 对 Rocchio 算法进行概率分析，用于文本分类。技术报告。卡内基梅隆大学匹兹堡大学计算机科学系。

[28] Armand Joulin、Edouard Grave、Piotr Bojanowski、Matthijs Douze、Hérve Jégou和 Tomas Mikolov。 2016. 快文。 zip：压缩文本分类模型。 arXiv 预印本 arXiv:1612.03651 (2016)。
[29]柯国林，孟启，Thomas Finley，王泰峰，陈伟，马卫东，叶启伟，刘铁燕。 2017. Lightgbm：一种高效的梯度提升决策树。在神经信息处理系统的进展。 3146–3154。

[30]尹金。 2014. 用于句子分类的卷积神经网络。 arXiv预印本 arXiv:1408.5882 (2014)。

[31]罗恩·科哈维。 1995. 用于准确性估计和模型选择的交叉验证和引导程序研究。在第十四届国际人工智能联合会议论文集上，IJCAI 95，加拿大蒙特利尔， 1995 年 8 月 20 日至 25 日，2 卷。 1137–1145。

[32] Alex Krizhevsky、Ilya Sutskever 和 Geoffrey E Hinton。 2012. Imagenet 分类与深度卷积神经网络。在神经信息处理系统的进展中。 1097–1105。

[33] Jonathan K. Kummerfeld、Sai R. Gouravajhala、Joseph Peper、Vignesh Athreya、Chulaka Gunasekara、Jatin Ganhotra、Siva Sankalp Patel、Lazaros Polymenakos和 Walter S. Lasecki。 2019. 用于对话解开纠缠的大规模语料库。在计算语言学协会第 57 届年会论文集（第 1 卷：长篇论文）中。

[34]安迪·廖、马修·维纳等人。 2002. 随机森林的分类和回归。 R 新闻 2, 3 (2002), 18–22。

[35] Bin Lin、Alexey Zagalsky、Margaret-Anne D. Storey 和 Alexander Serebrenik。 2016. 开发人员为何懈怠：了解软件团队如何使用 Slack。在第 19 届 ACM 计算机支持的合作工作和社会计算会议论文集上。 333–336。

[36] 查尔斯 X 凌和李成辉。 1998. 直接营销的数据挖掘：概率
lems 和解决方案.. 在 Kdd，Vol。 98. 73–79。
[37]瓦利德·马莱杰和哈迪尔·纳比尔。 2015. 错误报告、功能请求还是简单的赞美?关于自动分类应用评论。 2015年IEEE第23届国际需求工程会议（RE）。 IEEE，116-125。

[38]安德鲁·麦卡勒姆、卡迈勒·尼加姆等人。 1998. 朴素贝叶斯文本分类的事件模型比较。在关于学习文本分类的 AAAI-98 研讨会上，卷。 752. Citeseer，41-48。

[39]科林·麦克米兰、科林·麦克米兰、科林·麦克米兰和科林·麦克米兰。 2017 年。
检测开发人员与客户对话中的用户故事信息以生成提取摘要。在 ieee/acm 国际软件工程会议上。 49-59。

[40] Thorsten Merten、Matú Falis、Paul Hübner、Thomas Quirchmayr、Simone Bürsner和 Barbara Paech。 2016. 问题跟踪系统中的软件功能请求检测。在 2016 年 IEEE 第 24 届国际需求工程会议 (RE) 中。
IEEE，166-175。
[41] Thorsten Merten、Bastian Mager、Paul Hübner、Thomas Quirchmayr、Barbara Paech 和 Simone Bürsner。 2015.四个开源项目中问题跟踪系统中的需求沟通.. 在 REFSQ 研讨会上。 114–125。
[42] Tomas Mikolov、Ilya Sutskever、Kai Chen、Greg S Corrado 和 Jeff Dean。 2013 年。单词和短语的分布式表示及其组合性。在神经信息处理系统的进展中。 3111-3119。

[43] Itzel Morales-Ramirez、Fitsum Meshesha Kifetew 和 Anna Perini。 2017.支持需求发现的在线讨论分析。在高级信息系统工程中。 159–174。

[44] Itzel Morales-Ramirez、Fitsum Meshesha Kifetew 和 Anna Perini。 2018. 基于语音行为的在线讨论需求发现分析。信息系统（2018 年）。

[45]乔纳斯·穆勒和 Aditya Thyagarajan。 2016.用于学习句子相似性的连体循环架构。在第三十届 AAAI 人工智能会议上。

[46] Fabio Palomba、Mario Linares Vásquez、Gabriele Bavota、Rocco Oliveto、Massim iliano Di Penta、Denys Poshyvanyk 和 Andrea De Lucia。 2015. 用户评论很重要！跟踪众包评论以支持成功应用程序的演变。
在 2015 年 IEEE 软件维护和演进国际会议上， ICSME 2015，德国不来梅，2015 年 9 月 29 日至 10 月 1 日。291–300。
[47]新野家林潘、杨强． 2010. 迁移学习调查。 IEEE Transactions on Knowledge and Data Engineering 22, 10 (2010), 1345–1359。
[48] Sebastiano Panichella、Andrea Di Sorbo、Emitza Guzman、Corrado Aaron Vis aggio、Gerardo Canfora 和 Harald C Gall。 2015. 我如何改进我的应用程序？
对软件维护和发展的用户评论进行分类。 (2015), 281–
290。
[49]杰弗里·彭宁顿、理查德·索彻和克里斯托弗·曼宁。 2014. 手套：词表示的全局向量。在 2014 年自然语言处理经验方法会议论文集(EMNLP) 上。 1532-1543 年。

[50] Antônio Mauricio Pitangueira、Paolo Tonella、Angelo Susi、Rita Suzana Pi tangueira Maciel 和 Márcio de Oliveira Barros。 2017. 最小化利益相关者在下一个发布计划的需求选择中的不满意风险。
信息与软件技术 87 (2017), 104–118。
[51] Germán Poo-Caamaño、Eric Knauss、Leif Singer 和 Daniel M German。 2017 年。在 FOSS 生态系统中放牧猫：发布管理的沟通和协调故事。互联网服务与应用杂志 8, 1 (2017), 12。
[52]卢茨·普雷切尔特。 1998. 提前停止 但是什么时候?在神经网络中：交易技巧。施普林格，55-69。

[53] 抓取中心。 2019. 刮痧。 https://scrapy.org/。
[54] Lin Shi、Celia Chen、Qing Wang 和 Barry W. Boehm。 2016. 它是一个新功能还是只是“还不知道”？：关于自动冗余 OSS 功能请求识别。在第 24 届 IEEE 国际需求工程会议上， RE 2016，北京，中国，2016 年 9 月 12-16 日。377-382。

[55] Lin Shi、Celia Chen、Qing Wang、Shoubin Li 和 Barry W Boehm。 2017.通过利用模糊方法和语言分析来理解特征请求。自动化软件工程（2017），440–450。

[56] Emad Shihab、Zhen Ming Jiang 和 Ahmed E. Hassan。 2009. 关于GNOME GTK+ 项目的开发人员使用 Internet 中继聊天 (IRC) 会议。
在第 6 届国际挖掘软件存储库工作会议论文集上，MSR 2009（与 ICSE 共同举办），加拿大不列颠哥伦比亚省温哥华，2009 年 5 月 16-17日，论文集。 107-110。

[57] Emad Shihab、Zhen Ming Jiang 和 Ahmed E. Hassan。 2009. 研究在开源项目中使用开发者 IRC 会议。在第 25 届 IEEE软件维护国际会议 (ICSM 2009) 上，2009 年 9 月 20-26 日，加拿大艾伯塔省埃德蒙顿。 147–156。

[58] Andrea Di Sorbo、Sebastiano Panichella、Corrado Aaron Visaggio、Massim iliano Di Penta、Gerardo Canfora 和 Harald C. Gall。 2015. 开发电子邮件内容分析器：开发人员讨论中的意图挖掘 (T)。在第 30 届 IEEE/ACM 自动化软件工程国际会议上，ASE 2015，美国内布拉斯加州林肯市，2015 年 11 月 9-13 日。12-23。

[59] Nitish Srivastava、Geoffrey Hinton、Alex Krizhevsky、Ilya Sutskever 和 Ruslan Salakhutdinov。 2014. Dropout：一种防止神经网络过度拟合的简单方法。机器学习研究杂志 15, 1 (2014), 1929–1958。

[60] 引导团队。 2019. 引导程序。 https://getbootstrap.com/。

[61]何塞·阿波利纳里奥·特谢拉和海伦娜·卡斯滕。 2019. 设法在 OpenStack 软件生态系统中尽早、经常和按时发布。互联网服务与应用杂志 10, 1 (2019), 7。

[62] Lorenzo Villarroel、Gabriele Bavota、Barbara Russo、Rocco Oliveto 和 Massimil iano Di Penta。 2016. 基于用户评论的移动应用发布规划。在第 38 届国际软件工程会议论文集上。 14-24。

[63]拉杜 E 弗拉斯和威廉 N 罗宾逊。 2012.开源软件开发项目中需求发现和分类的两种基于规则的自然语言策略。管理信息系统杂志 28, 4 (2012), 11–38。

[64]王亚庆、姚全明． 2019. 小样本学习：一项调查。 arXiv预印本 arXiv:1904.05046 (2019)。

[65]安德鲁·伍德、佩奇·罗德盖罗、阿米尔·阿玛利和科林·麦克米兰。 2018 年。
在错误修复期间检测开发人员问答对话中的言语行为类型。在 2018 ACM 欧洲软件工程会议和软件工程基础研讨会的会议记录中， ESEC/SIGSOFT FSE。 491-502。

[66]于立国、斯里尼·拉马斯瓦米、阿洛克·米什拉和迪普蒂·米什拉。 2011. 全球软件开发中的通信：使用 GTK+ OSS存储库的实证研究。在 2011 年关于转向有意义的利益系统的国际联合会议的论文集中，OTM 11。 218–227。