基于提示学习的 API 评论知识图谱构建方法

杨延军[1,2]　刘名威*[1,2]　彭　鑫[1,2]　赵文耘[1,2]
[1](复旦大学计算机科学技术学院　上海 200438)
[2](上海市数据科学重点实验室　上海 200438)

摘　要　API 是现代软件开发中提升效率的重要组件，开发者经常在问答社区的评论中学习不同类型的 API 知识，由此引发一系列对 API 评论进行分类的研究工作。然而相关工作在分类时均需要大量标注数据，成本较高。为此，提出了一种基于提示学习的 API 评论知识图谱构建方法，能够将评论分类任务转化为模型擅长的单词预测任务从而提升分类效果。相对于基线方法，该方法的 F1 指标提升 16.2%-21%，召回率提升 30.6%-36.2%。此外，所提出的知识图谱结构可以将不同帖子中的 API 评论按照不同粒度、不同类别、去冗余地整合在一起。并且通过实验证明所建知识图谱能够有效帮助开发者学习相关 API 知识。

关键词　API 评论　提示学习　知识图谱

中图分类号　TP3　　**文献标志码**　A　　DOI: 10.3969/j.issn.1000-386x.2018.01.001

CONSTRUCTION METHOD OF API REVIEWS KNOWLEDGE GRAPH BASED ON PROMPT LEARNING

Yang Yanjun[1,2]　Liu Mingwei*[1,2]　Peng Xin[1,2]　Zhao Wenyun[1,2]
[1](*School of Computer Science, Fudan University, Shanghai 200438,China*)
[2](*Shanghai Key Laboratory of Data Science, Fudan University, Shanghai 200438,China*)

Abstract　Application Programming Interfaces are important software components to improve development efficiency in modern software development. Developers often learn different types of API knowledge from comments in question-and-answer communities such as Stack Overflow, which leads to many research works on API comments classification. However, existing works need to use a large amount of manually labeled data in the classification which is of high cost. To solve these problems, this paper proposes a method for constructing an API reviews knowledge graph based on prompt learning. This method improves the classification effect by converting the review classification task into the masked token prediction that the model is good at. Compared with baseline methods, our F1 metric has increased by 16.2%-21%, and the recall has increased by 30.6%-36.2%. In addition, the knowledge graph structure proposed in this paper can integrate API comments in different posts according to different granularities and categories, and eliminate redundancy. The experiments prove API reviews knowledge graph can effectively help developers learn API knowledge.

Keywords　API reviews　Prompt Learning　Knowledge Graph

0 引　言

API（Application Programming Interface）是可以复用的应用程序编程接口，现代软件开发过程中对 API 的使用往往能够提升开发效率并降低开发成本[1]。如今随着第三方托管平台（如 Maven、NPM、PIPY 等）的兴起，越来越多的第三方库随之发布，极大丰富了开发人员对于 API 的选择。然而，众多 API 也给开发人员带来了较大的学习压力和选择困难。例如部分 API 缺少完整的 API 参考文档，开发者难以找到较好的学习资源；API 参考文档通常都是围绕 API 的功能进行描述，较少涉及非功能信息，如性能、可用性、安全性和潜在软件缺陷等。

针对以上问题，开发问答社区（如 Stack Overflow）中对于 API 的讨论可以为开发者提供帮助。然而，针对某一个特定的 API 而言，相关的评论往往零散存在

收稿日期: xxxx-xx-xx。国家自然科学基金(61972098)。**杨延军**，硕士生，主研领域：智能化软件开发。**刘名威**，博士。**彭鑫**，教授。**赵文耘**，教授。

于不同帖子中，并不利于开发者在特定场景下进行学习了解。因此开发人员通常要根据特定开发需求分析总结相应评论知识[4]。目前存在一些对 API 评论进行分类的研究工作[3-5]，然而这些工作通常基于机器学习算法或大规模预训练语言模型进行调优，均需要使用大量人工标注数据以一种监督的方式对模型进行训练，存在较高实现成本。此外，现有工作对于 API 评论知识的组织形式并不利于将多源知识进行关联。

为了解决以上问题，本文提出了一种基于提示学习的 API 评论知识图谱构建方法，可以在低资源条件下对 API 评论进行有效的类别划分，从而帮助开发人员更加高效且有选择性的利用 API 评论知识。相对于基线方法，本文方法在 F1 指标上提升了 16.1%-21%，召回率指标上提升了 30.6%-36.2%。本文通过实验证明了所建知识图谱包含的 API 评论知识具有较高的有用性和可读性，能够有效帮助开发人员学习相关 API 知识。本文主要贡献如下：

（1）提出了一种基于提示学习的 API 评论分类方法，能够在低资源的条件下对 API 评论进行有效分类。

（2）提出了一个 API 评论知识图谱的构建方法，能够按照不同粒度、不同类别、去冗余地将不同帖子中的 API 评论聚合到一起。并基于该方法构建了一个包含 3,185,819 个实体的 API 评论知识图谱。

（3）通过实验证明了本文分类方法的有效性，以及所建 API 评论知识图谱对于 API 学习的有用性。

1 相关工作

1.1 API 评论知识分析

随着问答社区的日益活跃，Stack Overflow 上涌现出越来越多关于 API 的讨论。与 API 参考文档提供的功能信息不同，问答社区中的用户评论通常含有大量 API 的非功能信息，特别是开发人员关注的性能、可用性、安全性和软件缺陷等。由此引发了一系列对 API 评论进行分析的研究工作。其中，Treude 等人[2]利用 Stack Overflow 中与特定 API 相关的语句，提出了一种自动增强 API 参考文档的方法，可以补充 API 参考文档中没有涉及到的 API 知识。Tang 等人[3]从 Stack Overflow 中挖掘评论编辑历史，并对编辑过的评论记录进行分类，发现大部分编辑原因在于更正、作废、 缺陷和扩展。Uddin 等人[4]调查了开发人员对于 API 评论的利用情况，总结了开发者希望在问答社区中看到的众多 API 评论类型，并提出了一种利用机器学习自动挖掘 API 评论的方法。Yang 等人[5]在最近的研究工作中微调了六个预训练语言模型对 Stack Overflow 中的 API 评论进行分类，并和传统的机器学习算法进行对比，验证了预训练语言模型的性能。

然而，上述工作使用机器学习算法或者微调后的预训练语言模型对 API 评论进行分类时，均需要大量的人工标注数据，成本较高且准确率有待进一步提升。而本文提出的基于提示学习的 API 评论分类方法能够在低资源条件下实现良好的分类效果。

1.2 软件工程领域少样本学习应用

与需要大量标注数据的传统监督学习相比，少样本学习近年来得到研究人员广泛关注，其中提示学习（Prompt Learning）更是成为自然语言处理领域的一种新范式[6]。Helmeczi 等人[7]将提示学习用于软件需求规范文档中的冲突检测，仅使用 32 个训练样本就可以在关键指标上达到较好效果。罗贤昌等人[8]应用提示学习的思想，将软件需求分类问题从 K 分类任务转化为二分类任务，并取得比传统机器学习算法和 BERT 模型更好的分类效果。此外，Chai 等人[9]提出了一种用于特定领域代码搜索的方法 CDCS，该方法利用迁移学习将通用编程语言数据上训练的模型参数应用到其它少样本的编程语言中（如 SQL 和 Solidity）。张宁豫等人[10]提出了一种低资源条件下的知识图谱补全方法，该方法利用知识图谱中已有的显式结构化知识和语言模型中的隐式知识，能够对初始知识图谱进行补全，并在链接预测和关系抽取任务上取得了较好效果。

据本文所知，本文是第一个提出将提示学习应用到 API 评论分类任务的研究工作。对 API 评论进行准确分类能够帮助开发人员更高效且有选择性的学习 API 知识。

1.3 API 知识图谱构建及应用

由于知识图谱的结构能够更好的将多源异构的知识组织在一起。近年来，研究人员为不同类型知识和不同目的构造了软件工程领域的知识图谱，从而实现对多源知识的高效利用。由于现代软件开发中 API 的广泛应用，一些研究人员构建了 API 知识图谱。例如，邢双双等人[11]利用 API 参考文档和开发问答文本中的概念和关系构造了一个软件知识图谱，基于该知识图谱可以实现为给定的代码片段添加语义标签，以辅助开发人员快速了解代码含义。马展等人[12] 通过挖掘 API 参考文档和 Stack Overflow 中问答帖子的知识,构建了一个 API 知识图谱, 以实现更准确的 API 推荐。Liu 等人[13]提出了一种基于知识图谱的 API 比较法，该方法能够将 API 参考文档中的知识构建为 API 知识图谱从而对 API 进行不同方面的比较。

受以上工作启发，为了更好的整合具有结构关系或者相似特性的 API 评论知识，本文将 Stack Overflow 中的 API 评论构建为知识图谱。

2 API 评论知识图谱构建方法

下面将从方法概述、知识图谱高层模式以及方法的具体实现细节对本文进行介绍。

2.1 方法概述

本文提出了一种基于提示学习的 API 评论知识图谱构建方法，该方法能够整合问答社区不同帖子中的 API 评论知识，并将 API 评论按照不同粒度，不同类别，去冗余地聚合到一起。

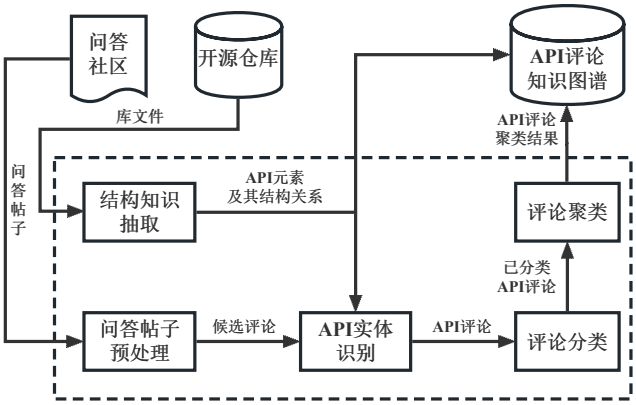

图 1 方法流程图

本文提出的 API 评论知识图谱构建方法流程如图 1 所示，方法的输入为开源仓库中的库文件（如 Java 文件）和问答社区的帖子，输出为 API 评论知识图谱。具体步骤如下：

（1）从开源软件仓库中获取大量第三方库文件，并利用静态解析从库文件中抽取不同粒度的 API 元素（例如：库、包、类和方法等）及其结构关系（例如：包含、继承和实现等），并将其构建为 API 评论知识图谱骨架。

（2）从问答社区中获取问答帖子，进行代码块替换，句子切分等预处理后，获得候选评论。

（3）利用基于规则的方法识别候选评论中包含的不同类型的第三方库 API。

（4）利用基于提示学习的方法，实现对 API 评论 11 个类别的划分。

（5）使用聚类算法对高相似度的评论进行聚类和去冗余处理后，链接到（1）中所构建的知识图谱骨架上，从而完成 API 评论知识图谱构建。

2.2 知识图谱高层模式

本文所构建的 API 评论知识图谱高层模式如图 2 所示（为了简洁起见，图中省略了一些关系及标注），图中不同图形表示不同类型的实体元素，不同线型表示不同类型的关系。下面对知识图谱中的实体和关系类型进行介绍（括号内容为 API 评论知识图谱中的类型标签）。

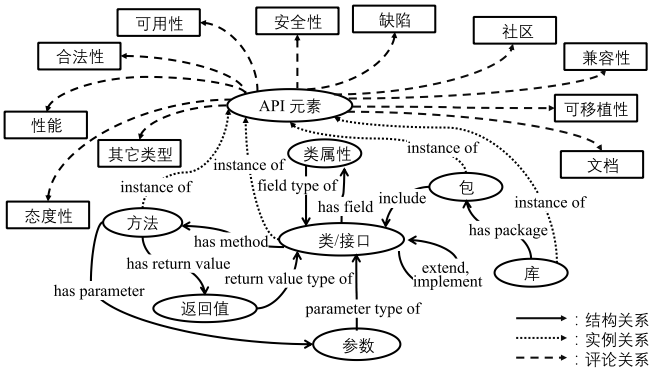

图 2 API 评论知识图谱高层模式

实体部分：API 评论知识图谱共有 2 种类型实体，即 API 实体和评论实体，其中椭圆表示 9 种类型的 API 实体，包括：库（Library）、包（Package）、类（Class）、接口（Interface）、类属性（Field）、方法（Method）、返回值（Return Value）参数（Parameter）和 API 元素（API Element）。矩形表示 11 种不同类别的评论实体，包括：性能（Performance）、可用性（Usability）、安全性（Security）、缺陷（Bug）、社区（Community）、兼容性（Compatibility）、文档（Documentation）、合法性（Legal）、可移植性（Portability）、态度性（Only Sentiment）和其它类型（Others）。评论类别的划分依据是 Uddin 等人[4]的研究工作，该工作调研并总结了开发者们最希望看到的一些评论类型。

关系部分：API 评论知识图谱共有 2 种类型关系，即结构关系和评论关系。其中实线表示结构性关系，包括：拥有包（has package）、拥有类（has class）、拥有接口（has interface）、扩展（extend）、实现（implement）、拥有属性（has field）、拥有方法（has method）、包含参数（has parameter）、包含返回值（has return value）、参数类型（parameter type of）和返回值类型（return value type of）等。图中的圆点线表示实例关系（instance of），图中所有 API 实体类型均属于 API 元素（API Element）的实例而与其够成<*API, instance of, API Element*>关系，为了简洁起见，图中只体现了部分 instance of 关系。图中与矩形连接的虚线表示评论关系，包括：拥有性能评论（has performance review）、拥有可用性评论（has usability review）、拥有安全性评论（has security review）、拥有缺陷评论（has bug review）等。

此外，当一条评论同时提到多个 API 时，可能代表不同 API 在相关特性上存在潜在联系，因此对于同一条评论提到的多个 API，为其建立相应特性的关联。例如在评论文本“*The calls to File.exists() and to File.canRead() are also expensive because they each entail a system call, and that is going to take thousands of clock cycles.*”中，同时提到了 File.exists()和 File.canRead()两个 API，则按照评论的类别（performance）为其建立

*<File.exists(),performance related to, File.canRead()>*和*<File.canRead(), performance related to, File.exists()>*两条对称关系。

基于本文提出的 API 评论知识图谱，来自不同帖子的评论可以通过 API 的结构关系或相似特性聚合在一起。例如：属于同一个类的方法评论可以聚合到相应类中，属于同一个库中不同 API 的评论可以聚合到该库中，拥有相似性能特性的 API 及其评论可以聚合到相应实体中。

2.3 结构知识抽取

API 结构知识抽取阶段分为库文件获取和库文件静态解析两个环节。值得说明的是，由于 Java 编程语言的广泛使用，本文将聚焦于 Java 语言的库，但本文的方法并不限定于 Java 编程语言，可以通过修改库文件的静态解析方法以支持其它编程语言库。下面介绍以上两个环节的方法流程。

库文件获取：Libraries.io 是一个开源的搜索引擎，利用该引擎可以找到各种编程语言的第三方库信息。因此本文通过 Libraries.io 获取关于 Java 库的元信息（包括 groupId 标签和 artifactId 标签信息），并利用这些信息从 Maven 中央仓库获得第三方库的 JAR 压缩包。然后，本文使用 ZipFile 工具提取所有第三方库的 Java 源文件，以作为后续静态解析的数据输入。

库文件静态解析：第三方库源文件包含着库中所有的 API 及其结构关系，这些信息可以通过对源文件进行静态解析来获取。静态解析技术是指在不运行程序的情况下对程序代码进行分析的方式，抽象语法树（Abstract Syntax Tree）是静态解析中常用的技术，本文通过将 Java 代码构建为抽象语法树，实现了对 9 类 API 实体和 11 类实体间结构关系的抽取。基于第三方库的所有 API 实体及其结构关系，本文将其构建为图 2 所示的知识图谱骨架（由图中椭圆和实线构成）。

2.4 候选评论获取

本文使用 Stack Overflow 作为抽取 API 评论的数据源。Stack Overflow 是一个流行的编程问答社区，允许用户以问答帖子的形式提出或回答问题，并支持其他用户对问题或答案给出评论。Stack Overflow 上的帖子结构分为标题、全文和标签等信息。其中包含描述性文本的是帖子标题和帖子全文。由于帖子标题通常以问题而非陈述客观事实的形式出现，因此本文选取帖子全文作为原始数据。此外，部分帖子内容中还包含用户附加的代码片段，而这些代码片段并不包含或极少包含 API 评论信息，因此本文使用基于规则的方式检测帖子中的代码片段，并将其用-CODE-符号替换。

Stack Overflow 帖子的全文包含了众多句子，不同句子可能包含不同的含义。为了让评论内容更加清晰简洁，本文对帖子全文进行句子级别的切分，从而更细粒度的解析帖子中包含的 API 评论信息。SpaCy 是一个自然语言工具包，本文利用 SpaCy 对帖子内容进行语法分析，完成帖子全文的句子级别切分，并将每一个句子视为候选评论。

2.5 API 实体识别

API 实体识别的目标是识别候选评论中包含的 API 实体，并过滤掉不包含 API 实体的候选评论。由于问答社区的讨论涉及大量同名的 API，简单依靠 API 名称难以进行区分，尤其如 get()、set()等常见的方法名。为了更准确的识别评论中的 API 实体，本文将结构知识抽取环节得到的 API 实体连同其上下位关系，共同作为 API 实体的唯一区分。例如：对于方法的识别通过“类名.方法名”来确认；对于类的识别，使用“包名.类名”；对于包的识别，使用“库名.包名”；对于库而言，本文构建了一个包含通用领域和软件工程领域常见词（共 3600 个）的停用词表，将不在列表中的 artifactId 作为相应库的识别方式。基于上述对不同类型实体的识别方式，本文构建了一个基于规则的 API 实体匹配器，该匹配器可以检索输入文本中包含的 API 实体及相应类型（如库、包、类、方法等）。通过匹配器识别，本文将不包含 API 实体的候选评论过滤，并保留其余句子作为 API 评论。

2.6 评论分类

评论分类的目标是将 API 评论按照其含义划分为不同类型，从而便于开发者高效且有选择性的利用 API 评论知识。根据 Uddin 等人[4]对 178 名开发者的调查发现，开发人员更希望在问答社区中获取到**表 1** 所示 11 个类别的 API 评论，因此本文基于此标准对 API 评论进行类别划分。

对于分类任务而言，大规模预训练语言模型几乎取代了传统的机器学习算法。本文对 API 评论进行分类时，使用的提示学习方法也是基于 BERT[14]和 RoBERTa[15]两种预训练模型进行调优的。其中，BERT（**B**idirectional **E**ncoder **R**epresentations from **T**ransformers）是一种基于转换器（Transformer）的双向编码器（Encoder）表示，它通过两个无监督任务进行预训练，即屏蔽的语言模型(Masked Language Model，MLM)和下一句预测(Next Sentence Prediction，NSP)。MLM 任务是随机将句子序列中的一些单词屏蔽后作为输入，然后根据序列中其它非屏蔽单词提供的上下文来预测被屏蔽的单词。NSP 任务是随机选取同一文

章中连续的两个句子或不同文章的两个句子，然后预测这两个句子序列是否连续。RoBERTa（**R**obustly **O**ptimized **BERT** Pretraining **A**pproach）是对 BERT 训练过程进行改进的模型，改进包括更多的训练数据、更长的训练序列以及更大的批量等。此外，RoBERTa 在训练时动态修改 MLM 任务中的单词屏蔽模式，并删除了 BERT 模型中的 NSP 任务。

表 1 API 评论类别及定义[4]

分类	定义
性能	在速度、内存占用、可伸缩性等方面的表现
可用性	是否易使用，以及满足特定开发需求时的效果
安全性	是否能够保障安全，括验证、授权、加密等
缺陷	特定 bug 或异常的出现、扩散、修复等
社区	相关社区的支持、活跃度等，包括邮件列表、开发者论坛、API 作者等
兼容性	API 的使用是否需要依赖其它软件或部署环境
文档	相关文档的可获得性，质量好/坏等，包括官方文档、社区讨论、博客帖子等
合法性	关于 API 的许可权限和定价问题，包括是否开源及相应开源协议
可移植性	在不同的平台的可用情况，如 Linux、Mac OS、Windows 等不同操作系统
态度性	未指明特定的方面主观态度，如偏好，情绪等
其它类型	无法被划分到上述 10 个类别的其它类型

虽然预训练模型如今在软件工程领域的众多任务中都拥有出色的表现，但为了获得较好的分类效果，仍然需要使用大量人工标注数据，并以一种监督的方式对预训练模型进行微调，而该方式存在较高成本。近几年，提示学习成为 NLP 领域研究的新模式，其核心思想是将下游任务转化为与模型预训练阶段相似的任务形式，从而缩小下游任务与预训练任务之间的差距。研究者们发现文本提示可以在少量标注数据下有效地提升预训练模型的性能[6]。本文的研究工作聚焦于低资源条件下的知识图谱构建，因此在评论分类时采用提示学习方法以获得较好的分类效果。

图 3 展示的是本文使用提示学习方法对输入文本 *“List.contains() is a slow operation in comparison to Set.contains(), so it may be worth coming up with a hybrid if speed is a concern.”*进行性能类别（图中[perfoamance]）的分类示例，图中 P1-P4 是不同类型的提示学习模版，[Mask]代表被屏蔽的标签词，PLM 代表预训练语言模型。提示学习方法在模型训练过程中，可以通过提前定义的提示模版（如“I think it [Mask] related to [preformance].”）对模型进行提示，然后使用不同预训练模型预测被屏蔽的标签词（如 is 或 isn’t）。可以发现，提示学习可以将原本的评论分类任务转化为屏蔽单词补全任务，而该任务与相应模型预训练阶段的 MLM 任务高度相似，因此可以取得较好的评论分类效果。

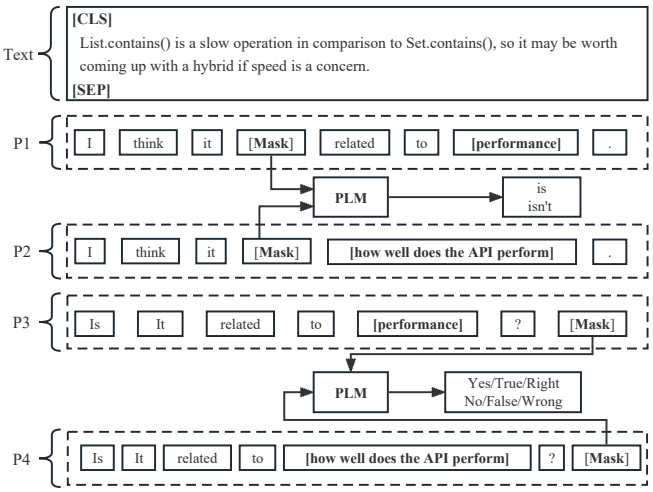

图 3 基于提示学习的评论分类示例

由于不同的提示模版对于提示学习效果存在影响，为了取得更好的分类表现，本文构造了图 3 所示的 4 种提示模版对 BERT 和 RoBERTa 模型进行调优，并选择表现最好的组合作为本文的评论分类模型。通过实验评估（见 3.1.1 节），本文发现采用 P4 模版的 RoBERTa 模型在评论分类任务中性能相对最好，因此使用此方法对 API 评论进行分类。

2.7 评论聚类

同一 API 在问答社区的大量评论中，存在高度相似且冗余的情况，开发者阅读这些相似的评论会浪费较多时间。为了缓解此问题，本文使用 Single-Pass 聚类算法对同一 API 下的同类评论进行聚类。Single-Pass 算法的核心思想是按照输入顺序将文本依次与已聚类的簇中心计算相似度，从而判断当前文本聚类到已有簇或是新簇。由于只需要要一次遍历即可完成聚类，因此该算法具有非常高效的处理效率。**算法 1** 为本文算法实现，该算法输入为待聚类的文本集合 *textMap* 和聚类采用的相似度阈值 *Threshold*，输出为聚类后的文本簇集合 *clusterMap*。该算法的核心思想是顺序读取 *textMap* 中的文本，针对每一个文本 *textId*，将其与当前簇集合 *clusterMap* 中所有簇中心计算相似度，并记录得到的最大相似度 *simMax* 及相应簇标签 *clusterId*。如果 *simMax* 大于定义的阈值 *Threshold*，则将 *textId* 存入 *clusterId* 对应的簇中，否则以当前文本作为簇中心向 *clusterMap* 中添加一个新的簇。

本文在聚类时相似度阈值设置为 0.8，相似度计算采用 Word2vec 模型，其原理是将文本中的单词作为特征并将其映射到 K 维向量空间，从而挖掘不同文本之间的相似性。为了获得更好的聚类效果，在计算相似度

前本文使用 NLTK 的停用词列表对文本中包含的停用词进行过滤。在簇中心更新时，本文将簇中长度最大的评论作为相应簇的中心。通过聚类处理，同一 API 具有高相似度的评论会聚集在一起形成独立的簇。

算法 1 Single-Pass 聚类算法

输入：输入的文本集合 *textMap*，
聚类的相似度阈值 *Threshold*
输出：聚类后的文本簇集合 *clusterMap*

```
function clustering (textMap, Threshold)
    clusterMap ← {} //定义输出的簇集合
    for textId in textMap.keys do
        if clusterMap is not empty then
            for clusterId in clusterMap.keys do
                //计算当前文本和所有簇中心的相似度
                semanticSim ← getSemanticSim (textId, clusterId)
                //记录最大相似度及相应簇标签
                simMax, clusterId ← getMax(semanticSim)
            end for
            if simMax > Threshold then
                clusterMap[clusterId].append(textId)
                update(clusterId) //更新簇中心
                continue //遍历下一条文本
            end if
        end if
        clusterMap[textId] ← [textId] //定义一个新簇
    end for
    return clusterMap
```

对于每个簇，本文只保留簇中心的评论，并将簇的大小（即该簇包含的评论数）记为该评论的得分作为参考。将聚类后的评论链接到知识图谱骨架（2.3 节构建）上，即可完成 API 评论知识图谱的构建。基于该知识图谱，来自不同帖子的评论可以通过 API 的结构关系或相似特性聚合在一起，如图 4 为 Stack Overflow 真实评论在 API 评论知识图谱中的结构示例。

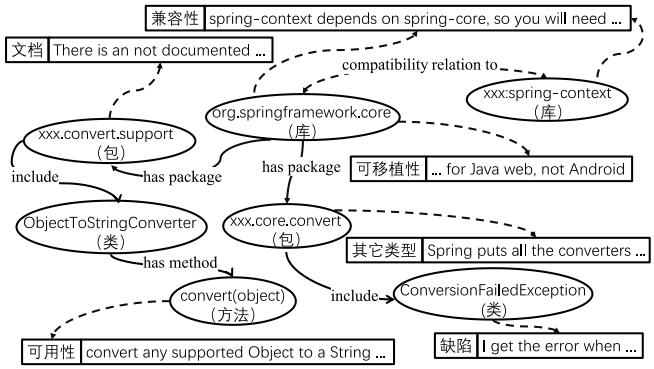

图 4 API 评论知识图谱示例

2.8 知识图谱构建实现

通过以上 API 评论知识图谱的构建方法，本文为 1000 个 Java 库构造了 API 评论知识图谱。具体地，在结构知识抽取环节，首先根据 Libraries.io 数据集（最后一次更新于 2020 年 1 月）获取了 Star 数排名前 1,000 的 Java 库的 groupId 和 artifactId 信息。并利用这些信息从 Maven 中央仓库下载相应库最新版本的 JAR 压缩包（截至到 2022 年 8 月）。然后，本文使用 ZipFile 从压缩包中提取所有第三方库的 Java 文件，将其静态解析后构建知识图谱骨架。在候选评论获取环节，本文获取了 Stack Overflow 备份的帖子文件（更新于 2022 年 6 月），并使用 SpaCy 对帖子进行代码块替换和句子切分后获取候选评论。接着，经过 API 实体识别、评论分类和聚类环节，将 API 评论链接到知识图谱骨架上，从而完成 API 评论知识图谱的构建。最终，本文构建的知识图谱共包含 3,185,819 个实体，**表 2** 列出了知识图谱中每种实体类型的统计情况。

表 2 API 评论知识图谱实体统计

实体类型	实体数	实体类型	实体数
库	1,000	可用性评论	34,621
包	8,636	安全性评论	1,857
类	176,114	缺陷评论	54
接口	13,007	社区评论	17
类属性	411,447	兼容性评论	95
方法	757,167	文档评论	3,158
参数	911,980	合法性评论	15,792
返回值	757,683	可移植性评论	156
API 元素	3,037,034	态度性评论	149
性能评论	3,544	其它类型评论	89,342

3 实　验

3.1 实验设计

为了评估本文方法在评论分类任务中的性能以及 API 评论知识图谱的有用性，本文设计并开展了两个评估实验。

3.1.1 分类模型性能表现

评估基准：Uddin 等人[4]在其工作中构建了一个人工标注的 API 评论分类数据集，为了降低数据标注带来的有效性威胁，本文将该数据集作为本实验的评估基准。该数据集由 1,338 个 Stack Overflow 帖子（带有 Java 标签）中的 4,522 个句子组成，其中每个句子都被人工标注了**表 1** 中所示的类别信息。其中，4,307 个句子标注了一个类别，209 个句子标注了两个类别，6 个句子标注三个及以上类别。

基线方法：本实验选择 BERT 和 RoBERTa 两个预训练语言模型作为基线方法。因为它们是目前 API 评

论分类任务效果最好的预训练模型[5]。为了对比不同提示学习模版对本文方法的影响，本文定义并使用 4 种提示模版（图 3 所示）进行性能比较。包括基础模版（P1）和 3 种基础模版的变体（P2-P4）。其中，P1 为简单陈述语句，P2 是将 P1 中的类别单词替换成更详细的解释（如：将 perfoamance 改为 how well does the API preform），P3 是将 P1 的陈述句式改为疑问句式，P4 是将 P2 的陈述句式改为疑问句式。为了便于区分，使用“基线模型-提示模版编号”对不同模型进行命名：如使用 P1 模版的 BERT 模型表示为 BERT-P1，本文的评论分类方法模型为 RoBERTa-P4。

实验设置：为了降低模型随机性带来的影响，本文采用 10 折交叉验证进行评估。为了验证提示学习在低资源条件下的表现，本文按照训练数据：校验数据：测试数据=1:1:8 的比例对基准数据进行分层采样，并采用样本均衡策略保证正负样本数相同。受预训练模型超参数设置的启发[14]，本文对所有模型训练的三个关键超参数，即学习率、批量大小和迭代轮次进行调优。其中，学习率设置为 5e-5、3e-5 和 1e-5，批量大小设置为 16 和 32，迭代轮次设置最大值为 5。本文选择分类任务中广泛使用的 F1、精确率和召回率三个指标对模型效果进行评估，并与之前的工作保持一致[5]，使用加权的方式计算相应指标。

表 3 API 评论分类方法性能对比

模型	F1 值(%)	精确率(%)	召回率(%)
BERT	77.6%	89.4%	70.2%
BERT-P1	89.7%	89.1%	90.5%
BERT-P2	89.6%	**89.5%**	89.8%
BERT-P3	89.3%	88.9%	89.8%
BERT-P4	89.3%	89.2%	89.6%
RoBERTa	74.5%	89.0%	66.8%
RoBERTa-P1	89.6%	89.2%	90.3%
RoBERTa-P2	89.4%	89.3%	89.7%
RoBERTa-P3	89.6%	89.0%	90.4%
RoBERTa-P4	**90.1%**	**89.5%**	**91.0%**

实验结果：错误!书签自引用无效。展示的是不同模型在所有评论类别下的平均分类效果（最佳指标已字体加粗）。实验结果显示，相对于基线方法，所有使用提示学习调优后的模型均有显著的效果提升。而本文使用的分类模型 RoBERTa-P4 在所有指标都达到最佳效果。具体而言，在保证精确率的同时，F1 值提升了 16.1%-21%，召回率提升了 30.6%-36.2%，实验结果表明本文提出的分类方法能更好的利用模型预训练阶段的先验知识，从而拥有更强的泛化能力。从实验结果中可以发现 RoBERTa 模型在使用提示学习调优后的效果提升更为明显，通过进一步分析，这是因为相对于 BERT 模型，RoBERTa 模型在预训练时删除了 NSP 任务，并优化了 MLM 任务中的单词屏蔽模式，从而使该模型更接近本文通过提示学习转化后的屏蔽单词补全任务。

表 4 展示的是本文方法对于不同评论类别的性能表现。可以发现本文方法在大部分评论类别的划分上均有出色的表现。在所有评论类别中，可用性评论和其它类型评论的分类表现相对较差。通过对训练数据进行分析，本文发现这是因为这两个类型的含义较为广泛，从而导致模型较难归纳相应类型的特征。

表 4 本文方法对于不同评论类别性能

模型	F1 值(%)	精确率(%)	召回率(%)
性能评论	89.9%	89.1%	91.5%
可用性评论	69.1%	69.5%	68.8%
安全性评论	94.6%	92.9%	96.4%
社区评论	97.0%	96.0%	98.0%
兼容性评论	97.0%	96.0%	98.0%
可移植性评论	97.7%	97.0%	98.5%
文档评论	91.6%	91.1%	92.2%
漏洞评论	93.8%	92.6%	95.6%
合法性评论	98.4%	97.8%	98.9%
态度性评论	90.5%	89.9%	91.2%
其它类型评论	72.1%	72.5%	71.9%

3.1.2 知识图谱所含知识有用性

为了评估本文构建的 API 评论知识图谱所含评论知识的有用性，本文从中随机选取 12 个不同粒度的 API（其中库、包、类和方法各取 3 个，选取标准为包含 5 条以上评论的 API）。然后邀请了 4 名参与者（均有三年以上 Java 开发经验的硕士在读研究生）通过阅读相应 API 包含的评论，对每一个 API（包含评论）的有用性和可读性给出 1-4 的评分（1 代表不同意、2 代表比较不同意、3 代表比较同意、4 代表非常同意），因此本实验共获得有用性和可读性得分各 48 个。

通过对实验结果进行分析，在所有打分情况中有 87.5%的有用性得分为 3 及以上，有 94%的可读性得分为 3 及以上。通过对有用性得分较低的 API 进行分析，本文发现这是由于将帖子内容按照句子级别切分后，会破坏上下文有紧密联系的句子表述。而本文之所以按照句子级别切分，是为了避免评论文本在链接到知识图谱时发生偏移。未来本文将会在语义完整性和链接准确性之间进行更合适的权衡。总体而言，该实验证明了 API 评论知识图谱所含评论知识具有良好的有用性和可读性。

3.2 有效性威胁

本文对所提出的方法进行了有效性威胁分析。

内部威胁：本文的一个主要内部威胁是在评估不同分类模型性能的实验中，模型的训练存在随机性，且会受到超参数的影响。为了降低此威胁，本文对 10 种模型在 11 个评论类别上均采用 10 折交叉验证，并在每一折中对模型的学习率、批量大小和迭代轮次三个关键超参数进行调优，即累计进行了 10×11×10×3×2×5=33000 次调优（见 3.1.1 节-实验设置）。此外，本文在评论分类任务上的性能可能会由于人工定义的提示模版策略而存在偶然性。因此，本文设置了 4 个不同类型的模版与基线方法进行对比，从而降低本文方法的有效性威胁。

外部威胁：本文的实现聚焦于 Java 语言的 API 评论知识图谱构建，因此存在编程语言泛化性的威胁。实际上，本文提出的知识图谱架构（图 2 所示）并不特定于 Java 语言，只需要修改结构知识抽取环节（见 2.3 节）中的静态解析方法，即可为其它编程语言构建 API 评论知识图谱。

3.3 实验总结

本文通过实验证明了所提方法在评论分类任务上的有效性以及所建知识图谱包含评论知识的有用性。本文方法进行评论分类时相对于目前最先进的基线方法在 F1 指标上提升了 16.1%-21%，召回率提升了 30.6%-36.2%。本文构建的知识图谱所含 API 评论知识具有较高的有用性和可读性，能够有效帮助开发者学习相关 API 知识。

4 结 语

本文提出了一种基于提示学习的 API 评论知识图谱构建方法，该方法能够在低资源条件下对 API 评论进行有效分类。此外，本文构建的知识图谱能够按照不同粒度、不同类别、去冗余地将不同帖子中的 API 评论聚合到一起。基于该方法，本文构建了一个包含 3,185,819 个实体的 API 评论知识图谱，并且通过实验证明了该知识图谱能够有效帮助开发者学习 API 知识。

参 考 文 献

[1] Mohagheghi P, Conradi R. Quality, productivity and economic benefits of software reuse: a review of industrial studies[J]. Empirical Software Engineering, 2007, 12: 471-516.

[2] Treude C, Robillard M P. Augmenting API documentation with insights from Stack overflow[C]//Proceedings of the 38th International Conference on Software Engineering. 2016: 392-403.

[3] Tang H, Nadi S. On using Stack Overflow comment-edit pairs to recommend code maintenance changes[J]. Empirical Software Engineering, 2021, 26(4): 68.

[4] Uddin G, Baysal O, Guerrouj L, et al. Understanding how and why developers seek and analyze API-related opinions[J]. IEEE Transactions on Software Engineering, 2019, 47(4): 694-735.

[5] Yang C, Xu B, Khan J Y, et al. Aspect-Based API Review Classification: How Far Can Pre-Trained Transformer Model Go?[C]//2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE, 2022: 385-395.

[6] Liu P, Yuan W, Fu J, et al. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing[J]. ACM Computing Surveys, 2023, 55(9): 1-35.

[7] Helmeczi R K, Cevik M, Yıldırım S. A Prompt-based Few-shot Learning Approach to Software Conflict Detection[J]. arXiv preprint arXiv:2211.02709, 2022.

[8] 罗贤昌, 薛吟兴. 基于 BERT 的提示学习实现软件需求精确分类[J]. 信息技术与网络安全, 2022.

[9] Chai Y, Zhang H, Shen B, et al. Cross-domain deep code search with few-shot meta learning[J]. arXiv preprint arXiv:2201.00150, 2022.

[10] 张宁豫, 谢辛, 陈想, 等. 基于知识协同微调的低资源知识图谱补全方法[J]. 软件学报, 2022, 33(10): 3531-3545.

[11] 邢双双, 刘名威, 彭鑫. 基于软件知识图谱的代码语义标签自动生成方法[J]. 软件学报, 2021, 33(11): 4027-4045.

[12] 马展, 王岩, 王微微, 等. 基于多源信息融合的 API 知识图谱构建[J]. 计算机系统应用, 2021, 30(12): 202-210.

[13] Liu Y, Liu M, Peng X, et al. Generating concept based API element comparison using a knowledge graph[C]//Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering. 2020: 834-845.

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

[15] Liu Y, Ott M, Goyal N, et al. Roberta: A robustly optimized bert pretraining approach[J]. arXiv preprint arXiv:1907.11692, 2019

联系方式

本文负责人：杨延军；手机号码：13395458480；E-mail：yangyj20@fudan.edu.cn。

杨延军，硕士生；主研领域：智能化软件开发；身份证号：140603199411170519；手机号码：13395458480；单位：复旦大学计算机科学技术学院；通信地址：上海市杨浦区复旦大学江湾校区淞沪路 2005 号交叉二号楼；邮政编码：200438；E-mail：yangyj20@fudan.edu.cn。

刘名威，博士，主研领域：智能化软件开发；手机号码：15221926558；单位：复旦大学计算机科学技术学院；通信地址：上海市杨浦区复旦大学江湾校区淞沪路 2005 号交叉二号楼；邮政编码：200438；E-mail：liumingwei@fudan.edu.cn。

彭鑫，教授，主研领域：软件开发大数据分析、智能化软件开发、云原生与智能化运维、泛在计算软件系统；手机号码：13916819965；单位：复旦大学计算机科学技术学院；通信地址：上海市杨浦区复旦大学江湾校区淞沪路 2005 号交叉二号楼；邮政编码：200438；E-mail：pengxin@fudan.edu.cn。

赵文耘，教授，主研领域：软件工程、软件开发工具及其环境、企业应用集成（EAI）；手机号码：13601756188；单位：复旦大学计算机科学技术学院；通信地址：上海市杨浦区复旦大学江湾校区淞沪路 2005 号交叉二号楼；邮政编码：200438；E-mail：wyzhao@fudan.edu.cn。

修改说明

非常感谢老师的耐心审阅！本文确实存在如评审意见中指出的不足之处。目前本文已针对老师提出的每个问题进行修改完善，下面对修改情况进行简要阐述：

问题 1：摘要中“所建知识图谱具有良好的有用性和可读性”，“有用性”、“可读性”是否不太准确？

原文确实存在表述不准确的问题。本文已将摘要处表述修改为：

“相对于基线方法，该方法的 F1 指标提升 16.2%-21%，召回率提升 30.6%-36.2%。此外，所提出的知识图谱结构可以将不同帖子中的 API 评论按照不同粒度、不同类别、去冗余地整合在一起。并且通过实验证明所建知识图谱能够有效帮助开发者学习相关 API 知识。”

此外，由于该表述是对文中实验二结论的概述，因此本文将实验二以及与实验二相关的表述进行同步修改。修改后实验二的总结为：

“本文通过实验证明了所提方法在评论分类任务上的有效性以及所建知识图谱包含评论知识的有用性。本文方法进行评论分类时相对于目前最先进的基线方法在 F1 指标上提升了 16.1%-21%，召回率提升了 30.6%-36.2%。本文构建的知识图谱所含 API 评论知识具有较高的有用性和可读性，能够有效帮助开发者学习相关 API 知识。”

问题2:引言研究背景和应用背景、研究意义说明不足。

对于原文存在的研究背景及研究意义说明不足的问题，本文已重新组织逻辑表述，并引入相关参考文献补充说明文本的应用背景和研究意义。修改后的部分表述如下：

“针对某一个特定的API而言，相关的评论往往零散存在于不同帖子中，并不利于开发者在特定场景下进行学习了解。因此开发人员通常要根据特定开发需求分析和总结相应评论知识[4]。目前存在一些对API评论进行分类的研究工作[3-5]，然而这些工作通常基于机器学习算法或大规模预训练语言模型进行调优，均需要使用大量人工标注数据以一种监督的方式对模型进行训练，存在较高实现成本。此外，现有工作对于API评论知识的组织形式并不利于将多源知识进行关联。

为了解决以上问题，本文提出了一种基于提示学习的API评论知识图谱构建方法，可以在低资源条件下对API评论进行有效的类别划分，从而帮助开发人员更加高效且有选择性的利用API评论知识。相对于基线方法，

本文方法在F1指标上提升了16.1%-21%，召回率指标上提升了30.6%-36.2%。本文通过实验证明了所建知识图谱包含的API评论知识具有较高的有用性和可读性，能够有效帮助开发人员学习相关API知识。”

问题3:第1.3节现有的软件工程领域知识图谱构建对本文有哪些借鉴意义？本节内容重点介绍API相关的知识图谱，是否修改节标题更合适？

本文工作确实借鉴了软件工程领域知识图谱构建的相关研究工作，现已突出相关工作对本文研究工作的启发，并补充了最新的参考文献。此外，本文将节标题修改为：“**API 知识图谱构建及应用**”。修改之后的 1.3 节表述为：

“由于知识图谱的结构能够更好的将多源异构的知识组织在一起。近年来，研究人员为不同类型知识和不同目的构造了软件工程领域的知识图谱，从而实现对多源知识的高效利用。由于现代软件开发中 API 的广泛应用，一些研究人员构建了 API 知识图谱。例如，邢双双等人[11]利用 API 参考文档和开发问答文本中的概念和关系构造了一个软件知识图谱，基于该知识图谱可以实现为给定的代码片段添加语义标签，以辅助开发人员快速了解代码含义。马展等人[12] 通过挖掘 API 参考文档和 Stack Overflow 中问答帖子的知识,构建了一个 API 知识图谱, 以实现更准确的 API 推荐。Liu 等人[13]提出了一种基于知识图谱的 API 比较法，该方法能够将 API 参考文档中的知识构建为 API 知识图谱从而对 API 进行不同方面的比较。

受以上工作启发，为了更好的整合具有结构关系或者相似特性的 API 评论知识，本文将 Stack Overflow 中的 API 评论构建为知识图谱。”

问题4:第2节对Single-Pass聚类算法补充说明。

原文对于评论聚类环节使用的 Single-Pass 算法介绍不够充分，现已从核心思想、实现原理和具体实现细节三个方面对该算法进行补充说明。补充后的算法介绍为：

“Single-Pass 算法的核心思想是按照输入顺序将文本依次与已聚类的簇中心计算相似度，从而判断当前文本聚类到已有簇或是新簇。由于只需要要一次遍历即可完成聚类，因此该算法具有非常高效的处理效率。**算法 1** 为本文算法实现，该算法输入为待聚类的文本集合 *textMap* 和聚类采用的相似度阈值 *Threshold*，输出为聚类后的文本簇集合 *clusterMap*。该算法的核心思想是顺序读取 *textMap* 中的文本，针对每一个文本 *textId*，将其与当前簇集合 *clusterMap* 中所有簇中心计算相似度，并记录得到的最大相似度 *simMax* 及相应簇标签 *clusterId*。如果 *simMax* 大于定义的阈值 *Threshold*，则将 *textId* 存入 *clusterId* 对应的簇中，否则以当前文本作为簇中心向 *clusterMap* 中添加一个新的簇。”

问题5:进一步完善结语。

本文已对包含结语在内的全文表述进行完善。完善后的结语为：

“本文提出了一种基于提示学习的 API 评论知识图谱构建方法，该方法能够在低资源条件下对 API 评论进行有效分类。此外，本文构建的知识图谱能够按照不同粒度、不同类别、去冗余地将不同帖子中的 API 评论聚合到一起。基于该方法，本文构建了一个包含 3,185,819 个实体的 API 评论知识图谱，并且通过实验证明了该知识图谱能够有效帮助开发者学习 API 知识。”