基于多粒度语义匹配的编程任务解决方案推荐

郁思敏[1,2] 刘名威[1,2] 彭 鑫*[1,2] 王 翀[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 现有的编程任务检索工作或是仅基于问题标题进行检索；或是将问题全文与标题拼接后一起表征，忽视了不同类型文本之间的差异，导致推荐结果与任务之间存在差距。为解决上述问题，提出一种基于多粒度语义匹配的编程任务解决方案推荐方法 MGSMR。该方法分别基于标题和全文两个粒度使用不同的语义匹配模型寻找相关问答讨论作为候选解决方案，然后整合两个粒度的检索结果进行重排，最后补充 API 文档等外部知识生成解决方案推荐给开发人员。实验结果表明方法在两个数据集的多个评测指标上较对比方法提升了 18%-26%和 2%-4%，可缩短用户 23%的搜索时间，并提升所选答案 22%的正确性。

关键词 技术问答 多粒度语义匹配 句向量 稠密段落检索 解决方案生成

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

PROGRAMMING TASK-ORIENTED SOLUTION RECOMMENDATION METHOD BASED ON MULTI-GRANULARITY SEMANTIC MATCHING

Yu Simin[1,2] Liu Mingwei[1,2] Peng Xin*[1,2] Wang Chong[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 Existing programming solution recommendation methods either directly ignore the full text of the question or fail to find a suitable way to exert its value, resulting in inaccurate recommendation results. In this paper, we propose MGSMR, a programming task-oriented solution recommendation method based on multi-granularity semantic matching. For a given programming task, MGSMR uses the semantic matching model to find relevant question-and-answer discussions as candidate solutions based on two different granularities of the question title and the full text of the question. Then it further re-ranks the candidate solutions based on the multi-granularity matching method. Finally, it combines candidate solutions with external knowledge such as API documentation and third-party libraries to generate recommendation solutions. The evaluation shows that this method outperforms baselines in terms of multiple IR indicators of two datasets by 18%-26% and 2%-4%, and solution generation can help participants solve programming tasks 23% faster and 22% more accurately.

Keywords Stack Overflow Multi-Granularity Semantic Matching Sentence Embedding Dense Passage Retrieval Solution Generation

0 引 言

技术问答社区如 Stack Overflow 为开发人员提供了一个交流与知识共享的平台[21]。截止到 2022 年 9 月，仅 Stack Overflow 一家技术问答社区，就拥有两千多万的问题帖子，其中包含大量重复但问题和答案质量不一的帖子，用户需要浏览多个类似问题才能选出最合适的回答。此外，这些帖子中也不乏标题极其类似，但实际提问内容不同的帖子。若仅依靠标题进行检索，可能会推荐给用户并不相关的答案。编程任务和问题之间的差异也往往导致用户得不到想要的解决方案。

为了帮助开发人员快速解决编程任务，已有大量的研究基于信息检索和深度学习领域的相关技术，为给定的问题检索并推荐 Stack Overflow 上相关的帖子[4,22,23]。其中基于文本关键词匹配的方法如 TF-IDF[12]、BM25[13]，依赖编程任务和问题标题之间相同的文本表述，忽略了不同词汇可表达相同或相近的含义。使用词嵌入技术，如 Word2Vec[14]将文本向量化，再通过相似

收稿日期: 2023-03-27。国家自然科学基金(61972098)。**郁思敏**，硕士生，主研领域：智能化软件开发。**刘名威**，博士。**彭鑫**，教授。**王翀**，博士生。**赵文耘**，教授。

度计算的方式则可弥补这一缺陷。而自 BERT[16]诞生以来，越来越多的工作使用语义匹配的方式进行检索，并证实了其效果较前两种更具优势[24]。

然而，现有的绝大多数工作或是只采用了问题标题进行检索[5,10,24]，忽略了问题全文中隐含的重要信息；或是利用了问题全文，但没能充分挖掘其中的语义信息[4,7,27]。关于前者，一方面是因为仅用标题已经能取得不错的效果，且效率较高；另一方面则是由于查询通常是一句话的短文本，而问题全文是篇幅不限的长文本，其中包含大量和问题不相关的噪音，致使长短文本的匹配较为困难[25]。关于后者，已有研究证明了问题全文的有用性，但他们设计了复杂的方法从中提取关键信息进行索引匹配，可扩展性较差[7]。还有工作直接将标题和问题全文进行拼接，然后使用 CNN 网络获得全部文本的向量表示，这样简单的拼接方式无法发挥不同文本对检索的作用[27]。若能有效结合标题匹配和问题全文匹配，或许能得到更高质量的候选结果。

对于编程任务而言，除问题检索的效果以外，解决方案推荐的另一个难点是如何帮助开发人员快速判断候选回答是否与任务相关。Xu 等[5]对所有候选帖子的答案进行了汇总，使用最大边界相关算法 MMR 删除了回答中的相似表达。Silva 等[4]利用问题和回答中出现的文本为答案中的代码生成简要解释，并删除了没有代码的候选结果。但他们都忽略了 API 和三方库在开发中的重要地位，若能为答案文本和代码中出现的 API 和库提供补充信息，则可辅助开发人员高效编程。此外，为每个解决方案推荐相关帖子既可以作为答案的补充，又可以作为用户在不知如何准确描述任务时的参考，帮助他们进一步完善提问的方式。

针对上述问题，本文提出了基于多粒度语义匹配的编程任务解决方案推荐方法 MGSMR (**M**ulti-**G**ranularity **S**emantic **M**atching based **R**ecommendation)，从问题标题和问题全文两个粒度进行语义匹配，并对这两个粒度的检索结果进行重排过滤，最后补充信息为每个编程任务生成解决方案。实验表明本方法可以有效提升编程问题检索的准确率和召回率，生成的解决方案能缩短用户 23%的搜索时间，并提升所选答案 22%的正确性。

综上所述，本文的贡献主要有以下几点：

1）提出了一种基于多粒度的语义匹配模型，通过联合训练两个分别针对长短文本的独立模型深入挖掘问题全文的检索价值，并与基于标题的语义匹配相结合，从而得到比单粒度匹配更好的检索效果；

2）提出了一种为检索结果生成解决方案的方法，利用外部软件开发知识丰富解决方案的知识关联；

3）提出了一种自动化构建语义匹配模型训练数据集的方法，为 Java 语言的编程任务构建了包含 310,198 条数据的训练数据集。

1 相关工作

Stack Overflow 作为一个高质量的编程问答网站，吸引了众多开发人员在该平台上提问和回答问题。每隔一段时间研究者们就能从 Stack Exchange 上下载当前全部帖子的转储文件作为基础数据，从而开展各方面的研究。其中包括对网站问题的分类[3,7]、对重复帖子的检测[2,8]、对问题质量的预测[9]等一系列工作。还有一部分工作和软件工程领域紧密结合，如 API 推荐[10]、软件实体识别[11]、编程回答整合[5]等。其中，对 Stack Overflow 上的问题做检索是很多研究的重要步骤之一，其搜索准确率将很大程度影响后续工作的效果。Xu 等[5]检索问题相关帖子，然后挑选有用的回答段落，生成多样化的回答概述，节省了用户翻阅多个问题帖子的时间。Huang 等[10]搜索编程任务相关的问题帖子，然后从中抽取候选 API 辅助用户编程，通过 Stack Overflow 帖子弥补了直接搜索 API 官方文档存在的语义和知识鸿沟。Silva 等[4]聚焦于代码的搜索，首先尝试了 11 种问题检索的方法并取其中最优的方式得到相关帖子，接着多维度挑选高质量回答，并对答案进行重排，最后为结果中的代码生成简短的描述信息再返回给用户。这些工作大多是根据需求对结果进行整合，而忽略了软件开发相关知识的补充对用户浏览答案的帮助，本文则通过生成解决方案弥补这一缺口。

信息检索领域传统的方法如 TF-IDF[12]和 BM25[13]自诞生以来被广泛应用于问题检索，工业界在搜索领域最普遍使用的 Elastic Search，其默认相似度计算方法便是 BM25。该类方法侧重于问题表述的词形相似度，而 Word2Vec[14]和 Glove[15]等方法则使用词嵌入的方式弥补了词义相似度上的不足。Xu 等[5]融合了 Word2Vec 模型和 IDF 矩阵来计算提问和语料库中问题之间的相似性。诸如 BERT[16]的上下文词嵌入模型还能捕获不同句型语义上的区别，使问题检索的准确率有了更大的提升。Wei 等[17]提出了基于 BERT 句向量的 CLEAR 方法，用于方法级别和类级别的 API 推荐问题上。不同于之前的工作将标题短文本和描述长文本直接拼接后使用一个模型进行向量计算，或者直接忽略了问题全文所蕴含的补充信息，本文则进一步考虑了这两种类型文本之间的差异。针对两者不同的特点训练了两个独立且架构不同的模型，从而得到更好的特征表示，提升整体检索的效果。

2 方法实现

图 1 展示了解决方案推荐的方法流程，主要分为四个部分：数据集构建、模型训练、问题检索和解决方案生成。搜索库的内容来自 Stack Overflow，解决方案的补充知识主要来源于 API 官方文档和 Libraries.io。当用户查询一个编程任务时，本方法将对提问进行编码，从问题标题和问题全文两个粒度进行语义匹配，进而生成解决方案帮助用户快速找到有价值的回答。

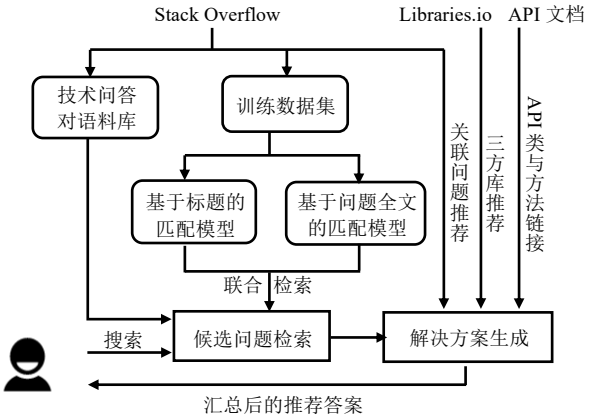

图 1 MGSMR 方法流程图

2.1 数据集构建

数据集的构建包含两个部分：技术问答对语料库的构建和模型训练数据集的构建。

2.1.1 技术问答对语料库

技术问答对语料库是编程问答系统的搜索主体，本文为最常见的编程语言 Java 构建了语料库，其他编程语言亦可通过相同的方式构建。为了提升语料库中答案的质量，本文规定挑选的数据需满足以下三个要求：1）问题的标签包含“java”或者“android”；2)问题的得分要大于 0；3）问题需要有被采纳的回答。本文最终从 Stack Overflow 2021 年 9 月的仓库中获取了 737,963 个问答讨论构成 Java 的技术问答对语料库。

2.1.2 训练数据集

模型的训练数据集由相似问题帖子对组成。为避免手动构建数据集所需的大量时间和人力成本，并保证数据质量，本文基于 Stack Overflow 提供的重复帖子数据集和帖子历史修改数据集挑选正样本数据。重复帖子数据集是由 Stack Overflow 标注的被认为是同一提问意图的帖子对，如“Convert String date to String Date different format”[3] 和“How to parse date string to Date?”[4]，这两个帖子都是对字符串类型的日期如何转换进行提问。帖子历史修改数据集则是帖子提问者对同一个帖子标题的历史修改记录，如“Sum up every N elements of a stream?”[5] 就是帖子发起者对原提问“How to sum up every N elements of java stream?”修改后的标题。这两类数据均可认为是对相同问题的不同表述，且历史修改数据较重复帖子数据在词形上更为相似。

然而这两个数据集均存在低质量的问题对，如“java - Variable type (generic or not) changes attributes behavior”[6] 和“What is a raw type and why shouldn't we use it?”[7] 被认为是重复问题，但从标题和问题全文的表述上无法认为两者是相似的，不适合作为训练正样本。因此，为保证训练数据集的质量，对两个数据集进行如下约束：1）问题必须出现在技术问答对语料库中；2）数据对之间标题和文本的雅卡尔相似度必须大于 0.1。本文实现时使用了 Python 库 TextDistance 来计算雅卡尔相似度。此外，考虑到完全一致的数据对训练没有意义，排除除标点符号和大小写以外均一致的数据对。

Stack Overflow 仓库中原本包含 Java 的 66,507 对重复帖子数据和 298,697 对帖子历史修改数据，根据规则过滤，除各自随机 1,000 对作为测试数据外，共获得 310,198 对训练正样本数据。数据集中的标题对用于训练标题匹配模型，一个帖子的标题和另一帖子的问题全文对则用于训练基于问题全文的匹配模型。

2.2 模型训练

本章涉及三个模型的训练：一个基于标题的匹配模型，以及两个分别针对编程任务短文本和问题全文长文本的问题全文匹配模型。

2.2.1 基于标题的匹配模型

预训练语言模型 BERT[16]可直接作为文本匹配模型使用，并已被证明有良好的检索效果[17,24]。但在特定领域的问题上，仍需对模型进行微调，使之达到最好的效果。

图 2 所示的是本文用于微调模型的对比学习架构，其中 SeBERT[1]是用于句子表征的基础模型，接着使用平均池化来连接句子的向量表示和孪生网络[18]。模型调优的输入数据是一个三元组<问题S，正相关标题P，负相关标题N>，本文先通过 SeBERT 模型得到向量表示，再经过池化层得到平均池化后的句子特征三元组$< E_S, E_P, E_N >$。三个句子使用的是同一个 SeBERT 模型，即它们彼此共享模型的参数权重，而更新参数所使用的损失函数则以最小化问题S和正相关标题P之间的距离，并最大化问题S和负相关标题N之间的距离为目标。其形式化的表述如公式(1)所示：

$$Loss(d_+, d_-, \in) = \max(d_+ - d_- + \in, 0) \quad (1)$$

[3] https://stackoverflow.com/questions/14999506
[4] https://stackoverflow.com/questions/4496359
[5] https://stackoverflow.com/questions/43089666
[6] https://stackoverflow.com/questions/56190396
[7] https://stackoverflow.com/questions/2770321

其中，d_+代表 E_S和E_P之间的欧式距离，$d_$代表 E_S和 E_N之间的欧式距离，∈则代表两个距离之间的边距，通常情况取 1。

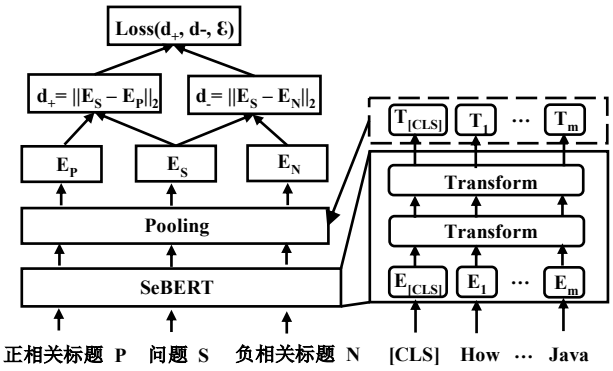

图 2 标题匹配模型的孪生网络架构

训练数据集的正样本在 2.1.2 节中已有提及，本文将相似问题对中一个帖子的标题作为问题S，另一个帖子的标题作为正相关标题P。负样本的选择使用了互相负样本的方式，即将同一批训练数据中其他数据的正样本标题P'作为本数据的负样本标题N，由此构建输入三元组。此外，本文选择 SeBERT 而不是其他 BERT 模型作为预训练语言模型，主要缘于 SeBERT 是基于大量软件开发相关语料训练的，其中包括 Stack Overflow 帖子和 Github 提交等文本信息，使其在编程问答领域较其他 BERT 模型有更好的效果。

2.2.2 基于问题全文的匹配模型

基于问题全文的语义匹配同样使用对比学习的方式来微调预训练语言模型，但其与基于标题的匹配模型有较大不同。基于问题全文的语义匹配包含两个模型，分别用于编程任务和问题全文的向量化，而标题匹配中编程任务和问题标题使用同一个模型向量化，这从架构上决定了两者是非常不同的两种语义匹配模型。问题全文匹配采用这种架构的主要原因在于编程任务一般为短句，而帖子的问题全文较长，两者的文本长度相差较大，表述方式也存在较大差异，使用两个模型能更好地提取对应的特征，从而使相似度计算更为准确。

图 3 展示了问题全文语义匹配模型的双塔网络架构，主要包含一个针对编程任务的查询编码模型和一个针对问题全文的编码模型。整个模型的输入是一个多元组，由一个查询Q，一个正相关全文P^+，和多个负相关全文P^-组成，正负相关全文共享一个模型，而查询独享一个模型。查询和全文之间的相似度计算遵从 Vladimir 在论文中[6]采用的内积计算方式，即

$$sim(q,p) = E_Q(q)^T E_P(p) \tag{2}$$

其中E_Q代表对查询的编码，E_p代表对问题全文的编码，q指查询的编程任务，而p指问题全文。训练的目标则是使查询和相关全文之间的距离在向量空间上接近，而查询和不相关全文之间的距离较远。因此，本文在训练过程中采用了如下极大似然损失函数：

$$Loss\left(E_Q, E_{P^+}, E_{P_1^-}, \dots, E_{P_n^-}\right) \tag{3}$$
$$= -log\frac{e^{sim\left(E_Q,E_{P^+}\right)}}{e^{sim\left(E_Q,E_{P^+}\right)} + \sum_{i=1}^{n} e^{sim\left(E_Q,E_{P_i^-}\right)}}$$

实现时，本文使用了 Facebook 提供的在多数据集上训练的预训练语言模型 *dpr-question_encoder-multiset-base* 和 *dpr-ctx_encoder-multiset-base* 分别作为查询和问题全文的基础模型。训练正样本同样使用了 2.1.2 节中构建的数据集，该模型将数据对中一个帖子的标题作为查询Q，另一个帖子的问题全文作为正相关全文P^+，并采用了两种不同的方式为每对数据构造 5 个负相关全文。其中一个负相关全文的构造方式和训练标题匹配模型时选择负样本的方式一致，即选择训练时同一批次其他数据的正相关全文作为本数据的负相关全文。为进一步提升模型区分正负样本的能力，其余 4 个负相关全文选择了难以区分的困难负样本，其构造方式包含两个步骤：1）基于标题匹配模型，检索出前 100 个和查询Q最相关的帖子；2）从这 100 个帖子中随机选择 4 个帖子的问题全文作为负相关全文，如果选中的帖子恰好是标准答案则再随机换一个帖子。由此可选出标题和查询相近，但实则无法解决问题的高质量负样本。

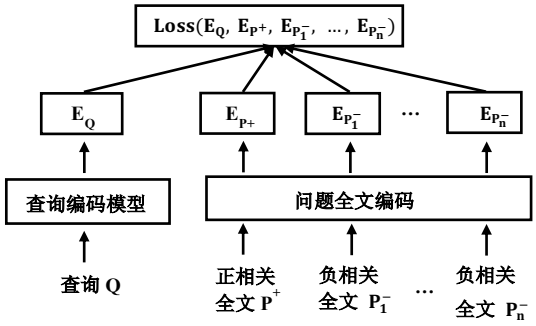

图 3 问题全文匹配模型的双塔网络架构

2.3 问题检索

问题检索的质量高低是评判一个问答系统优劣的决定性因素，不仅要求返回的结果中与问题相关的回答尽量靠前，还要求尽可能缩短检索的响应时间。

为改善返回结果的质量，本文提出了一种基于多粒度语义匹配的检索模型，融合了标题匹配和问题全文匹配两个粒度的优点。已有的工作[4,5]证明了仅用问题标题就能为很多问题找到非常相关的回答，为后续工作奠定良好的基础。但部分问题仅依靠标题相似度可能会错失最相关的帖子，如问题“How to fix ConcurrentModificationException when using foreach to iterate over a List? ”[8] 中非常关键的“Concurrent-ModificationException”在其最相关的帖子“Java -

adding elements to list while iterating over it”[9] 标题中是缺失的，但该帖子的问题全文中包含了这个至关重要的表述“I want to avoid getting ConcurrentModification-Exception.”。由此可见，问题全文对帖子意图的理解和新问题的搜索都有相当重要的价值。但帖子提问的详细信息中又会有部分冗余，甚至与主题毫不相关的文本，对段落的向量编码和特征提取造成干扰，从而影响检索的效果。而本文提出的多粒度语义匹配模型能有效结合这两个粒度检索的优点。

当给定编程任务后，基于多粒度的语义匹配方法将执行以下三个步骤：1）使用基于标题的匹配模型对任务进行编码，从技术问答对语料库中选出问题标题与任务最相关的前 k 个帖子；2）使用问题全文匹配模型中的问题编码模型对任务向量化，从技术问答对语料库中选出问题全文与任务最相关的前 k 个帖子；3）使用倒数排名[19]融合两种方式选出来的帖子，去重并返回与任务最相关的前 k 个帖子。

为提升检索的响应时间，本文提前为技术问答对语料库构建了两个 Milvus 向量数据库，分别计算并存储问题标题向量和问题全文向量，以便在给定编程任务后，可以快速从 Milvus 数据库中检索到最相近的前 k 个向量，从而得到对应的相关帖子。

2.4 解决方案生成

问题检索得到前 k 个最相关的帖子后，为保证答案的正确性，本文选择帖子的被采纳回答作为候选解决方案，并补充软件开发相关的知识辅助用户进一步解决问题。解决方案的补充知识主要包括以下三个方面：1）关联问题推荐，可帮助用户在不知如何准确描述问题时，产生更多的联想，并进一步完善提问的方式，从而快速找到想要的回答；2）API 类与方法链接，针对文本和代码中出现的 API 给出辅助信息，包括其全限定名，描述文本如 API 功能、返回值、异常等信息，以及 API 官方文档的超链接，帮助用户快速甄别推荐的 API 是否能够满足其需求，并简单掌握 API 的使用规则和注意事项；3）三方库推荐，针对文本中描述的功能需求，推荐类似功能的库，展示内容包括库的名称、功能性描述以及库的官方链接，以供用户参考。

为了在生成的解决方案中补充以上知识，本文从以下三个来源获取相关信息：1）从 Stack Overflow 上获得每个帖子的关联问题，技术问答对语料库中共 318,422 个帖子存在关联问题；2）从 Java 和 Android 的官方文档中获取所有 API 类和方法的名称、描述等有用信息，共收集到 Java 相关的 7,195 个类和 67,828 个方法；3）从 Libraries.io 网站上获取和 Java 相关的库及对应的描述信息用于相关三方库的推荐，共收集到 70,942 条数据。

基于上述收集到的信息，本文为生成的解决方案补充关联的知识。其中关联问题推荐直接使用相关问题帖子的 ID 查询数据库中对应的关联帖子，在实验中本文为每个回答推荐至多 3 个关联问题，并按照问题与帖子的文本雅卡尔相似度和帖子得分从高到低进行排序。而 API 类与方法的链接则需对文本和代码进行识别，分词后将首字母大写、驼峰式或以点相连的单词作为候选 API，然后与官方文档抽取出来的 API 全限定名和对应的别名进行链接。三方库的推荐则是根据回答文本的功能描述进行相关性搜索，查找的范围是收集到的全部三方库的功能性描述文本，并为其构建向量数据库进行快速搜索。相似度计算的模型则采用基于标题的语义匹配模型的架构。

综上，当用户搜索一个编程任务时，本方法为基于多粒度语义匹配模型检索出的每一个帖子的被采纳回答增加以上三部分补充信息，进而整合前 k 个相关回答生成解决方案推荐给用户。

3 实 验

为验证本文提出的基于多粒度语义匹配方法的有效性和生成的解决方案的有用性，本章共设置了两个对比实验对这两点进行验证分析。

3.1 多粒度语义匹配的有效性评估

目前主流的搜索服务大多是基于 Elastic Search（ES）实现的，具有高性能、高可用的特点，因此本实验也选择 ES 默认的检索算法 BM25[13]作为对比方法之一。为进一步探索方法的有效性，本文还设置了消融实验分别验证每个模块的效果。其中基于标题的语义匹配也代表了现有工作的最佳效果。此外，实验还探究了回答全文对问题检索的作用。

本对比实验共包含 7 种方法：1）BM25-标题：使用 BM25 检索问题标题；2）BM25-问题全文：使用 BM25 检索问题标题和问题全文；3）BM25-回答全文：使用 BM25 检索问题标题和回答全文；4）MGSMR-标题：使用基于标题的语义匹配模型检索问题标题向量库；5）MGSMR-问题全文：使用基于问题全文的语义匹配模型检索问题全文向量库；6）MGSMR-回答全文：使用基于回答全文的语义匹配模型检索回答全文向量库；7）MGSMR：基于多粒度的语义匹配，即本文提出的方法。注意，5）和 6）两种全文语义匹配检索所用的模型是不同的，分别在各自的搜索语料上进行训

[8] https://stackoverflow.com/questions/ 43606011

[9] https://stackoverflow.com/questions/ 11624220

练，互不干扰。

实验采用了重复帖子数据集和帖子历史修改数据集，两者各包含 1,000 条测试数据，在 2.1.2 章节已有说明。本文认为当帖子 A 是帖子 B 的重复帖子，且帖子 B 是帖子 C 的重复帖子时，帖子 A 也是帖子 C 的重复帖子。因此重复帖子数据集中的一条数据可能包含多个正确答案，而帖子历史修改数据集中每条数据仅包含一个正确答案。

评测指标使用了在信息检索领域广泛使用的四个指标[20]，包括 Accuracy@k, Recall@k, MRR 和 MAP。Accuracy@k 代表前 k 个结果中是否包含一个正相关的回答；Recall@k 代表前 k 个回答的召回率；MRR 平均倒数排名反映了第一个正相关回答在结果中所处的位置；MAP 平均准确率则反映了所有正相关回答在返回结果中所处的位置。这四个指标的值均是越接近 1，表明方法的检索效果越好。

从表 1 的结果中可以观察到，在两个数据集的不同评测指标上，均是本文方法的检索效果最好。相较于广泛使用的 BM25 算法，MGSMR 在两个数据集上比 BM25 最好的版本在各个指标上分别提升了 18% - 26% 和 2% - 4%。帖子历史修改数据集 BM25 效果已经很好，故而提升百分比不够明显。

对比 BM25 的三个不同版本， BM25-问题全文在重复帖子数据集上表现最佳，而在帖子历史修改数据集上仅用标题的 BM25 效果最好。由此可以推断，帖子历史修改数据集问题对之间的词形相似度较高，故仅用标题搜索就能取得较好的效果，增加的文本信息反而对搜索造成了干扰；而重复帖子数据集的问题文本中蕴含较多对搜索有益的补充信息，这也是多粒度语义匹配在重复帖子数据集上表现更加突出的原因。而回答全文在 BM25 和 MGSMR 上的表现均不如标题或者问题全文，主要是因为回答的文本侧重于描述问题的解决方案，其与搜索之间的文本相似度普遍低于问题全文或标题与搜索之间的相似度。因此，本文的多粒度语义匹配方法仅基于标题和问题全文两个粒度。

表 1 不同方法在两个测试集上的表现

数据集	评测指标	BM25-标题	BM25-问题全文	BM25-回答全文	MGSMR-标题	MGSMR-问题全文	MGSMR-回答全文	MGSMR
重复帖子数据集	Accuracy@1	0.085	0.134	0.058	0.155	0.147	0.132	**0.159**
	Accuracy@3	0.146	0.223	0.106	0.253	0.250	0.222	**0.278**
	Accuracy@5	0.191	0.264	0.131	0.305	0.310	0.265	**0.326**
	Accuracy@10	0.240	0.323	0.181	0.371	0.377	0.346	**0.401**
	Recall@1	0.082	0.127	0.054	0.147	0.139	0.123	**0.152**
	Recall@3	0.136	0.207	0.098	0.240	0.234	0.209	**0.261**
	Recall@5	0.178	0.246	0.119	0.290	0.294	0.249	**0.310**
	Recall@10	0.226	0.302	0.168	0.352	0.355	0.325	**0.381**
	MRR@10	0.128	0.191	0.092	0.219	0.213	0.192	**0.232**
	MAP@10	0.122	0.180	0.085	0.209	0.202	0.181	**0.222**
帖子历史修改数据集	Accuracy@1	0.865	0.752	0.622	**0.880**	0.863	0.835	**0.880**
	Accuracy@3	0.898	0.841	0.726	0.920	0.920	0.898	**0.934**
	Accuracy@5	0.913	0.871	0.767	0.928	0.936	0.906	**0.945**
	Accuracy@10	0.922	0.900	0.822	0.939	0.953	0.926	**0.957**
	Recall@1	0.865	0.752	0.622	**0.880**	0.863	0.835	**0.880**
	Recall@3	0.898	0.841	0.726	0.920	0.920	0.898	**0.934**
	Recall@5	0.913	0.871	0.767	0.928	0.936	0.906	**0.945**
	Recall@10	0.922	0.900	0.822	0.939	0.953	0.926	**0.957**
	MRR@10	0.885	0.805	0.685	0.901	0.895	0.868	**0.910**
	MAP@10	0.885	0.805	0.685	0.901	0.895	0.868	**0.910**

此外，在这两个数据集前 1、前 3 指标上的表现均是 MGSMR-标题优于 MGSMR-问题全文，但在前 5、前 10 指标上的表现则是后者好于前者。观察数据可发现前 3 就能找到的正样本，一般标题相似度较高，因此只用标题检索就能取得较好的效果；而靠后一些找到的正样本，其问题全文提供了很好的补充信息，故而基于问题全文语义匹配的检索能发挥更大作用。本文提出的多粒度语义匹配方法则融合了两者的优点，在全部的指标上均表现最佳，充分证明了方法的有效性。

3.2 解决方案生成的有用性评估

为探究生成的解决方案能否帮助用户更加高效准确地解决编程问题，实验邀请了 6 位软件工程专业的硕士研究生，比较了他们在特定编程任务上分别给出解决方案和只有 Stack Overflow 原始帖子两种情况下的表现。本文按照参与者们 Java 开发的经验将其平分为两组，简称 *PA* 和 *PB*。接着从两个测试集中随机选择 10 个编程任务，按照多粒度语义匹配检索到的相关问题出现的位置将其平分为两组，每组均包含排名靠前和排名靠后的任务，且数量相同，简称 *TA* 和 *TB*。

针对每个编程任务，实验要求参与者们从多粒度语义匹配检索给出的前 10 个回答中挑选并记录下他们认为可以解决问题的 0 个或者多个回答。具体地，*PA* 完成 *TA* 任务时，返回的是本方法生成的解决方案，完成 *TB* 任务时则是以 Stack Overflow 的方式展示的检索结果；而 *PB* 则与之相反。实验过程中，还要求参与者们记录每个任务完成的时间，超过 10 分钟仍未找到答案或认为没有合适的回答，则按照 10 分钟记录。实验结束后，参与者们需要对解决方案的可读性和有用性进行打分，采用四分制（1-不认可；2-比较不认可；3-比较认可；4-认可）。

表 2 使用 MGSMR 和对比方法完成编程任务的情况

	完成任务的平均时长	答案的平均准确率
MGSMR	130.57s	0.933
对比方法	169.63s	0.767

表 2 展示了参与者们在使用 MGSMR 和对比方法完成编程任务时，挑选检索结果的平均时长和平均准确率。可以看到 MGSMR 可以缩短用户 23%的搜索时间，并提升选中正确答案 22%的概率。经分析，针对正确答案排名靠前的问题，MGSMR 给出的辅助信息可以帮助用户快速确定该答案能否解决问题，从而有效缩短用户完成任务的时间。而针对个别答案靠后的问题，使用本方法的参与者们虽然较对比方法花费了更多的时间，但能更高概率地找到正确答案。据反馈，有时参与者们也会消耗更多的时间查看关联问题，尤其是当关联问题的标题更符合他们理想的问题表述时，从而快速定位相关问题并减少翻阅帖子的数量。

此外，参与者所打可读性的均分为 3.67，有用性的均分为 3.5，且两者均无低于 3 分的评分，对生成的解决方案给出了肯定的评价。一位评分 3 分的参与者反馈，本文方法为每个解决方案补充的信息维度都是相同的，如果针对不同类型的任务，补充不同维度的信息能更好地帮助用户。此外，也有参与者认为三方库的推荐不够准确，可以进一步扩大语料库并提升推荐模型的准确率。这两点也将成为本文未来的改进方向。

总体而言，用户对生成的解决方案较为认可，能有效辅助他们查看编程问题的搜索结果，增强他们对所选答案的信心，并缩短浏览的时间。

4 结 语

本文提出了基于多粒度语义匹配的编程任务解决方案推荐方法 MGSMR，训练了一个基于问题标题的匹配模型，和两个分别针对编程任务短文本和问题全文长文本的匹配模型。此外，本文还提出并实现了为检索结果生成解决方案的方法，通过知识的补充进一步辅助用户高效准确地完成编程任务。实验结果充分证明了多粒度语义匹配模型在答案检索上的有效性，以及生成的解决方案对用户浏览答案的有用性。未来可以进一步优化解决方案补充的知识，针对不同类型的编程任务提供更有侧重点的辅助信息。

参 考 文 献

[1] Von der Mosel J, Trautsch A, Herbold S. On the validity of pre-trained transformers for natural language processing in the software engineering domain[J]. IEEE Transactions on Software Engineering, 2022.

[2] Ahasanuzzaman M, Asaduzzaman M, Roy C K, et al. Mining duplicate questions of stack overflow[C]//2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR). IEEE, 2016: 402-412.

[3] Zhu W, Zhang H, Hassan A E, et al. An empirical study of question discussions on Stack Overflow[J]. Empirical Software Engineering, 2022, 27(6): 148.

[4] Da Silva R F G, Roy C K, Rahman M M, et al. CROKAGE: effective solution recommendation for programming tasks by leveraging crowd knowledge[J]. Empirical Software Engineering, 2020, 25(6): 4707-4758.

[5] Xu B, Xing Z, Xia X, et al. AnswerBot: Automated generation of answer summary to developers' technical questions [C] // 2017 32nd IEEE/ACM international conference on automated software engineering (ASE). IEEE, 2017: 706-716.

[6] Karpukhin V, Oğuz B, Min S, et al. Dense passage retrieval for open-domain question answering[J]. arXiv preprint arXiv: 2004.04906, 2020.

[7] Liu M, Peng X, Marcus A, et al. API-Related Developer Information Needs in Stack Overflow[J]. IEEE Transactions on Software Engineering, 2021.

[8] Zhang Y, Lo D, Xia X, et al. Multi-factor duplicate question detection in stack overflow[J]. Journal of Computer Science and Technology, 2015, 30(5): 981-997.

[9] Ponzanelli L, Mocci A, Bacchelli A, et al. Improving low quality stack overflow post detection[C]//2014 IEEE international conference on software maintenance and evolution. IEEE, 2014: 541-544.

[10] Huang Q, Xia X, Xing Z, et al. API method recommendation without worrying about the task-API knowledge gap[C]//2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2018: 293-304.

[11] Ye D, Xing Z, Foo C Y, et al. Software-specific named entity recognition in software engineering social content[C]//2016 IEEE 23rd international conference on software analysis, evolution, and reengineering (SANER). IEEE, 2016, 1: 90-101.

[12] Salton G, Buckley C. Term-weighting approaches in automatic text retrieval[J]. Information processing & management, 1988, 24(5): 513-523.

[13] Robertson S E, Walker S. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval[C]//SIGIR'94. Springer, London, 1994: 232-241.

[14] Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality[J]. Advances in neural information processing systems, 2013, 26.

[15] Pennington J, Socher R, Manning C D. Glove: Global vectors for word representation[C]//Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014: 1532-1543.

[16] 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.

[17] Wei M, Harzevili N S, Huang Y, et al. CLEAR: contrastive learning for API recommendation[C]//Proceedings of the 44th International Conference on Software Engineering. 2022: 376-387.

[18] Chopra S, Hadsell R, LeCun Y. Learning a similarity metric discriminatively, with application to face verification[C]//2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). IEEE, 2005, 1: 539-546.

[19] Cormack G V, Clarke C L A, Buettcher S. Reciprocal rank fusion outperforms condorcet and individual rank learning methods[C]//Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval. 2009: 758-759.

[20] Manning C D. Introduction to information retrieval[M]. Syngress Publishing, 2008.

[21] Treude C, Barzilay O, Storey M A. How do programmers ask and answer questions on the web?(nier track)[C]//Proceedings of the 33rd international conference on software engineering. 2011: 804-807.

[22] Huang Q, Xia X, Xing Z, et al. API method recommendation without worrying about the task-API knowledge gap [C] // Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. 2018: 293-304.

[23] Lin Z, Zou Y, Zhao J, et al. Improving software text retrieval using conceptual knowledge in source code[C]//2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2017: 123-134.

[24] 徐成龙. 融合Stack Overflow和API文档信息的API推荐方法研究[D]. 江苏：东南大学，2021.

[25] Pang L, Lan Y, Cheng X. Match-ignition: Plugging pagerank into transformer for long-form text matching[C]//Proceedings of the 30th ACM International Conference on Information & Knowledge Management. 2021: 1396-1405.

[26] Liu M, Peng X, Marcus A, et al. Generating query-specific class API summaries[C]//Proceedings of the 2019 27th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering. 2019: 120-130.

[27] Gao W, Wu J, Xu G. Detecting duplicate questions in stack overflow via source code modeling[J]. International Journal of Software Engineering and Knowledge Engineering, 2022, 32(02): 227-255.