计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.191200010

融合领域知识的 API 推荐模型

李 浩 钟 声 康 雁 李 涛 张亚钏 卜荣景
云南大学软件学院 昆明 650504
(12018002170@mail.ynu.edu.cn)

摘 要 应用程序接口(Application Programming Interfaces,API)在现代软件开发中起着重要的作用,开发人员经常需要为他们的编程任务搜索合适的 API。但是随着信息产业的发展,API 参考文档变得越发庞大,传统的搜索方式会因为互联网上的冗余和错误信息给工程师的查询带来不便。与此同时,由于编程任务的自然语言描述与 API 文档中的描述之间存在词汇和知识上的差距,很难找到合适的 API。基于这些问题,提出一种融合领域知识的 API 推荐算法 ARDSQ (Recommendation base on Documentation and Solved Question)。ARDSQ 能够根据工程师对某个功能的自然语言描述去知识库里检索到最为贴近的 API。实验表明,与两种先进的 API 推荐算法(BIKER,DeepAPILearning)比较,ARDSQ 在推荐系统关键评价指数(Hit-n,MRR,MAP)上都有较大的优势。

关键词:应用编程接口;信息检索;代码推荐;程序分析;深度学习
中图法分类号 TP311.5

API Recommendation Model with Fusion Domain Knowledge

LI Hao,ZHONG Sheng,KANG Yan,LI Tao,ZHANG Ya-chuan and BU Rong-jing
College of Software,Yunnan University,Kunming 650504,China

Abstract Application Programming Interfaces (API) play an important role in modern software development,and developers often need to search for the appropriate API for their programming tasks. However,with the development of the information industry,API reference documents have become larger and larger,and traditional search methods have also caused inconvenience to engineers' queries because of redundant and erroneous information on the Internet. At the same time,due to the vocabulary and knowledge gap between the natural language description of programming tasks and the description in the API documentation,it is difficult to find a suitable API. Based on these issues,this paper proposes an algorithm called ARDSQ (Recommendation base on Documentation and Solved Question) which is an API recommendation algorithm that integrates domain knowledge. ARDSQ can retrieve the closest API in the knowledge base based on the natural language description given by the engineer. Experiments show that,compared with two advanced API recommendation algorithms(BIKER,DeepAPILearning),ARDSQ has greater advantages in the key evaluation index (Hit-n,MRR,MAP) of the recommendation system.

Keywords API,Information retrieval,Code recommendation,Program analysis,Deep learning

1 引言

在现代软件开发中,软件库提供的应用程序接口给软件开发带来了极大的方便。然而,要熟悉大型库中的所有 API 并不容易。以 Java 编程语言为例,在 1996 年 Java 1.0 发布的时候,其 Java Development Kit (JDK)包含了 211 个类和接口。在 2014 年 Java 8.0 发布的时候,其 JDK 包含了 4240 个类和接口[1],超过 140000 个类和方法[2]。18 年的时间,类和接口的数量增长了 20 倍。在实际开发过程中,除了使用标准库,开发人员还会用到大量的第三方库,在这样的现状下,开发人员一般会使用搜索引擎和查阅官方 API 文档两种办法来定位所需 API。但是,这两种办法各有不足。目前,搜索引擎基本都是通用搜索引擎,而 API 推荐是一个封闭域的需求。这样的冲突就造成了在使用搜索引擎搜索 API 的过程中,会出现很多质量不高、无关甚至是错误的结果。使用查询 API 文档的方法来定位 API 固然可以得到相关度和质量都非常高的结果,但是因为 API 文档往往十分巨大,所以在查询的过程中会耗费开发人员大量的时间和精力,而且由于某个问题的解决经常是以多个 API 共同组合完成,因此单纯依靠官方 API 会使得整个开发过程变得非常低效[3]。

为了解决上述存在的问题,许多代码推荐系统被开发了出来。不同于一般的搜索引擎,代码推荐系统主要是为了解

基金项目:国家自然科学基金(61762092,61762089);云南省软件工程重点实验室开放基金项目(2017SE204);材料基因工程-基于 Metcloud 的集成计算功能模块计算软件开发(2019CLJY06)
This work was supported by the National Natural Science Foundation of China (61762092,61762089),Yunnan Provincial Key Laboratory of Software Engineering Open Fund Project (2017SE204) and Development of MatCloud-based High-throughput Computational Module for MGI Platform(2019CLJY06).
通信作者:康雁(kangyan@ynu.edu.cn)

决代码搜索的问题，在代码推荐系统中，有对代码段进行推荐的，也有对具体某个 API 进行推荐的。本文主要研究 API 推荐系统。

目前 API 推荐系统主要有两类：信息检索模型和深度学习模型。起初，API 推荐系统一直被信息检索模型所统治，主要有 APIBot[4]，RACK[5]，RASH[6]，BIKER[7]等，它们使用的技术主要有 TF-IDF[8]，bag-of-words[9]等。近些年来，随着深度学习的飞速发展，很多新的技术和模型被提出来，这些技术和模型被用在了 API 推荐系统中，从而出现了基于深度学习的 API 推荐系统，主要有 DeepAPILearning[10]，CODEnn[11]，NLP2Code[12]等，它们使用的核心算法是循环神经网络（Recurrent Neural Network，RNN）[13]。上述系统都在 API 推荐领域取得了良好的效果，但是依然存在一些问题，一是 API 推荐的精确度不足，二是没有更好地利用现有的领域知识库来解决新的 API 查询问题。

针对上述的问题，本文提出了一种融合领域知识的 API 推荐模型。本文的主要贡献有以下几点：

（1）有别于上述提到的基于信息检索的系统和基于深度学习的系统，本系统把信息检索技术和 BERT[14]深度学习语言模型结合在一起，提高了 API 推荐的准确度。

（2）提出了一种新的未知问题与已知问题的匹配方法，不同于现在最为主流的基于 word2vec[15]和 bag-of-words 的方法，该方法将整个句子转换为句向量，并基于句向量进行比较，能准确地匹配词相似度外更为深层次的语义。

（3）提出了一种从 StackOverflow 文档中提取 API 的方法，相较于传统的方法，本方法利用了文档之间的内在联系，比传统的方法 RASH[7]拥有更广的适用范围，并在一定程度上解决了所提取的 API 的噪音问题。

2　相关工作

2.1　基于 API 文档的直接推荐模型

早期的 API 推荐系统大多是利用传统的信息检索技术来进行问题与 API 文档的直接匹配和推荐。以 APIBot 为例，其主要利用 TF-IDF 和正则表达式等传统方法来匹配问题与文档。但是由于 API 文档主要是对某个 API 功能进行描述，而开发人员提的问题大多是目的描述，即单纯描述需要达到的效果，例如“How do I generate random integers within a specific range in Java?”，这样的问题和文档很难做到匹配度比较高的匹配，所以研究人员开始尝试利用其他的匹配方法而不是直接的问题和文档匹配。

2.2　基于 API 文档的间接推荐模型

由于问题与 API 文档之间存在内在的差异，研究人员开始寻找一个中间量来弥补问题与 API 文档之间的这种差异，比较典型的有 RASH，RACK 等模型。这类模型主要将开发者提出的问题与已经解决的 Stack Overflow 问题进行配对，然后将配对成功的 API 检索结果返回。该方法取得了良好的效果，但其中有两个难点对结果的影响比较大。第一个难点是未知问题与已知问题的匹配，现在最为主流的方法是基于 word2vec，将问题转化为一串词向量来进行匹配，这种方法可以很好地解决这类问题，即两个问题中的词相似性。因为 word2vec 的特性，相似的词会有比较高的匹配度，所以对于该类问题，会有比较高的匹配度，但是无法匹配更深层次的语义，这也是目前亟需解决的一个问题。第二个难点是对已知问题中的 API 的抽取，现在主要有两个方法来对已知问题下的 API 进行抽取。一是直接匹配答案中的 API，这种方法的优点是基本能把正确的 API 抽取出来；缺点是抽取出来的 API 中会有很多噪音，因为答案中很少有只提及正确的 API 的，会有很多其他的 API 掺杂其中，这些噪音 API 会很大程度地影响最终系统的 MRR 等系数。二是利用正则表达式来匹配回答中指向 Java 文档的超链接[6]，这种方法的优点是抽取出的 API 非常纯净，没有噪声；缺点是绝大部分回答中不包含指向 Java 文档的超链接，这样构建出来的知识库就会非常小，从而影响最终的效果。

2.3　基于深度学习的 API 推荐模型

在 API 推荐领域，一直没有一个比较权威而且数据量较大的数据集，导致了神经网络无法训练，从而使得深度学习的技术直到 2016 年才应用在了 API 推荐中[8]，该系统收集了 GitHub 上面 2008 年到 2014 年的星数大于 1 的 442 928 个 Java 项目。该系统将代码中注释的第一句话作为问题，把相应的代码解析回 AST 并抽取相应的 API，构成了{问题，API}对。该类型的系统特点是利用庞大数据量提高了准确度；缺点和直接基于 API 文档的推荐系统相似，因为代码注释还是对某个 API 进行功能性的描述，与开发人员的问题匹配度不高，但是巨大的数据量使得其精确度相较于直接基于 API 文档的推荐系统有所提高。

3　ARDSQ 模型

本文提出的 ARDSQ 主要用在 API 推荐。ARDSQ 读取开发者输入的一段自然语言描述，然后经过系统处理后输出相应的 API。本文提出的模型主要分为 3 个部分：第一部分是建立知识库，该知识库将用在以后的相似度分析、API 提取和 API 排序等方面；第二部分是相似问题的搜索；第三部分是对提取出来的 API 进行排序。本系统架构如图 1 所示。

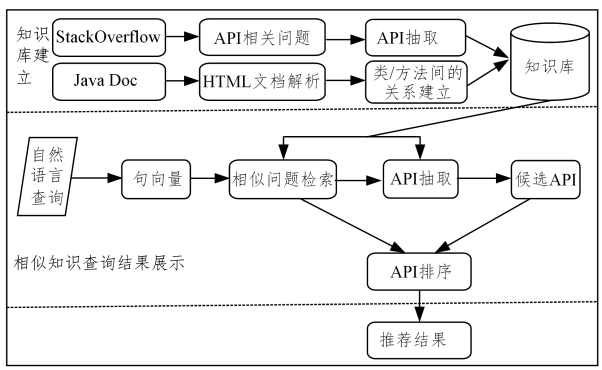

图 1　ARDSQ 模型

Fig. 1　ARDSQ model

3.1　建立知识库

第一步基于 Stack Overflow 和 Java API 文档构建知识库。对于 Java API 文档部分，我们首先将 API 提取出来，主要提取的是类和接口，还有类与类之间的关系，接着将每个类和接口下面的函数原型和函数描述也提取出来。类或接口和其中含有的函数原型构成了{类/接口，函数原型}这样的键值对，函数原型和对应的函数文档构成了{函数原型，函数文档}这样的键值对，这个 API 知识库是 Stack Overflow 知识库的基础。

Stack Overflow是一个综合性的技术社区，上面有各种类型的IT相关的问题，在该系统中，我们仅提取Java相关的问题。首先通过Stack Overflow上的标签，筛选出带有Java标签的问题，但这样筛选出的问题存在很多问题。例如，有的问题本身质量不高，主要体现在问题描述本身并不明确，还有些问题并没有答案。上述问题都不能被该系统利用，所以还需要对筛选出来的Stack Overflow上带Java标签的问题做进一步的过滤。在该模型中，我们仅提取问题得分高于1分并包含已接受答案的问题。接下来，我们要把每个问题对应的答案中的API实体抽取出来。这一步有两个目的，一是过滤掉概念性的问题，因为Stack Overflow上的问题并不全是API相关的问题，其中还包含很多概念性的问题，这些问题会对我们系统的最后结果造成干扰；二是将每个问题对应答案中的API提取出来以后，就可以做到一次处理，多次使用。相较于现在主流的BIKER[8]模型，这样的处理方法避免了对同一问题所包含API的反复提取，对系统整体性能有一定程度的提高。提取API分为3个步骤，(1)使用上一步构建出来的API知识库，对类级别的API实体进行抽取，在该过程中会抽取出很多不相关的API；(2)对第一步中抽取出来的API数据进行去噪，通过与当前问题linked的问题计算各个类级API的出现频率，然后将类级API相应的API文档的描述与问题进行文本匹配，完成类级API的排序；(3)将第二步得到的类级API下含有的函数原型通过API知识库提取出来，在回答中进行第二次匹配，不同于上次的类级API匹配，这次匹配主要是匹配函数原型。通过上述步骤，我们就建立了{问题，API}键值对，这些键值对构成了Stack Overflow知识库。

3.2 基于BERT模型的句向量生成器

API推荐系统存在数据少的问题，因此从无标注数据中挖掘有用数据就变得非常必要。在自然语言处理中，一个最直接的有效利用无标注数据的任务就是语言模型，因此很多任务都使用了语言模型作为预训练任务。

BERT[14]并不是第一个无监督的语言模型，相较于其他的语言模型(例如只有三层的Elmo[15])，BERT结合了深度模型和无监督两个特点。BERT的深度体现于使用了transformer[16]作为解码器，transformer已经在机器翻译任务上取得了很大的成功，并且可以做得非常深，在本文中用到的预训练的模型有24层，3亿多个参数。BERT的无监督体现在训练过程，BERT使用了BooksCropus与英文维基百科共33亿个词的数据来训练BERT模型。

3.3 API匹配与排序

3.3.1 API匹配

API匹配部分的目的是找出与问题描述最接近的已知的知识。在进行匹配之前，我们首先使用BERT将自然语言描述的问题转化为一个长度固定的句向量，用来和已知的知识进行匹配。

$$score=\left|\frac{\sum_{i=1}^{q}q_i * p_i}{|\boldsymbol{p}|}\right| \tag{1}$$

其中，$\boldsymbol{q}$为开发者输入的一个未知问题对应的句向量，$\boldsymbol{p}$为一个已知的Stack Overflow上的问题的问题描述的句向量，q_i为句向量中的一个元素，$|\boldsymbol{p}|$为当前已知问题描述的句向量的模。不同于一般的余弦值计算，本系统只用$\boldsymbol{p}$向量的模做归一化运算，其原因是$\boldsymbol{q}$向量在该次推荐中一直不变，因此不影响最终得分，减少了运算量，并且防止了*score*的区间落在[0,1]。因为多个*score*可能十分接近，如果把*score*区间压缩在[0,1]区间，可能会因为浮点数在计算机中的不精确存储而造成系统误差。

3.3.2 API排序

在得到与问题最相关的前K个知识以后，下一步是对这些知识中含有的API进行排序，计算API与当前问题的相关度，以此来确定API的顺序。

$$SimilarityScore=\left|\frac{\sum_{i=1}^{q}q_i * c_i}{|\boldsymbol{c}|}\right| * \log_2 fre_c + \left|\frac{\sum_{i=1}^{q}q_i * m_i}{|\boldsymbol{m}|}\right| * \log_2 fre_m \tag{2}$$

其中，$\boldsymbol{c}$为类文档的文档描述对应的句向量，c_i为句向量中的一个元素，$|\boldsymbol{c}|$为当前已知问题描述的句向量的模，$\boldsymbol{m}$为方法文档的文档描述对应的句向量，m_i为句向量中的一个元素，$|\boldsymbol{m}|$为当前已知问题描述的句向量的模，fre_m代表的是方法$\boldsymbol{m}$在待排序API中出现的频率，fre_c代表的是类$\boldsymbol{c}$在待排序API中出现的频率。本系统将API与问题的关联主要分为了两个部分，一是问题与文档的相似程度，二是API出现的频率，因为如果某个API在多个待选知识中都被提及，那么我们可以认为它和该问题的关联程度是较高的。不过如果直接用出现频率与文档相似度相乘，则会大幅度削弱文档相似度对整个评分的影响，让API排序转换为API频率排序，故对其用底为2的指数函数进行限制，让API频率对整个评分的影响控制在一定范围内。在完成上述匹配和排序后，根据SimilarityScore对候选的API进行排序，最后抽取前K个API返回给开发者。本系统默认的模式是返回方法级的API，如果需要返回类级的API，只需把SimilarityScore改为如下定义：

$$SimilarityScore=\left|\frac{\sum_{i=1}^{q}q_i * c_i}{|\boldsymbol{c}|}\right| * \log_2 fre_c \tag{3}$$

这样，在API排序过程中就只考虑类级文档和问题的相似度，从而完成了类级API推荐的排序工作。

4 实验结果和分析

4.1 实验数据

本次实验使用了BIKER和RASH公布的数据进行测试，其中BIKER公布的数据含有413条函数级的问答对，RASH公布的数据含有1234条类级的问答对。

4.2 评测指标

在本次实验中，我们采用Hit@K，MRR@K，MAP@K 3个指标来评估整个系统，并且分别将其与近期先进的两个开源模型DeepAPILearning和BIKER模型进行对比。在这3个指标中，Hit@K测量的是系统推荐的API的正确性，而MRR@K和MAP@K测量的是系统所推荐的API的顺序的质量。在API推荐系统中，所推荐的API的正确性是很关键的，但除了给出正确的推荐，推荐的先后次序也很重要，如果能把正确的API尽量放在前面，能很大程度上节省开发者的时间，提高工作效率。Hit@K测量的是在仅推荐K个API的情况下，能被正确解决的问题的个数占总问题数的比例。

$$Hit@K=\frac{1}{|Q|}\sum_{i=1}^{Q}\frac{\sum_{j=1}^{Q_i}rel(j)}{|Q_i|} \tag{4}$$

其中，Q 代表问题的集合，$|Q|$ 代表问题的总数；$rel(i)$ 是一个二元函数，当在第 i 个位置上的 API 是正确的 API 时，$rel(i)$ 的值为 1，否则为 0；Q_i 为第 i 个问题，$|Q_i|$ 为第 i 个问题正确的 API 的数量。

MRR@K 测量的是推荐结果的顺序的优劣，正确的答案在推荐结果中位置越靠前，结果越好。具体来说，对于一条查询，如果某个正确的答案排在第 n 位，则 MRR 的得分就是 $1/n$。

$$MRR@K=\frac{1}{|Q|}\sum_{j=1}^{Q}\frac{1}{pos_j} \tag{5}$$

其中，pos_j 是第 j 个问题第一个正确答案的位置。

MAP@K 同样测量推荐结果的顺序的优劣，它也是一个在信息检索系统里面非常常用的指标。不同于 MRR 只关心第一个出现的正确答案的位置，MAP 测量的是所有正确答案在推荐结果中的位置关系，因为对于一个 API 相关的查询，有时正确的 API 不止一个，对于存在多个正确 API 的查询，使用 MAP 指标能反映所有正确 API 在查询结果中的位置。

$$MAP@K=\frac{1}{|Q|}\sum_{j=1}^{Q}\frac{\sum_{j=1}^{Q}P(i)*rel(i)}{|C_j|} \tag{6}$$

$$P(i)=\frac{\sum_{k=1}^{i}rel(k)}{i} \tag{7}$$

其中，C_j 是第 j 个问题的正确答案的集合，$|C_j|$ 是第 j 个问题对应正确答案的集合；$P(i)$ 代表的是前 i 个推荐结果的正确比例。

4.3　实验结果

为了测量整个系统的性能，我们选取了两个数据集，一个是 BIKER 公布的数据集，它含有 413 条函数级的问题和相应的正确的 API；另一个是 RASH 公布的数据集，它含有 1 234 条类级的问题和相应的正确的 API，分别用来测量整体系统的函数级推荐和类级推荐。在整个实验中，我们仅对推荐结果的前 5 个进行考量。

表 1 列出在函数级推荐下本文模型的质量。这里我们使用了两个基准方法作为对比，其中 DeepAPI 主要使用的是深度学习技术来推荐 API，DeepAPI 使用一个基于 RNN 的 encoder-decoder 将问题映射到一个固定长度的向量上，然后基于这个向量把 API 方法的序列生成出来。BIKER 主要使用的是传统信息检索的方法（如 TF-IDF 等）来推荐 API。可以看出相比于 DeepAPI 和 BIKER 在 BIKER 数据集下推荐结果的质量，我们的模型在 MRR，MAP 和 HIT 指标上均有提升。其中，由于 DeepAPI 系统网页失效，因此无法测量 Hit@5 值。

表 1　BIKER 数据集下函数级的推荐质量

Table 1　Recommended quality at method level under BIKER dataset

Model	MRR@5	MAP@5	Hit@5
BIKER	0.573	0.521	0.600
DeepAPI	0.188	0.153	—
ARDSQ	0.665	0.622	0.689

表 2 列出在 RASH 数据集下类级的推荐准确度，可以看出，我们模型的效果也有一定程度的提升。

表 2　RASH 数据集下类级的推荐质量

Table 2　Recommended quality at class level under RASH dataset

Model	MRR@5	MAP@5	Hit@5
BIKER	0.342	0.541	0.386
ARDSQ	0.445	0.685	0.483

表 3 和表 4 综合比较了在一些具体的 StackOverflow 问题上，BIKER 和 ARDSQ 的推荐准确度和正确度。

表 3　用例研究

Table 3　Case study

PID	StackOverflow ID	Query	Answers
Q1	363681	How do I generate random integers within a specific range in Java?	java. util. Random. nextInt
Q2	415953	How can I generate an MD5 hash?	java. security. MessageDigest. getInstance
Q3	2885173	How do I create a file and write to it in Java?	java. io. PrintWriter. println java. nio. file. Files. write
Q4	5585779	How do I convert a String to an int in Java?	java. lang. Integer. parseInt
Q5	3481828	How to split a string in Java	java. lang. String. split
Q6	157944	Create ArrayList from array	java. util. Arrays. asList

表 4　正确度比较

Table 4　Correctness compare

Model	Q1	Q2	Q3	Q4	Q5	Q6	Average
BIKER	1.00	0.25	0.52	0.67	0.25	0.33	0.50
ARDSQ	1.00	0.88	0.70	0.95	1.00	1.00	0.92

结束语　本文针对现有的 API 推荐系统存在的不足，提出一种深度学习和信息检索融合的系统。BERT 预训练语言模型通过双向 Transformer 结构动态生成句的上下文语义表示，相比于传统的 word2vec 模型更能表征两个句子的深层语义联系。该模型优于目前最优的 BIKER 模型，提升了 API 推荐系统的效果。

另外，当提出的问题在知识库中没有相似问题时，该系统推荐结果的准确率会大幅下降；而且所推荐的函数级 API 中无法精确到参数，但是函数重载又是面型对象语言中普遍存在的，这也是现在主流 API 推荐系统都存在的问题。

下一步的研究方向是将该系统拓展为一个通用 API 系统，不再限于 Java 语言，并且增加多个信息源，做到对未知问题的尽可能覆盖；增加实时爬虫对信息源进行爬取，保证对最新的问题有对应的知识；引入开源代码库，利用现有的代码辅助 API 推荐，解决函数重载的问题。

参 考 文 献

[1] CAY S. Horstmann and Gary Cornell. Core java Volume 1 Foundations[M]. Prentice-Hall, 2018: 12.

[2] YU H, SONG W, MINE T. Apibook: an effective approach for finding APIs[C]// Asia Pacific Symposium on Internetware. 2016: 45-53.

[3] CHEN Z Z, JIANG R H, PAN M X, et al. Empirical Study of Code Query Technique Based on Constraint Solving on StackOverflow[J]. Computer Science, 2019, 46(11): 137-144.

[4] TIAN Y, THUNG F, SHARMA A, et al. Apibot: Question answering bot for api documentation[C]// 32nd IEEE/ACM International Conference on Automated Software Engineering. 2017: 153-158.
[5] RAHMAN, MOHAMMAD M, ROY, et al. ACK: Automatic API recommendation using crowdsourced knowledge [C] // 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER). 2016: 14-18.
[6] ZHANG J, JIANG H, REN Z, et al. Recommending apis for api related questions in stack overflow[J]. IEEE Access, 2018, 6: 6205-6219.
[7] HUANG Q, XIA X, XING Z C, et al. API Method Recommendation without Worrying about the Task-API Knowledge Gap [C]// Proceedings of the 2018 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE'18). 2018: 3-7.
[8] GU X, ZHANG H, ZHANG D, et al. Deep API learning[C] // FSE 2016: Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. 2016: 631-642.
[9] WALLACH, HANNA M. Topic modeling: beyond bag-of-words [C]// Proceedings of the 23rd International Conference on Machine Learning. ACM, 2006.
[10] RAMOS J. Using tf-idf to determine word relevance in document queries[C]// Proceedings of the First Instructional Conference on Machine Learning. 2003: 133-142.
[11] GU X D, ZHANG H Y, KIM S H. Deep Code Search[C]// 40th International Conference on Software Engineering (ICSE'18). 2018: 12.
[12] CAMPBELL B A, TREUDE C. NLP2Code: Code snippet content assist via natural language tasks[J]. arXiv: 1701. 05648, 2017.
[13] MIKOLOV T, KARAFIÁT M, BURGETL L, et al. Recurrent neural network based language model[C]// Eleventh Annual Conference of the International Speech Communication Association. 2010.
[14] DEVLIN J, CHANG M W, LEE K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv: 1810. 04805, 2018.
[15] MATTHEW E P, NEUMANN M, IYYER M, et al. Deep contextualized word representations[J]. arXiv: 1802. 05365, 2018.

LI Hao, born in 1970, Ph.D, professor. His main research interests include hybrid cloud computing, computer vision and robotics.

KANG Yan, born in 1972, Ph.D, associate professor. Her main research interests include software engineering, system optimization, big data processing and mining.

(上接第 514 页)

[26] WANG M, CAO D, LI L, et al. Microblog sentiment analysis based on cross-media bag-of-words model[C]// Proceedings of the International Conference on Internet Multimedia Computing and Service. Xiamen, China, ACM, 2014: 76.
[27] CAO D, JI R, LIN D, et al. A cross-media public sentiment analysis system for microblog[J]. Multimedia Systems, 2016, 22(4): 479-486.
[28] YOU Q, LUO J, JIN H, et al. Cross-modality consistent regression for joint visual-textual sentiment analysis of social multimedia[C]// Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. San Francisco, CA, USA, ACM, 2016: 13-22.
[29] YOU Q, CAO L, JIN H, et al. Robust visual-textual sentiment analysis: When attention meets tree-structured recursive neural networks[C]// Proceedings of the 24th ACM International Conference on Multimedia. Amsterdam, Netherlands, ACM, 2016: 1008-1017.
[30] ZADEH A, CHEN M, PORIA S, et al. Tensor fusion network for multimodal sentiment analysis [C]// Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017). Copenhagen, Denmark, 2017: 1103-1114.
[31] BAHDANAU D, CHO K, BENGIO Y. Neural machine translation by jointly learning to align and translate[J]. arXiv: 1409. 0473, 2014.
[32] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[C]// Advances in Neural Information Processing Systems. 2017: 5998-6008.
[33] YANG Z, YANG D, DYER C, et al. Hierarchical attention networks for document classification[C]// Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. San Diego, California, USA, 2016: 1480-1489.
[34] LU J, XIONG C, PARIKH D, et al. Knowing when to look: Adaptive attention via a visual sentinel for image captioning [C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017). Honolulu, HI, USA, 2017: 375-383.
[35] 杨琬琪, 高阳, 周新民, 等. 多模态张量数据挖掘算法及应用[J]. 计算机科学, 2012, 39(1): 9-13.

LIN Min-hong, born in 1995, postgraduate. Her main research interests include data mining and cross-media mining.

MENG Zu-qiang, born in 1974, Ph.D, professor, Ph.D supervisor, is a member of China Computer Federation. His main research interests include data mining, cross-media mining and KDD (Knowledge Discovery in Database).