一种基于学习模型的 Web API 功能互补推荐方法

陈琪琪[1),2)] 康国胜[1),2)] 聂雅梅[1),2)] 刘建勋[1)] 曹步清[1)]

[1)](湖南科技大学计算机科学与工程学院，湘潭)
[2)](湖南科技大学服务计算与新软件技术湖南省重点实验室，湘潭)

摘要 随着服务计算技术的发展，互联网上的 Web API 呈指数级增长。然而，从这个庞大的 API 池中选择合适的 API 来创建 Mashup 对用户来说是一个挑战。已经提出了各种 Web API 推荐方法来解决这个问题，旨在简化复杂的选择过程。尽管做出了这些努力，但关于互补功能推荐的研究还很有限。在这种情况下，基于学习模型(名为 CoWAR)的通用互补 API 推荐框架，被设计用于根据用户已选 Web API 推荐为 Mashup 创建量身定制的互补 Web API。具体来说，我们提出了一种数据标记算法，该算法基于源自历史 Mashup 和 Web API 的 Mashup-API 交互生成标记数据集。此外，我们采用 BERT 模型基于功能描述文档生成 Web API 的表示向量。随后，我们利用 SANFM (Self - Attention Neural Factorization Machines，自注意神经分解机)基于 Web API 的表示向量，用标记的样本数据集训练互补 Web API 推荐模型。据我们所知，这是第一个用学习模型解决互补功能推荐问题的工作。通过在真实数据集上进行一组实验，验证了所提出方法的有效性。实验结果表明，该学习模型优于传统的基于机器学习的模型和几种基于深度学习的模型。

关键词 Web API；Mashup；互补功能；服务推荐；学习模型

A General Complementary Web API Recommendation Framework based on Learning Model

Qiqi Chen[1),2)] Guosheng Kang[1),2)] Yamei Nie[1),2)] Jianxun Liu[1)] Buqing Cao[1)]

[1)] (*School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan*)
[2)] (*Hunan Provincial Key Lab. for Services Computing and Novel Software Technology, HNUST, Xiangtan*)

Abstract With the advancement of service computing technology, the Internet has witnessed an exponential proliferation of Web APIs. However, the selection of suitable APIs from this vast pool for Mashup creation poses a challenge for users. Various Web API recommendation methods have been proposed to address this issue, aiming to simplify the complex selection process. Despite these efforts, limited study has been conducted on complementary function recommendation. In this context, a general complementary API recommendation framework based on a learning model, named CoWAR, is designed to recommend complementary Web APIs tailored for Mashup creation, based on the user's selected Web APIs. Specifically, we propose a data labeling algorithm to generate the labeled dataset based on Mashup-API interactions derived from historical Mashups and Web APIs. Additionally, we employ BERT model to generate representation vectors of Web APIs based on the functionality description documents. Subsequently, we utilize SANFM (Self-Attentional Neural Factorization Machines) to train the complementary Web API recommendation model with the labeled sample dataset based on representation vectors of Web APIs. To the best of our knowledge, this is the first work addressing the complementary function recommendation problem with a learning model. By conducting a set of experiments over a real-world dataset, the effectiveness of the proposed approach is validated. The experimental results demonstrate that the learning model outperforms the traditional machine learning-based models and several deep learning-based models.

Keywords Web API; Mashup; Complementary Function; Service Recommendation; Learning Model

1. 引言

Mashup 代表一种新型的 Web 应用程序，它利用了来自多个数据源的内容，形成了创新的、全新的服务。Web API（即 RESTful Web 服务）提供了一系列功能，可以通过编排来创建增值和原始服务——我们通常将其称为 Mashup [1]. Mashup 和 Web API 的广泛在线可用性证明了与 Mashup 技术相关的多种好处和经济优势。在如此丰富的 Web API 中，用户面临着识别用于 Mashup 创建的理想 Web API 的挑战。为了解决这个问题，已经提出了各种 Web API 推荐方法来简化给定服务需求的选择过程。然而，关于互补功能推荐的研究还很有限。考虑到用户在整个 Mashup 开发过程中经常采用多个 Web API，基于所选 Web API 的补充 Web API 的建议变得至关重要。这不仅帮助用户驾驭 Mashup 开发的复杂性，而且还提高了整个过程的效率。与一般由活跃用户输入特定服务需求的 Web API 推荐不同，在互补功能推荐场景中，用户输入需求可能是未知的。唯一的基础是活动用户选择的 Web API。

近年来，人们对 Web API 推荐的研究越来越关注，提出了各种方法来缓解 Web API 选择的挑战。通过对现有 Web API 推荐工作的回顾，这些方法可以大致分为三组：基于内容的方法、基于协同过滤的方法和基于图的方法。基于内容的方法主要关注用户指定的当前需求 [1]. 在某些研究中，附加的辅助信息被认为可以增强描述文档的语义。尽管包含了更多的信息，但通过主题建模进行简单匹配被认为是不够的。基于 CF 的方法尝试通过考虑类似用户已经选择的 Web API 来推荐 Web API [2, 3]. 基于图的方法尝试从服务网络中提取语义以改善服务匹配 [4].

上面提到的工作集中在基于用户需求输入或足够的历史数据查找相似用户的通用 Web API 推荐场景。然而，关于互补功能推荐的研究有限。Tang 等人 [5] 提出了一种挖掘 API 之间协作模式的方法，以帮助创建 Mashup。此外，He 等人 [6] 考虑到用户在一个开发周期内会调用多个功能，提出了一种互补的功能推荐方法。但是，推荐结果受以前使用的流行 Web API 的影响，忽略了长尾 Web API 和从未调用过的 Web API。在实践中，只有少数 Web API 被使用，大多数都未被使用。以 ProgrammableWeb 的 Web API 为例，总共有 12919 个 Web API，但只有 1718 个 Web API 被调用（即 13.29%）。在使用的 Web API 中，生成了 13523 个调用。14.54%的 Web API（即 250）占用了 80%的调用，85.46%的 Web API（即 1468=1718-250）占用了其余 20%的调用，如 Fig. 1 所示。因此，为了服务生态系统的公平和可持续发展，最好在互补 API 推荐场景中平等地考虑所有 Web API（包括长尾 API 和未使用的 API）。这是我们工作的动力。

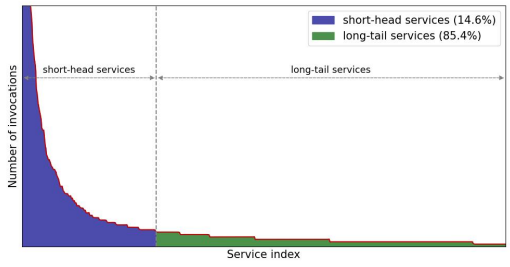

Figure 1. 长尾短头服务的例证（服务按照调用次数降序排序）

由于 Mashup-API 生态系统中存在明显的长尾效应和使用稀疏性，因此在服务推荐方法中，考虑未使用的 Web API 变得至关重要。为了克服现有工作在互补功能推荐场景中的局限性，本文基于基于深度学习的学习模型，设计了一种新颖通用的互补 Web API 推荐框架，命名为 CoWAR（Complementary Web API Recommendation）。该框架在推荐过程中充分考虑了长尾和未使用的 Web API。具体来说，数据集是基于 Mashup 及其调用的 Web API 进行标记的，作为互补 Web API 推荐模型的训练数据。同时，基于 BERT 模型获取 Web API 的表示向量。随后，我们提出使用 SANFM（Self-Attention Neural Factorization Machines，自注意神经分解机）以标记的数据集和 Web API 的表示向量作为输入，训练互补的 Web API 推荐模型。据我们所知，这是第一个用学习模型解决互补功能推荐问题的工作。我们的主要工作贡献总结如下。

- ✧ 为了促进服务使用的公平性，互补功能推荐考虑了长尾 API 和未使用 API。因此，所有 Web API 候选者都被视为第一公民。
- ✧ 基于 Mashup 与 API 之间的交互，提出了一种标记算法来生成标记数据集。此外，还学习了基于深度学习的 SANFM 模型，用于互补 API 推荐，该模型结合了复杂的关系，包括低阶和高阶，同时考虑了特征交互的不同重要性。
- ✧ 在 ProgrammableWeb 的真实数据集上进行了大量的实验，实验结果表明，在提出的通用 CoWAR 框架下，SANFM 模型优于基于选择性机器学习的基线方法和一些基于竞争性深度学习的基线方法。

本文的其余部分结构如下。第 II 部分概述了拟议的互补 Web API 推荐框架。第 III 部分详细介绍了拟议的方法。第 IV 部分通过一系列实验对所提出的互补 Web API 推荐框架下的学习模型进行了评估。最后，第 V 部分对本文进行了总结，并对未来的工作进行了展望。

2. 框架

互补 Web API 推荐涉及在 Mashup 开发过程中，根据所选 Web API 主动向用户提供互补 Web API 列表。因此，从功能的角度来看，所提供列表中的 Web API 应该补充所选的 Web API。在互补 Web API 推荐的场景中，默认的假设是活动用户已经选择了一个或多个 Web API。现在，我们描述设计的互补 Web API 推荐的综合框架，包括数据标记、服务表示和学习模型，如 Fig. 2 所示。

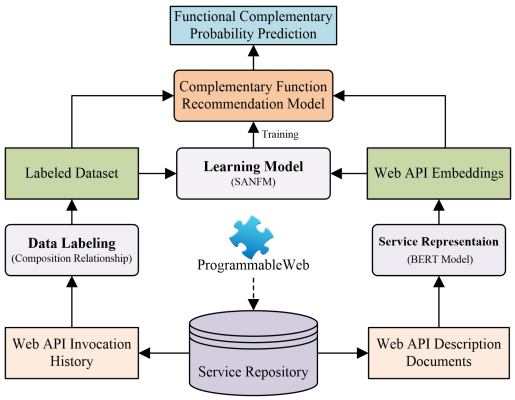

Figure 2. 创建的互补 Web API 推荐提出的的体系结构

在该框架中，从 ProgrammableWeb 抓取的关于 Mashup 和 API 的数据包括它们的功能描述文本以及 Mashup 和 API 之间的组合关系。抓取的数据存储在 Service Repository 中。一些常用的预处理技术，如规范化、词干提取、标记和停止词删除，在表征之前用于功能描述文档。然后通过在 Service Representation 组件中应用 BERT 模型生成 Web API 嵌入。此外，通过 Data Labeling 组件，可以根据 Mashup 与 Web API 之间的组合关系获得互补 Web API 样本数据集。直观地看，同一个 Mashup 所使用的 Web API 从功能的角度来看是互补关系[7]。通过标记数据集和 Web API 嵌入，在 Learning Model 组件中采用基于深度学习的 SANFM 模型，推导出互补的 Web API 推荐模型。最后，基于预测的互补概率，以所选 API 为前提，为活动用户生成 top k 互补 Web API。请注意，可以使用更高级或更有效的表示模型或推荐模型来代替 BERT 模型和 SANFM 模型，而我们的贡献更多地集中在通用框架上。

3. 学习用于 Mashup 创建的互补 Web API 推荐模型

在本节中，将介绍在第 II 节中提到的拟议的互补 Web API 推荐框架中包含的主要组件。首先，介绍了基于 API 之间的组合关系生成互补 Web API 数据的方法。然后，利用 BERT 模型获取 Web API 的表示向量。最后，详细描述了互补 Web API 推荐模型的学习过程。

A. 基于组合关系的数据标记

通常，将多个 API 组合到 Mashup 中以创建增值应用程序。以 Mashup Vizlingo[1] 为例，它由三个 Web API 组成：YouTube, Facebook, 和 Twitter。 正式地，Mashup 被定义为 $M=\{M_1,M_2,\cdots,M_N\}$，以及 API 被定义为 $A=\{A_1,A_2,\cdots,A_n\}$。每个 Mashup 都是一组 Web API。例如， $M_k=\{A_o,A_p,A_q\}$ 表示 Mashup M_k 由三个 API 组成： A_o, A_p, and A_q。因此 $M_k \subseteq A$，并且 A_o, A_p 和 A_q 存在一种构成或组成关系。通常，基于元素和子集的概念，我们定义一个 Web API 和一个 Web API 集合之间的互补关系如 Def. 1。

Definition 1（互补关系）。给定 Mashup 集合 $M=\{M_1,M_2,\cdots,M_N\}$ 和 API 集合 $A=\{A_1,A_2,\cdots,A_n\}$，对于任意 $A_i \in A$，以及任意非空真子集 $M_s \subset M_k$，若 $A_i \in (M_k - M_s)$，则称 A_i 和 M_s 具有互补关系。

基于 Def. 1，可以得到正样本。对于 $M_k=\{A_o,A_p,A_q\}$，如果一个活跃用户选择了 A_o 和 A_p，那么互补的 Web API A_q 应该被推荐到推荐列表中，即$\{A_o,A_p\}$ 与 A_q 互补，这种情况成为一个正样本，即一个三元组 $(\{A_o,A_p\},A_q,1)$。同样，其他 8 个正样本可以得出，如 Table I 所示。

TABLE I. 数据标记的正样本示例

Mashup	Selected APIs	Complementary API	Label
$M_k=\{A_o,A_p,A_q\}$	$\{A_o,A_p\}$	A_q	1
	$\{A_o,A_q\}$	A_p	1
	$\{A_p,A_q\}$	A_o	1
	$\{A_o\}$	A_p	1
	$\{A_o\}$	A_q	1
	$\{A_p\}$	A_o	1
	$\{A_p\}$	A_q	1
	$\{A_q\}$	A_o	1
	$\{A_q\}$	A_p	1

同时，负样本的选取应以正样本为基础。它不仅考虑了局部非互补性，而且考虑了全局非互补性。如果不考虑两者中的一个，则再负样本的选取上可能不准确，这将干扰推荐模型的训练和预测。局部非互补性： 如果 $A_x \notin M_k$，那么 A_x 与 $\{A_o,A_p\}$、$\{A_o,A_q\}$ 或 $\{A_p,A_q\}$ 不具有互补关系。因此，非互补 Web API $A_x \notin M_k$ 可以剩余的 Web API 中被随机选取，即 $M-M_k$。全局非互补性：假设 Mashup 数据集仅由四个 Mashup 构成，分别是 $M_k=\{A_o,A_p,A_q\}$, $M_A=\{A_o,A_p,A_M\}$, $M_B=\{A_p,A_q,A_N\}$, 和 $M_C=\{A_o,A_q,A_H\}$。 如果在M_k的数据标记过程中仅仅考虑局部非互补性， 当已选 API 是 $\{A_o,A_p\}$，并且随机选取的非互补 Web API 是 A_M，这个 API 仅满足局部非互补性。由于 M_A 是$\{A_o,A_p,A_M\}$中的组成元素，在数据标记过程中对于样本 $\{A_o,A_p,A_M\}$ 会被同时识别为正样本和负样本。因此，这样的样本数据应该被剔除。总之，负样本的选择需要满足局部和全局的非互补性。基于以上观察，定义非互补关系为 Def. 2。

Definition 2（非互补关系）。给定 Mashup 集合 $M=\{M_1,M_2,\cdots,M_N\}$ 和 API 集合 $A=\{A_1,A_2,\cdots,A_n\}$，对于 $M_k \in M$，任意 $M_p \in M$， $M_k \neq M_p$，任何非空真子集 $M_s \subset M_k$，以及 $A_i \in A$，如果 $A_i \in (A-M_k)$ 和 $(\{A_i\} \cup M_s) \notin M_p$，那么则称 A_i 和 M_s 具有非互补关系。

TABLE II. 数据标记的负样本示例

Mashup	Selected APIs	Complementary API	Label
$M_k=\{A_o,A_p,A_q\}$	$\{A_o,A_p\}$	A_a	0
	$\{A_o,A_q\}$	A_b	0
	$\{A_p,A_q\}$	A_c	0
	$\{A_o\}$	A_d	0
	$\{A_o\}$	A_e	0
	$\{A_p\}$	A_f	0
	$\{A_p\}$	A_g	0
	$\{A_q\}$	A_h	0
	$\{A_q\}$	A_i	0

基于 Def. 2，可以导出负样本。如 Table II 所示，对于 $M_k=\{A_o,A_p,A_q\}$，对应M_k 可以得到 9 个负样本。 根据上面的观察，如果 Mashup 包含 n 个 Web API， 则可以生成 $\sum_{k=2}^{n} kC_n^k$ 个相关的正样本。在模型训练过程中，每个正样本后面都有一个相关联的负样本，而由于每次迭代中从其余 Web API 中随机选择非互补的 Web API，因此不同迭代中的负样本可能不同。在这种情况下，负样本可以更加多样化，从而使导出的模型更加准确。

根据互补关系和非互补关系的定义，分别生成正样本和负样本。数据标记算法如 Alg. 1 所示。Alg. 1 的计算复杂度为 $O(|M|^2PQ)$，其中 P 为 M_i 的子集的平均值，以及 Q 为这些子集中元素的平均值。

Algorithm 1 Data Labeling

Input: Mashup set $M=\{M_1,M_2,\cdots,M_N\}$ and API set $A=\{A_1,A_2,\cdots,A_n\}$

Output: Labeled sample dataset S

1: $S=\emptyset$; //initialization

[1] http://www.programmableweb.com/mashup/vizlingo

2: **for** each $M_i \in M$ **do**
3: **if** $|M_i| \geq 2$ **then**
4: **for** each nonvoid subset $M_s \subseteq M_i$ **do**
5: **for** each $A_i \in M_s$ **do**
6: $s_+ = (M_s - \{A_i\}, A_i, 1)$; //generate sample
7: $S = S \cup \{s_+\}$; // add s_+ to S
8: flag=True;
9: **while** flag==True **do**
10: randomly select $A_j \in (A - M_i)$; //local
11: **for** each $M_j \in (M - \{M_i\})$ **do**
12: **if** $(\{A_j\} \cup M_s) \subseteq M_j$ **then**
13: flag=False; break;
14: **end if**
15: **end for**
16: **if** flag==True **then** //global
17: break;
18: **end if**
19: **end while**
20: $s_- = (M_s - \{A_i\}, A_j, 0)$; //generate sample
21: $S = S \cup \{s_-\}$; // add s_- to S
22: **end for**
23: **end for**
24: **end if**
25: **end for**
26: **return** S;

B. 基于 BERT 模型的 Web API 表示

BERT（Bidirectional Encoder Representations from Transformers）是 Google 在 2018 年提出的一种预训练的自然语言处理模型，以无监督的方式利用大量未标记的文本[8]。BERT 模型建立在变压器网络结构的基础上，是自然语言处理(NLP)领域的一个突破。它在自然语言处理中的广泛应用包括文本分类、词汇表示学习、关系提取、语义相似度、问答系统等。除了其基础应用之外，BERT 在上下文理解中起着关键作用，捕捉复杂的语言细微差别和语义复杂性。它的双向编码能力使其能够全面掌握上下文信息，从而增强语言理解能力。因此，BERT 不仅在传统的 NLP 任务中表现出色，而且还使模型能够更深入地理解上下文，促进跨不同领域的更细致和准确的语言处理。

BERT 模型以句子为输入，通过融合 Token Embedding、Segment Embedding 和 Position Embedding 三个嵌入特征[8]，如 Fig. 2 所示。这些元素在 BERT 识别和表示给定句子的丰富语义细微差别的能力中起着关键作用。Token Embedding 为每个单词创建一个独特的令牌序列。这种独特的序列构成了 BERT 对单个词语义的熟练理解和对整个句子意义的贡献的基础。Segment Embedding 可以揭示一个词所属的句子语境。这种上下文意识提高了 BERT 在捕捉句子中单词之间关系方面的熟练程度，有助于对输入的整体理解。Position Embedding 将单个单词的精确位置编码成一组向量。这些位置信息进一步完善了 BERT 的理解，使其能够辨别句子中的顺序和层次结构[9]。BERT 中的编码过程反映了这些嵌入的动态相互作用，允许多种编码模式。无论是学习基于数据的编码还是遵循特定用户需求的编码规则，BERT 的适应性使其成为全面理解语言的有力工具，Token、Segment 和 Position embedding 协同工作，解开句子语义的复杂织布。

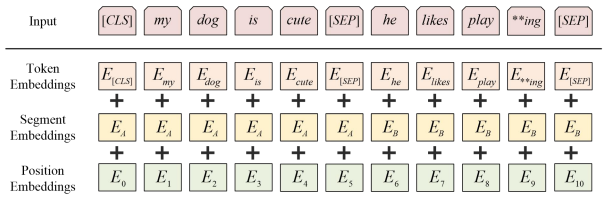

Figure 3. 一个 BERT 输入表示的例子

BERT 模型是一种预训练的语言模型，它利用上下文相关向量对冗长的文本句子进行编码，从而通过微调达到最佳性能。在预训练阶段，BERT 模型通过利用大量未标记文本获得语言表示，涉及两个并行任务:掩模语言建模任务(MLM)和下一个句子预测任务(NSP)。Task #1: MLM。在将单词序列输入 BERT 之前，一个独特的特征以[MASK]标记的形式出现，策略性地替换每个序列中 15%的单词。这一步为 BERT 的传销任务奠定了基础。在 MLM 期间，[MASK]令牌引入了不可预测性元素，因为单词暂时隐藏在序列中。该模型接下来面临的挑战包括解决语言之谜——预测被蒙面词的原始值。该任务要求模型使用围绕被遮挡元素的未改变的、未被遮挡的单词提供的上下文线索。本质上，BERT 预训练阶段的 MLM 任务作为模型的一种机制，通过推断更广泛序列上下文中的隐藏组件来推断语言的复杂性。这个动态过程增强了 BERT 的语言表示能力，使其能够熟练地捕捉输入数据中的语义细微差别和关系。Task #2: NSP。在 BERT 复杂的训练方案中，模型的输入以句子对的形式呈现，显示了语言的复杂性。这个过程的核心是 NSP 任务，这是预训练的关键阶段。在这里，模型试图提高其预测第二个句子是否与原始文档中的后续句子无缝衔接的能力。NSP 任务起着关键作用，对 BERT 的语境把握和语义连贯做出了深远的贡献，促进了对文本环境中顺序关系的高度理解。在 BERT 被预训练之后，它已经学习了单词和句子的表示以及它们所连接的潜在语义关系。基于预训练模型，BERT 可以针对特定任务在更小的数据集上进一步微调。这种战略性自定义确保与给定数据集的内容和任务目标保持一致。微调阶段改进了 BERT 的功能，增强了它在不同场景中的适应性和有效性。

一般来说，BERT 的本质是通过预训练得到原始表示。预训练模型可以通过与目标数据集的微调来适应不同的任务。在我们的工作中，利用 BERT 模型来表示 Web API 描述文档，以派生 Web API 嵌入。所选 Web API 的表示向量通过池操作聚合为单个表示向量。可以选择三种池化操作，即平均池化、最大池化和最小池化。随后，将聚合的表示向量与候选互补 Web API 的表示向量连接起来，形成下游互补 API 推荐任务的输入。

C. 基于自注意力神经分解机的互补 Web API 推荐

Factorization Machine（FM）是 Steffen Rendle 在 2010 年提出的一种值得注意的基于矩阵分解原理的机器学习算法[10]。它使模型能够有效地捕获数据中的复杂关系，为预测分析提供了强大的工具。通过合并线性项和交互项，FM 在具有稀疏数据集的场景中表现出色，使其非常适合各种实际应用程序。它理解复杂模式和依赖关系的能力使 FM 成为机器学习技术库中的宝贵资产。FMs 代表了一种监督学习方法，通过无缝集成二阶特征交互来增强传统的线性回归框架。FM 可以被定义为式 (1)。

$$\hat{y}_{FM}(\boldsymbol{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle x_i x_j \qquad (1)$$

其中 $w_0 \in \mathbb{R}$, $w \in \mathbb{R}^n$, $V \in \mathbb{R}^{n\times k}$. w_0 表示全局偏置， w_i 表示第 i^{th} 个特征对目标的权重。$\langle \cdot,\cdot \rangle$ 表示两个嵌入维度的点积，即 $\langle v_i, v_j \rangle = \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f}$，表示因式交互。$v_i$ 表示有 k 个因子的 i^{th} 变量，用于识别元素 *i* 的嵌入。在我们的工作中，我们采用了 FM 的改进版，命名为 SANFM[11]，定义为式(2)，其中前两项与 FM 一致。

$$y_{SANFM}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \hbar\ (x) \tag{2}$$

其中 $h(\boldsymbol{x})$ 是 SANFM 的核心，其中融合了因子分解、自注意力和深度神经网络。另外两项与式(1)相同。Fig. 4. 展示了 SANFM 的神经网络结构。从 Fig. 4 可以看出，SANFM 由五层组成：*Input Layer*, *Factorization Layer*, *Self-Attention Layer*, *Deep Layer* 和 *Prediction Layer*。

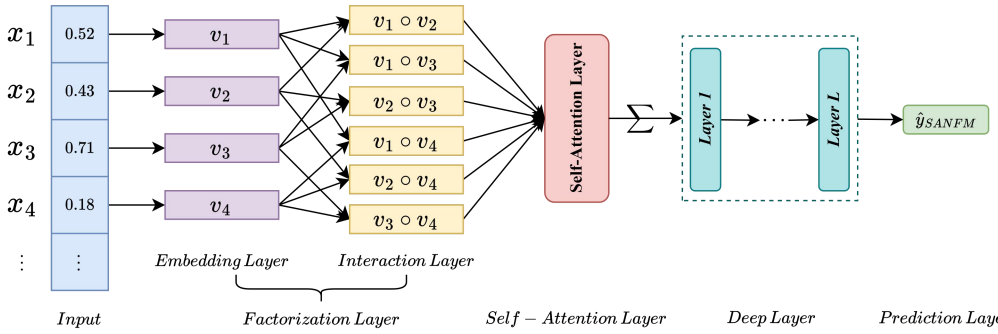

Figure 4. SANFM 的神经网络结构

分解层。为了将输入 x 的每个条目映射到嵌入向量，一组密集向量 $\mathcal{V}_x^{'} = \{v_1, \cdots, v_n\}$ 被初始化为嵌入查找表。嵌入向量变成了 $\mathcal{V}_x = \{x_1 v_1, \cdots, x_n v_n\}$。然后，将嵌入向量传递到交互层，对特征交互进行建模，定义如式(3)，其中将嵌入转换为成对的特征交互。

$$\mathcal{Z}_x = \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i v_i \odot x_j v_x \tag{3}$$

其中 $\odot$ 表示两个向量的逐元相乘，即 $(v_i \odot v_j)_k = v_{ik} v_{jk}$，由此导出了二阶特征交互。

自注意力层。由于不同的特征交互具有不同的重要性，注意机制的目标是找出不同成分的重要性，并将它们打包到单个向量中。因此，自注意力层采用自注意力机制，定义为式(4)。

$$Z_x = \text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{4}$$

其中 Z_x 是与注意力的两两互动。 Q, K, 和 V 分别是查询矩阵、键矩阵和值矩阵。它们本质上是基于交互层导出的向量的线性变换，其目的是提高模型的拟合能力。到目前为止，两两交互的重要性是通过自注意力层来区分的。

深层。为了生成高阶关系，将向量传递给一堆完全连接的神经网络，定义为公式(5)。

$$\begin{aligned} o_1 &= \sigma_1(W_1 Z_x + b_1) \\ o_2 &= \sigma_2(W_2 o_1 + b_2) \\ &\cdots \\ o_L &= \sigma_L(W_L o_{L-1} + b_L) \end{aligned} \tag{5}$$

其中 Z_x 表示加权交互向量； L 表示深层层数；W_l, b_l, 和 σ_l 分别表示权重、偏置和激活函数。对于 l^{th} 层， 由于深度神经网络的表达能力，可以学习低阶和高阶关系，最后生成预测概率。我们将 Z_L 与预测层的神经参数 h 结合。因此，预测层可由式(6)来实现。一般情况下，将 f(x) 展开为式(7)，可将上述各分量整合成结论形式。

$$f(x) = h^T z_L \tag{6}$$

$$h(\boldsymbol{x}) = \boldsymbol{h}^T \sigma_L(\boldsymbol{W}_L(\cdots \sigma_1(\boldsymbol{W}_1 Z_x + \boldsymbol{b}_1)\cdots) + \boldsymbol{b}_L) \tag{7}$$

预测层。考虑到互补 Web API 推荐可以作为一个二元分类问题，即互补性和非互补性。因此，在输出层中使用了一个 sigmoid 函数。因此，将结果转换为 $\hat{y}(x) = \text{Sigmoid}(\hat{y}_{SANFM}(x))$。对于一般的二分类问题，使用交叉熵来评估目标函数(即损失函数)来估计参数。具体而言，SANFM 的损失函数定义为式(8)，其中使用 L_2 正则化来防止潜在的过拟合。

$$L(\theta) = -\frac{1}{n}\sum_{i=1}^{N} \left[y^{(i)} log\left(\hat{y}^{(i)}\right) + \left(1 - y^{(i)}\right) log\left(1 - \hat{y}^{(i)}\right) \right] + \frac{\lambda}{2m}\sum_{j=1,\theta_j \in \Theta}^{m} \theta_j^2 \tag{8}$$

其中 Θ 为 SANFM 中模型参数的集合。 N, m = | Θ |, λ 分别表示训练集的大小、需要调优的参数集的基数和 L_2 正则化的权重。采用随机梯度下降法对损失函数进行优化 [12]。在预测阶段，最终输出是根据预测的互补概率按降序排序的前 k 个 Web API 列表。

4. 实验

为了确保对所提出的方法进行可靠的评估，我们在真实数据集上进行了广泛的实验。在本节中，我们将介绍数据集，强调其关键特征，这些特征有助于实验的现实性和复杂性。评估指标经过仔细选择，以捕获推荐模型性能的各个方面。此外，我们建立了基线方法作为比较的基准，增强了我们评估的稳健性。对实验结果的后续分析不仅仅是性能比较，还从学习模型的性能方面探讨了超参数之间复杂的相互作用。

A. 实验设置

数据集。该数据集最初是在 2022 年 1 月从 ProgrammableWeb 上抓取的。即使原来的平台已经过渡到一个新的网站，导致其原始来源无法访问。尽管如此，我们的数据集仍然在网上公开供参考和使用。Table II 提供了所选数据集的详细统计概述。该数据集包含 6,457 个 Mashup 和 1,718 个 Web API。为了提高数据处理质量，进行了严格的

预处理。具体来说，只有一个 Web API 的 mashup 被认为对有意义的分析没有用处，因此被排除在外。这个细化过程产生了一个更集中的数据集，包含 2,230 个 Mashup 和 881 个 Web API。本实验使用的数据集是基于子集的。但是，有些 Mashup 包含大量 API，导致子集过大。为了解决这个问题，在数据标记过程中对 Mashup 进行过滤。具体来说，在我们的实验中，只选择包含三个或更少 API 的 Mashup，它们的子集必须由两个或更多 API 组成。数据标记过程总共产生了 6615 个正样本。随后，将数据集分成 8:2 的比例，创建训练和测试数据集。每两轮训练随机生成一个相对于正样本的负样本。

TABLE III. 数据集特征

Dataset	#Mashups	#Invoked APIs
Crawled Dataset	6457	1718
Processed Dataset	2230	881

评估指标。在评估模型性能时，有两个标准指标，$R@K$ 和 MRR (Mean Reciprocal Rank) [13]，发挥关键作用。$R@K$ 用于估计互补 Web API 推荐方法是否能够在最优推荐结果中返回正确答案。$R@K$ 值越高表示性能越好，强调模型在最优结果中返回相关推荐的有效性。MRR 是正确答案的倒数的平均值。MRR 数值升高表示性能增强，反映了模型在向用户提供准确和优先级推荐方面的熟练程度。$R@K$ 和 MRR 被定义为式 (9) 和 (10)。

$$R@K = \frac{1}{N^+}\sum_{i=1}^{N^+}\delta(Rank_i \le k) \tag{9}$$

$$MRR = \frac{1}{N^+}\sum_{i=1}^{N^+}\frac{1}{Rank_i} \tag{10}$$

其中 N^+ 正测试样本的数量，$Rank_i$ 是真实答案在推荐列表中的排名位置，以及 $\delta(Rank_i \le k) \in \{0,1\}$ 是一个函数，如果真实答案的排名在推荐结果的前 k 个，则返回 1，否则返回 0。

基线方法。基线方法。我们将采用的学习模型 SANFM 与其他一些学习模型进行了比较，包括基于机器学习的模型和一些基于深度学习的模型。选择的基线如下：**LR** (Logistic Regression) [14], **SVM** (Support Vector Machine) [15], **GBDT** (Gradient Boosting Decision Trees) [8], **MLP** (Multi-Layer Perceptron) [16], **BasicFM** (Basic Factorization Machines) [17], **DeepFM** (Deep Factorization Machines) [18], **AFM** (Attentional Factorization Machines) [19] 和 **NAFM** (Neural and Attentional Factorization Machines)。请注意，在我们的工作中没有比较 [5, 6] 中基于关联规则的方法，因为它们只关注于推荐常用的 Web API。下面将逐一介绍基线方法。

- ✧ **LR** [14]。LR 是一种用于二元分类的统计模型，旨在预测二元变量的概率。该模型的核心思想是利用线性组合的特性，将线性组合的值转化为具有 s 型函数的概率值。输出值的取值范围是 0 ~ 1，可以理解为属于某类的概率。
- ✧ **SVM** [15]。支持向量机是一种强大的监督学习算法，用于分类任务。该算法通过在高维特征空间中寻找最优决策边界(超平面或曲面)，将数据实例划分为所属类。
- ✧ **GBDT** [8]。GBDT 是一种通过学习过程中的连续步骤来优化预测值的机器学习模型。决策树每次迭代都涉及到调整系数、权重或偏差的值来预测目标。
- ✧ **MLP** [16]。MLP 是一种神经网络，由多个神经元组成，分布在一个或多个隐藏层中，这些隐藏层通过权重连接相互关联。每个神经元对其输入加一个加权和，然后应用一个非线性激活函数来产生输出。
- ✧ **BasicFM** [17]。BasicFM 利用分解机器来捕获特征之间的复杂交互，在变量之间的关系是非线性的情况下特别有效。BasicFM 的关键原理包括使用 s 型函数将特征值的线性组合转换为概率估计。
- ✧ **DeepFM** [18]。DeepFM 将分解机器的优势与神经网络的表达能力相结合，使其能够捕获高维数据中的复杂模式和依赖关系。通过集成浅层和深层组件，DeepFM 在涉及复杂特征交互的任务中实现了卓越的性能。
- ✧ **AFM** [19]。AFM 框架将注意力机制整合到传统的因式分解机器中，从而增强了模型关注关键特征的能力，并为不同的特征交互分配不同程度的重要性，使 AFM 在处理稀疏和高维数据方面特别有利
- ✧ **NAFM** [9]. NAFM 集成了神经网络来捕捉非线性特征相互作用和注意力网络来捕捉这些特征相互作用的不同重要性，从而为数据中的复杂关系建模提供了一种全面和自适应的方法。

B. 性能评估

我们通过实验严格评估我们的方法的性能，采用 $R@K$ 和 MRR 作为测试数据集上的关键评估指标。该综合评估深入研究了 SANFM 中参数对预处理数据集的影响。所有报告的性能值代表 5 轮的平均值，确保了可靠的结果。Table IV 详细列出了 SANFM 及其对应的默认超参数值，为优化模型配置和比较分析提供了必要的见解。

TABLE IV. 超参数的默认值

Item	Value
Epoch number	50
Embedding size	32
Batch size	64
Learning rate	0.001
Self-attention size	64
Hidden layer size	5
Embedding size of BERT	768
Top k	10

性能对比。为了评估具有不同 top k 的互补 Web API 推荐结果，我们将 R@K 中的 K 设置为 2、4、6、8 和 10。对于我们的通用互补 Web API 推荐框架中的每个学习模型，通过三次聚合操作进行比较。Table V 展现了 SANFM 与第 V.A 节中提到的基线之间的性能比较。实验结果基于 MRR 升序排列，为了解不同模型在补充 API 推荐中的性能提供了有价值的见解。从传统模型开始，LR、GBDT 和 SVM 表现出相当的性能，反映了它们在捕获复杂特征交互方面的共同局限性。尽管它们很简单，但 LR 的线性方法、GBDT 的集成性质以及 SVM 处理复杂决策边界的能力都有助于它们的竞争地位。转向高级模型，MLP 引入了更强大的建模功能，展示了卓越的性能，特别是与 LR 和 SVM 相比。BasicFM 展示了有竞争力的结果，受益于它通过分解对复杂关系建模的能力。DeepFM 将深度学习与分解机器相结合，其性能优于几个基准。AFM 利用注意机制，展示了改进，特别是在通过基于注意的建模捕捉基本特征交互方面。NAFM 的集成结合了神经网络和注意机制的进步，使其与许多基线相比具有优越的性能。值得注意的是，本研究中选择的模型 SANFM 优于其他模型，表明其推荐补充 Web API 的能力更强。这可以归因于其结合深度学习技术、自关注机制和解决 Mashup-API 生态系统中的长尾效应的创新方法。自注意机制允许 SANFM 有效地捕获低阶和高阶特征交互，从而提高了性能。

TABLE V. 总体性能比较

Aggregation Operation	Learning Model	R@2	R@4	R@6	R@8	R@10	MRR
Average Pooling	LR	0.2445	0.3422	0.3971	0.4777	0.5039	0.2627
	GBDT	0.2462	0.3662	0.4314	0.5165	0.5414	0.2711
	SVM	0.2479	0.3642	0.4453	0.5068	0.5419	0.2732
	MLP	0.2343	0.3681	0.4489	0.5019	0.5368	0.2762
	BasicFM	0.2561	0.3546	0.4477	0.4849	0.5150	0.2794
	AFM	0.2687	0.3748	0.4589	0.4976	0.5261	0.2835
	DeepFM	0.2534	0.3699	0.4562	0.5001	0.5341	0.2866
	NAFM	0.2625	0.3803	0.4637	0.5133	0.5399	0.2897
	SANFM	**0.2835**	**0.3857**	**0.4683**	**0.5206**	**0.5581**	**0.3069**
Max Pooling	LR	0.2447	0.3343	0.3792	0.4563	0.4931	0.2628
	GBDT	0.2492	0.3565	0.4462	0.5102	0.5357	0.2582
	SVM	0.2405	0.3691	**0.4591**	0.5016	0.5314	0.2730
	MLP	0.2455	0.3717	0.4486	0.5044	0.5425	0.2794
	BasicFM	0.2543	0.3537	0.4236	0.4627	0.5032	0.2725
	AFM	0.2536	0.3594	0.4547	0.4950	0.5289	0.2731
	DeepFM	0.2560	0.3518	0.4481	0.5054	0.5379	0.2766
	NAFM	0.2492	0.3692	0.4470	0.5038	0.5424	0.2778
	SANFM	**0.2616**	**0.3751**	0.4578	**0.5143**	**0.5452**	**0.2872**
Min Pooling	LR	0.2429	0.3470	0.4102	0.4797	0.5142	0.2707
	GBDT	0.2370	0.3697	0.4331	0.5012	0.5309	0.2634
	SVM	0.2223	0.3661	0.4447	0.4987	0.5345	0.2614
	MLP	0.2373	0.3460	0.4253	0.4852	0.5313	0.2690
	BasicFM	0.2608	0.3648	0.4139	0.4522	0.5024	0.2770
	AFM	0.2651	**0.3739**	0.4424	0.4832	0.5193	0.2785
	DeepFM	0.2685	0.3735	0.4514	0.4932	0.5264	0.2792
	NAFM	0.2641	0.3695	**0.4580**	0.4991	**0.5353**	0.2749
	SANFM	**0.2712**	0.3721	0.4547	**0.5052**	0.5329	**0.2841**

超参数影响分析。我们深入研究了 SANFM 中四个主要超参数的影响：*嵌入维度*、*自注意力维度*、*深度神经网络层数*和*学习率*。在整个分析过程中，当一个特定参数产生影响时，其余参数保持默认值，如 Table IV 所示。此分析使我们能够全面了解每个超参数如何影响 SANFM 模型的整体性能和功能。

(1) 嵌入维度的影响。嵌入层在降维和获取有效特征表示方面发挥着关键作用。为研究嵌入维度的影响，实验将该参数在{8, 16, 32, 64, 128}范围内变化。如 Fig. 5 所示，嵌入维度的影响呈现出明显的趋势。随着嵌入维度从 8 增加到 32，R@10 和 MRR 都呈上升趋势。然而，当嵌入维度从 32 增加到 128 时，R@10 和 MRR 都出现下降趋势。当大小设置为 32 时，R@10 和 MRR 均达到峰值，实现了最佳平衡。因此，在实验中，将 32 确立为嵌入维度的默认值。

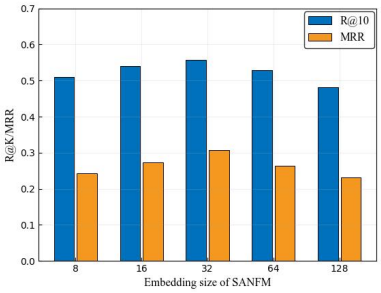

Figure 5. 嵌入维度的影响

(2) 自注意力维度的影响。自注意力机制使模型能够分配不同的权重来全面捕捉特征交互。通过研究在{16, 32, 64, 128, 256}范围内不同自注意力维度的影响，可看到在 Fig. 6 中，将自注意力维度从 16 增加到 64 会导致 R@10 和 MRR 呈上升趋势。然而，进一步增加自注意力维度从 64 到 256 时，R@10 和 MRR 都呈下降趋势。当自注意大小为 64 时，SANFM 实现了最佳性能。因此，在实验中将自注意力层维度的默认值设定为 64，选择适当的维度大小可实现有效的模型性能。

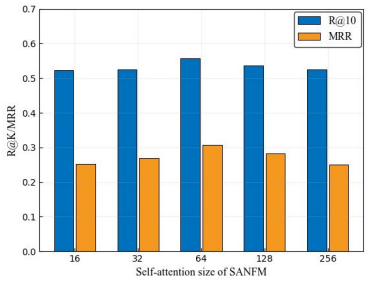

Figure 6. 自注意力维度的影响

(3) 深度神经网络层数的影响。深度神经网络层数的目的为挖掘高阶交互特征。深度神经网络层数的范围从 2 到 6 不等。如 Fig. 7 所示，当深度神经网络层数从 2 变化到 4 时，R@10 和 MRR 表现出一定的波动。当深度神经网络层数达到 5 时，模型达到最佳性能。之后，R@10 和 MRR 均呈下降趋势。这一观察结果表明，深度神经网络层数为 5 时产生了最佳结果。观察到大小为 2 到 4 的轻微波动可能归因于模型在训练期间调整其内部表示，反映了学习和适应的过程。因此，对于本实验数据集，选择深度神经网络层数为 5 可在模型复杂度和泛化能力之间实现良好的平衡。

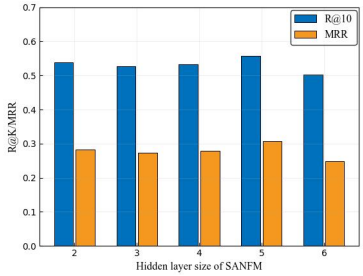

Figure 7. 深度神经网络层数的影响

(4) 学习率的影响。在深度学习模型中，学习率是一个至关重要的超参数，因为它控制着训练过程中参数更新的幅度，并显著影响着收敛速度和模型性能。由 Fig. 8 可以看出，从 1e-01 到 1e-03，随着学习率的降低，MRR 和 R@10 都增加了。从 1e-03 到 1e-05，该指标有轻微的波动，但相对于 1e-03 的学习率，该指标的值有所下降。因此，当学习率为 1e-03 时，模型性能达到最佳。因此，选择合适的学习率，既能实现快速收敛，又能避免因过高或过低的值而产生局部最优，从而对模型的性能产生深远的影响。

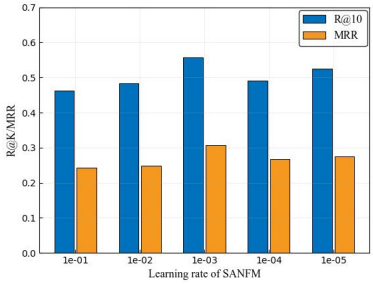

Figure 8. 学习率的影响

5. 相关工作

近年来，人们对 Web API 推荐的研究越来越关注，提出了各种方法来缓解 Web API 选择的挑战。通过对现有 Web API 推荐工作的回顾，这些工作可以大致分为四组：1）*基于内容的方法*，2）*基于协同过滤的方法*，3）*基于图的方法*，和 4）*基于关联规则的方法*。

1) 基于内容的方法 [1, 10, 11, 20, 21]. 这类工作集中于用户指定的要求。通常采用 TF-IDF（Term Frequency- inverse Document Frequency）、LDA（Latent Dirichlet Allocation）、HDP（Hierarchical Dirichlet Process）等方法利用 Mashup 和 API 的描述信息进行主题建模 [1]。在某些研究中，额外的辅助信息，如标签 [20]、网络关系 [21]等，被认为是对描述文档语义的补充，用于服务推荐、分类 [22]或聚类 [7]。尽管包含了更多的信息，但通过主题建模进行简单匹配仍然是不够的。因此，提出了一些基于改进因子分解机的方法来训练服务推荐的表达模型 [10, 11]。

2) 基于协同过滤的方法 [2, 3, 23, 24]。这类工作试图通过考虑用户已选 API 来推荐 API。在基于协同过滤的方法中，已选 Web API（通常称为历史数据）作为推荐的基础。Cao 等人 [3]提出了一种基于协同过滤的服务推荐方法，通过文本挖掘计算相似度。Yao 等人[23] 提出将隐式 Web API 关联正则化和矩阵分解相结合用于 Web API 推荐。此外，基于 CF 的 QoS 感知服务推荐因其简单、易于理解和实现而被广泛研究 [2, 24]. 尽管这些基于协同过滤的方法利用了用户和 API 之间的直接或间接连接进行推荐，但它们往往忽略或忽视了探索用户和 API 之间的高阶连接。此外，它们依赖于足够的历史数据来确保服务推荐的性能。

3) 基于图的方法 [4, 25-28]. 通常，基于协同过滤的服务推荐算法不能充分挖掘异构信息网络中的各类辅助信息。因此，利用一些网络表示方法来学习基于用户-服务交互关系的用户和服务嵌入。例如，Lian 等人 [4] 提出了一种基于神经图协同过滤的 API 推荐方法，该方法利用了历史用户-API 交互。Jiang 等 [25] 利用 Node2vec 获得 Mashup-API 交互网络的网络表示。Wang 等人 [26] 提出了一种基于 Web API 推荐的 Mashup 创建功能和结构融合模型，其中使用 LightGCN 从 Mashup-API 子图和 API-API 子图中提取 API 表示。Yu 等 [27] 采用 LightGCN 提取应用和服务嵌入向量，其中引入了对比学习和多任务学习。Tang 等人 [28] 利用异构信息网络中 API 的元路径，获取其上下文语义，用于 API 推荐。总而言之，这类方法利用网络嵌入技术为用户和服务提取更多的语义，从而实现更精确的服务匹配。

4) 基于关联规则的方法 [5, 6]。上面提到的工作主要集中在基于用户输入的 Web API 推荐场景，具体来说是通过分析所需 Mashup 的功能需求来推荐 Web API。然而，关于互补功能推荐的研究还很有限。Tang 等人 [5] 提出了一种挖掘 API 之间协作模式的方法，以帮助为物联网创建 Mashup。这项工作旨在发现在 Mashup 创建中经常组合的 Web API 类型以及流行的 API 组合类型。此外，He 等人 [6] 考虑到用户在一个开发周期内会调用多个功能，提出了互补功能推荐。这项工作将每个 Mashup 视为频繁模式挖掘的事务集，并引入了基于关联规则挖掘的互补功能推荐系统。这两项工作都采用了关联规则技术。

总而言之，基于内容的、基于协同过滤的和基于图的方法主要关注用户需求或足够的历史数据。关联规则是根据所选择的 Web API 推荐补充的 Web API，而推荐仅限于以前使用过的 Web API，而忽略了长尾 Web API 和未使用的 Web API。虽然有关于长尾服务推荐的研究 [29, 30]，但长尾服务只是被使用服务的一部分。在实践中，只有少数 Web API 被使用，大多数都未被使用。因此，为了服务生态系统的可持续发展，如果在互补的 Web API 推荐场景中平等地考虑长尾 Web API，将更加有利。这是我们工作的动力。

6. 结论

为了促进 Mashup 的发展，本文提出了一个通用的互补 API 推荐框架，其中同等考虑了长尾和未使用的 Web API。基于 Web API 之间的组合关系，通过数据预处理生成标记的互补 Web API 样本数据集，用于模型训练。同时，利用 BERT 模型，基于 Web API 的功能描述文档，推导出 Web API 的表示向量。此外，还适当设计了基于深度学习的推荐模型，以派生出互补的 Web API 推荐模型。通过标记的数据集和表示向量，可以学习最终的互补 Web API 推荐模型。在一个真实数据集上进行了全面的实验，实验结果表明，所提出的补充 Web API 推荐框架是有效的，并且 SANFM 模型优于 $R@K$ 和 MRR 指标下的基线模型。在未来的工作中，我们将尝试获得更多样化的样本，以进一步提高互补 Web API 推荐的性能。

参考文献

[1] G. Kang, Y. Xiao, J. Liu, Y. Cao, B. Cao, X. Zhang, and L. Ding, “Tatt‐BiLSTM: Web Service Classification with Topical Attention‐based BiLSTM,” *Concurrency Computation: Practice Experience,* vol. 33, no. 16, pp. 1-13, 2021.

[2] Z. Zheng, L. Xiaoli, M. Tang, F. Xie, and M. R. Lyu, “Web Service QoS Prediction via Collaborative Filtering: A Survey,” *IEEE Transactions on Services Computing,* vol. 18, no. 5, pp. 1-18, 2020.

[3] B. Cao, X. F. Liu, J. Liu, and M. Tang, “Domain-aware Mashup Service Clustering based on LDA Topic Model from Multiple Data Sources,” *Information and Software Technology,* vol. 90, no. 2017, pp. 40-54, 2017.

[4] S. Lian, and M. Tang, “API Recommendation for Mashup Creation based on Neural Graph Collaborative Filtering,” *Connection Science,* vol. 34, no. 1, pp. 124-138, 2022.

[5] M. Tang, Y. Xia, B. Tang, Y. Zhou, B. Cao, and R. Hu, “Mining Collaboration Patterns Between APIs for Mashup Creation in Web of Things,” *IEEE Access,* vol. 7, pp. 14206-14215, 2019.

[6] P. He, W. Qi, X. Liu, L. Liu, D. You, L. Shen, and Z. Chen, “Association Rule Guided Web API Complementary Function Recommendation for Mashup Creation: An Explainable Perspective,” *CCF Conference on Computer Supported Cooperative Work and Social Computing*, 2022, pp. 73-83.

[7] G. Kang, J. Liu, Y. Xiao, Y. Cao, B. Cao, and M. Shi, “Web Services Clustering via Exploring Unified Content and Structural Semantic Representation,” *IEEE Transactions on Network and Service Management,* vol. 19, no. 4, pp. 4082-4096, 2022.

[8] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, “Lightgbm: A Highly Efficient Gradient Boosting Decision Tree,” *Advances in neural information processing systems,* vol. 30, 2017.

[9] B. Liang, G. Kang, J. Liu, B. Cao, and J. Xiang, “Attentional Neural Factorization Machine for Web Services Classification via Exploring Content and Structural Semantics,” *2022 International Joint Conference on Neural Networks (IJCNN)*, 2022, pp. 1-8.

[10] S. Rendle, “Factorization Machines,” *IEEE International Conference on Data Mining*, 2010, pp. 995-1000.

[11] G. Kang, L. Ding, J. Liu, B. Cao, and Y. Xu, “Web API Recommendation Based on Self-Attentional Neural Factorization Machines With Domain Interactions,” *IEEE Transactions on Network Science and Engineering,* vol. 10, no. 6, pp. 3953 - 3963, 2023.

[12] N. Ketkar, and N. Ketkar, "Stochastic Gradient Descent," *Deep Learning with Python: A Hands-on Introduction*, pp. 113-132, 2017.

[13] C. Wang, Z. Nong, C. Gao, Z. Li, J. Zeng, Z. Xing, and Y. Liu, “Enriching Query Semantics for Code Search with Reinforcement Learning,” *Neural Networks,* vol. 145, pp. 22-32, 2022.

[14] M. P. LaValley, “Logistic Regression,” *Circulation,* vol. 117, no. 18, pp. 2395-2399, 2008.

[15] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, “Support Vector Machines,” *IEEE Intelligent Systems and their Applications,* vol. 13, no. 4, pp. 18-28, 1998.

[16] M.-C. Popescu, V. E. Balas, L. Perescu-Popescu, and N. Mastorakis, “Multilayer Perceptron and Neural Networks,” *WSEAS Transactions on Circuits Systems,* vol. 8, no. 7, pp. 579-588, 2009.

[17] B. Cao, J. Liu, Y. Wen, H. Li, Q. Xiao, and J. Chen, “QoS-aware Service Recommendation based on Relational Topic Model and Factorization Machines for IoT Mashup Applications,” *Journal of Parallel and Distributed Computing,* vol. 132, no. 2019, pp. 177-189, 2019.

[18] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, “DeepFM: A Factorization-Machine based Neural Network for CTR Prediction,” *International Joint Conference on Artificial Intelligence*, 2017, pp. 1725-1731.

[19] Y. Cao, J. Liu, M. Shi, B. Cao, T. Chen, and Y. Wen, “Service Recommendation Based on Attentional Factorization Machine,” *IEEE International Conference on Services Computing*, 2019, pp. 189-196.

[20] M. Shi, Y. Tang, and J. Liu, “TA-BLSTM: Tag Attention-based Bidirectional Long Short-Term Memory for Service Recommendation in Mashup Creation,” *International Joint Conference on Neural Networks*, 2019, pp. 1-8.

[21] M. Shi, Y. Tang, X. Zhu, and J. J. A. T. o. t. W. Liu, “Topic-aware Web Service Representation Learning,” vol. 14, no. 2, pp. 1-23, 2020.

[22] X. Zhang, J. Liu, M. Shi, and B. Cao, “Word Embedding-based Web Service Representations for Classification and Clustering,” *2021 IEEE International Conference on Services Computing (SCC)*, 2021, pp. 34-43.

[23] L. Yao, X. Wang, Q. Z. Sheng, W. Ruan, and W. Zhang, “Service Recommendation for Mashup Composition with Implicit Correlation Regularization,” *IEEE International Conference on Web Services*, 2015, pp. 217-224.

[24] Z. Wang, C.-A. Sun, and M. Aiello, “Context-aware IoT Service Recommendation: A Deep Collaborative Filtering-based Approach,” *2022 IEEE International Conference on Web Services (ICWS)*, 2022, pp. 150-159.

[25] B. Jiang, H. Li, J. Yang, Y. Qin, L. Wang, and W. Pan, “Web Service Recommendation Based on Word Embedding and Node Embedding,” *Mobile Information Systems,* vol. 2022, no. 2022, pp. 1-12, 2022.

[26] X. Wang, M. Xi, and J. Yin, “Functional and Structural Fusion based Web API Recommendations in Heterogeneous Networks,” *IEEE International Conference on Web Services*, 2023, pp. 91-96.

[27] T. Yu, L. Zhang, H. Liu, H. Liu, and J. Wang, “Service Recommendation based on Contrastive Learning and Multi-task Learning,” *Computer Communications,* vol. 213, pp. 285-295, 2024.

[28] M. Tang, F. Xie, S. Lian, J. Mai, and S. Li, “Mashup-oriented API Recommendation via Pre-trained Heterogeneous Information Networks,” *Information Software Technology,* vol. 169, pp. 107428, 2024.

[29] D. Yu, T. Yu, D. Wang, and S. Wang, “Long Tail Service Recommendation based on Cross-view and Contrastive Learning,” *Expert Systems with Applications,* vol. 238, pp. 121957, 2024.

[30] B. Bai, Y. Fan, W. Tan, and J. Zhang, “DLTSR: A deep learning framework for recommendations of long-tail web services,” *IEEE Transactions on Services Computing,* vol. 13, no. 1, pp. 73-85, 2017.