API Recommendation for Mashup Creation: A Comprehensive Survey

Hadeel Alhosaini[1,2],Sultan Alharbi[1],Xianzhi Wang[1],Guandong Xu[1]

[1] *University of Technology Sydney, NSW 2007, Australia*
[2] *University of Jeddah, Jeddah, Saudi Arabia*
Email: hadeel.alhosaini@student.uts.edu.au

Mashups are Web applications that integrate multiple Application Programming Interfaces (APIs) to reuse existing resources and expedite software development maximally. API recommendations play a critical role in assisting developers in building such Web applications easily and efficiently. And the proliferation of publicly available APIs on the Internet has inspired the community to adopt various models to accomplish the recommendation task. Until present, considerable efforts have been made to recommend the optimal set of APIs, delivering fruitful results with varying performance in the recommendation accuracy. This paper presents a timely overview of the topic of API recommendations for mashup creation. We investigate and make comparisons between not only traditional approaches based on data mining and recommendation techniques but also more recent approaches based on network representation learning and deep learning techniques. Analyzing the merits and pitfalls of existing approaches, we pinpoint a few promising directions to resolve the remaining challenges. This survey provides a comprehensive overview of the API recommendation research and could be a useful reference for relevant researchers and practitioners.

Keywords: API recommendation; Collaborative Filtering; Network Representation Learning; Deep Learning; Future Directions

1. INTRODUCTION

The revolution of the World Wide Web infrastructure and the development of Web 2.0 technology has drawn the advancement of building innovative Internet software systems and multiple techniques to share resources. Enterprises and organizations have been attempting to apt to encapsulate their business applications into numerous lightweight blocks (Web APIs) to be remotely accessed by lots of potential users or customers. As a result, they brought numerous benefits to the Service-oriented computing (SOC) community and have been widely employed by a wide range of consumers [1].

Application Programming Interfaces (APIs) are a set of programming standards that allow interaction with external applications to facilitate software development. They have significantly helped developers accomplish their software construction efficiently, faster, and easier manner [2]. They can avail web developers by integrating or customizing advanced functionalities into their websites and as well can be profitable and a powerful marketing tool for their providers.

They have emerged in several industries such as cloud computing, mobile applications, Internet of Things (IoT) services, machine learning, and big data services. Hence, the dramatic proliferation of published web APIs by developers has imposed the need for competent storage to keep up with its high demand and usage and for efficient searching and recommendation techniques. Also, the number of invoked web APIs has grown huge in the past few years, and it started to become prevalent in several applications.

Mashup, a new development technique that was brought on by API; is an approach that allows developers to compose several, e.g., web APIs, to facilitate Web application creation. Mashups have become popular due to their features, yet due to the dynamic nature of web APIs and their rising number, the process of manually selecting a set of web APIs for mashup service is a difficult task for software developers.

Traditional mashup recommendation systems such as collaborative filtering and content-based filtering techniques have been widely applied, for instance, matrix factorization in [3, 4], and context-aware methods in [5]; as well as the hybrid approaches that merge both of the previous popular techniques [6, 7, 8, 9] to leverage historical records and different data to assist capturing the users' behavior and items

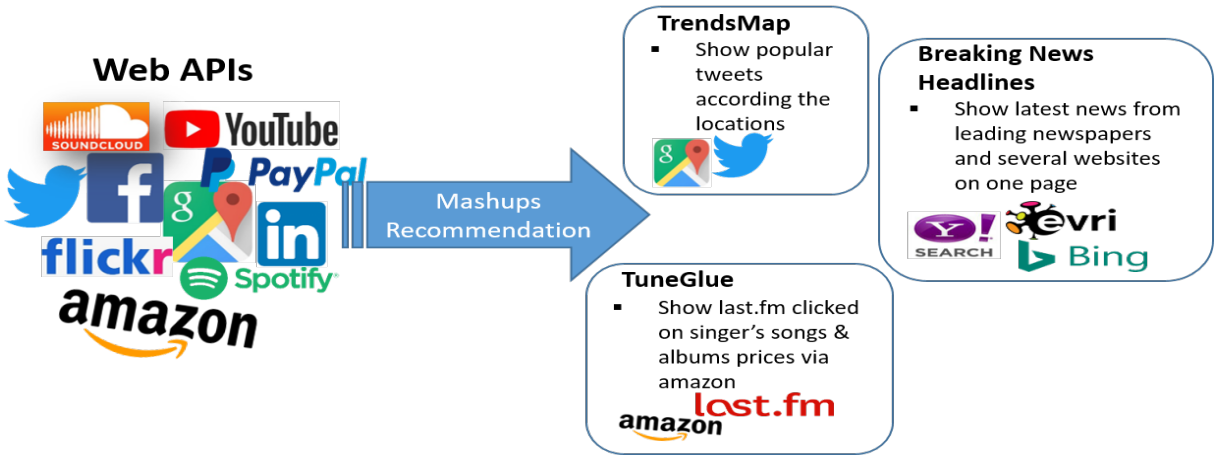

FIGURE 1. API Mashups Recommendation

features. Hybrid models have been used nowadays to alleviate issues like cold start and generate accurate recommendations. Traditional techniques have gained popularity yet still suffer serious issues like cold-start. Other studies [10, 11, 12] have endeavored to integrate deep learning methods, i.e. Doc2Vec, Deep-FM, LSTM, Attention Mechanism, etc, in their architectures implementation to learn the complex interaction among web APIs and tackle some of the traditional recommendation systems data issues like sparsity.

Though there were numerous attempts, there is still a need for an efficient mashup recommendation system, Fig. 1, to help select the potential relevant and diverse APIs for mashup development. The system should be able to identify the user(or previously built mashups) behaviors and the complex item features and interactions. It should also provide leverage by automatically suggesting and ranking top-K sets of API to suit developers' demands [13] best. Further, it is a crucial element that the system should be able to comprehend the type of APIs the users are requesting.

Platforms like Programmableweb[3] directory and several others can be very beneficial as they were established to offer open APIs storage for further analysis. They store a wide variety of readily available web APIs and their documents along with previously created mashups. The availability of such platforms encourages service clients and providers to communicate and share value-added composite services easily and quickly as well as provide the recommendation systems developers with input data.

There is a significant advancement in the computing community, which has positively affected the mashup recommendation systems in some aspects. Xu *et al.* [14] have inspected the characteristics and the disparity of several traditional mashup recommendation systems, i.e. the content-based and collaborative filtering techniques. They also presented the potential future directions researchers may be interested in. Yet, there are various other directions of recommendation techniques that can be further explored.

The adoption of recommender systems helps alleviate some of the challenges arising in mashup creation. Therefore, mashup recommender systems can assist developers in identifying suitable APIs based on the textual description of their requirements from collections of available Web APIs. The availability of reliable, effective recommender systems eases the process of building mashups and saves time and effort in software development. To our knowledge, no fully updated and comprehensive survey has investigated the APIs and mashup recommendation approaches. Our contribution in this paper is to highlight the growth of APIs and mashup recommendation systems and analyse the advantages and drawbacks of their current techniques.

Furthermore, we will discuss a few trends and directions in recommender systems research later in this article. The prospects of these approaches can be investigated further and applied to API recommendation systems. The implementation of explainable recommendation systems for API mashup building, for instance, can provide both a means of providing recommendations that satisfy user needs and also help them make an informed decision by providing insights into the reasoning behind the recommendations.

This study is organised as follows. A brief history and the basics of the field and is conveyed in Section 2. The initial procedures of searching the research area and obtaining a shortlist of studies to be examined are discussed in Section 3. In Section 4, we present different types of implemented APIs and mashup recommendation systems. Section 5 presents the potential future directions researchers are currently tempted to take in their research; finally, the paper conclusion is drawn in Section 6.

[3]http://www.programmableweb.com/

2. PRELIMINARIES

2.1. APIs and Mashups

Application Programming Interfaces, commonly cited as Web APIs, refer to the published services on the web that can be called by users via HTTP requests and responses. It enables the data transmission between software and another, in Fig. 2. As a software requests to access specific information or functionality from another software, it calls its API with the specified requirements on how to obtain the demanded data/functionality. There are two mechanisms for communication via HTTP used to build Web APIs and assist in exchanging messages between applications called Simple Object Access Protocol(SOAP) and Representational State Transfer(REST). While the former is a protocol that is structured, designed to be lightweight, and mainly uses XML format, the latter is an architectural style that represents a client-server architecture, is more flexible, and can support various other data formats. There are other technologies that can be used for communication; however, when it comes to security, scalability, and fault-tolerant distributed systems, SOAP and REST are more efficient. [15] The decision to select either one of them depends on the API users' requirements and should follow some criteria such as the overall application design, security, consistency, etc.

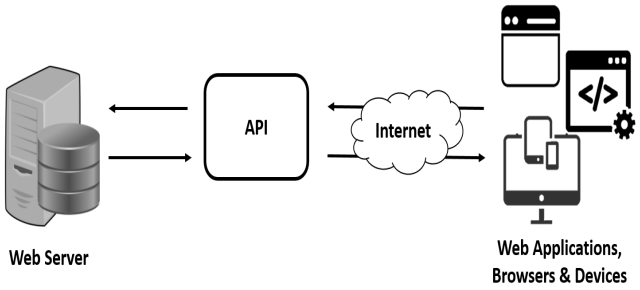

FIGURE 2. API Usage Setup

Since the 2000s, APIs have been of interest to researchers for their significance in interconnecting people, applications, and systems. Their types may vary based on their availability or use cases. The usability of APIs benefits not only the users but also programmers in some instances. Software developers don't have to grasp the knowledge of the source code of these APIs; they need to understand how APIs generally operate. The API specifications aim at standardizing the data interchange between web applications. The final documentation on integrating APIs into systems needs to be structured and straightforward to satisfy and attract the users' attention. Nowadays, the role of APIs not only in the community of software development but also in the collaboration of businesses is increasing and has become critical; hence, they need to be carefully selected to meet expectations.

With the advent of open APIs and data sources, a new trend toward web mashups has emerged in an effort to speed up and simplify the development process. The approach is defined as the composition of selected APIs that meet specific criteria to fit the demands. Mashups are generally data-centric and lightweight applications that are created to have a particular purpose. The notion of reusing pre-implemented functions of already published APIs has always enticed developers for its advantages.

The history of API Mashups began over a decade ago, and they have attracted wide attention in the software community. Mashups have diverse categories with distinct features, the most popular ones like mapping, video/photo, weather, shopping, etc. Early mashups for example aimed at utilizing Google Maps with their data for consumer use, and over the years, the number of elected APIs has increased to more than two APIs. As mashups have gained enormous support from various platforms, recommendation techniques were developed to assist the mashup creation.

2.2. Recommender Systems

Recommender Systems (RS) aim to help users discover a wide range of relevant products and solutions and select the best to suit their purpose. They can be applied in several domains, such as entertainment, research, social networks, software engineering, etc. Many recommendation systems have been applied to medicine and drug recommendations. For instance, Zhang*et al.* [18] have introduced a novel algorithm named LEAP, which captures the interdependence of diseases and medications, aiming to sequentially suggest the most suitable treatment plans for patients.

In another study, Wang*et al.* [19] have incorporated the advantages of both the supervised and reinforcement learning approaches in an approach named SRL-RNN that is designed to provide recommendations for treatment plans in dynamic medical contexts. In addition, Wang*et al.* [20] have presented a dynamic graph convolutional reinforcement learning framework named Combined Order-free Medicine Prediction Network (CompNet) as another means for medicine prediction. Moreover, various recommender solutions in the medical field have further attempted to focus on the safety aspect of the recommendation outputs, such as Wang*et al.* [21] where they proposed a novel model named Safe Medicine Recommendation (SMR) to learn the complex relations between diseases, patients and medicine for safe medicine recommendations for new patients.

There were several solutions attempting to utilize the adversarial Drug-Drug Interaction (DDI) knowledge graph along with EHR data for the recommendation. For example, Symeonidis*et al.* [22] used them for safe and explainable recommendations of medication

combinations for patients. Symeonidis*et al.* in their work [23] also attempted to predict the mortality of critically ill patients and identify any associated significant predictors. This study [24] has further utilized the Post Hoc Re-rank technique to minimise the risks and safely recommend the drug prescription for patients.

Similarly, Symeonidis*et al.* [25] have detailed an extended version of Matrix Co-Factorization (MCF) to include additional auxiliary data and a Post Hoc Re-Ranking technique with the purpose of learning the drug-drug interactions and generate safe medicine recommendations for patients. Others like Shang*et al.* [26] implemented an end-to-end framework called Graph Augmented Memory Networks (GAMENet) that incorporated Drug-Drug interactions knowledge graph and patient records to offer safe and personalized recommendations.

As online news platforms grow, news recommendation systems are also becoming popular. For instance, Symeonidis*et al.* in their study [27] have employed the Markov chain modelling to assist in generating session-aware news recommendations by analyzing the long-term preferences of the user over items in recent sessions.

In general, traditional recommender systems are characterised as either Content-Based Filtering (CBF), Collaborative Filtering (CF), or Hybrid Filtering (HF). The Content-based filtering methods consider the previous choices users have either bought or were interested in and recommend those relevant items. These systems depend on a single user's preferences and item ratings. As a result, they do not perform well with a few numbers of ratings. Moreover, the large datasets can be could cause difficulty in implementing such a system [28].

While CBF is attentive to the item description, the collaborative filtering (CF) approaches are much more drawn into constructing users' profiles. CF is built to recommend those liked items of similar users, those who are close to him in some way. It examines user-item historical data and users' relationships to draw the similarity in-between. Its systems are divided into two genera: memory-based approaches and model-based approaches. Memory-based CF techniques tend to study the users' similarities based on cosine similarity or Pearson correlation. The model-based methods use data mining or machine learning algorithms to find the rates of unrated items such as the latent semantic models and the Bayesian networks. CF has been hugely adopted in several studies and is considered the most common one.

Hybrid filtering models are another basic one that takes advantage of combining two or more of the above recommendation techniques in several ways to enhance the overall accuracy and efficiency. Though many studies have attempted to prove its benefits and how it overcomes the challenges of previous methods, it is still argued that it adds more complexity to the process; therefore, many find it strenuous to implement.

Despite the popularity and effectiveness of these traditional methods, they have been limited by several challenges such as data sparsity, cold-start, and diversity [29]. To overcome the stated problems, many papers have proposed other recommendation models as the ones based on the use of deep learning. The goal is to enhance the current methods in terms of their accuracy and diversity. In this paper, there will be an inspection of the published research on traditional and modern models related to API recommendation.

2.3. API Recommendation Procedure

2.3.1. Requirements Elicitation

Requirements engineering (RE) is referred to the critical steps of defining, documenting, and maintaining requirements in software development. The key activities of the RE process are elicitation & definition, quality assurance, negotiation, and release planning [30]. The poor execution of the requirements engineering steps would eventually lead to the failure of the established software.

The requirements elicitation phase aims to collect the system requirements from distinct stakeholders. The final output is textual documentation describing software requirements, scenarios, use cases, expected prototypes, or all of the previous. As recommender systems are a type of system that monitors and aggregates various patterns of information to conclude the final output. There are several key characteristics that distinguish the details of requirements of typical recommender systems from those of domain-dependent systems, e.g. API mashup recommender systems.

The next step would be to identify the scope of the problem and the expected solution requirements. It can be investigated by surveying customers/users or researching the area of research thoroughly. The process of gathering data from users can be classified into two approaches: explicit and implicit. While the former approach involves obtaining the data directly from users, such as by completing forms or questionnaires, the latter data is collected by monitoring user behavior or any information traces.

The purpose of an API mashup recommender system could vary from one enterprise to another; for example, some may be set only to recommend API mashups with regard to the stated project features, while others may demand the list to be ranked following specific standards. The appropriate recommendation technique would be selected according to the recommender model's articulated and verified specifications.

2.3.2. Method Preparation

The recommendation model preparation includes multiple steps of planning. One of them is to identify

the task of API recommendation and the challenges to be addressed. In addition, it includes the process of selecting the appropriate recommender algorithms and defining the testing methodology for evaluating their performance. Each recommendation technique varies in setting, implementation, and performance; selecting any of them can be based on the application's specific requirements, popularity, or effectiveness. For example, query-based approaches aim to build a knowledge base of all existing API candidates from the available data sources to retrieve query-relevant API selections. In addition, the decision for selecting the methods and testing metrics may involve other factors, such as the size and type of the available dataset. Metrics such as recall, precision, fall-out, etc., can also be selected depending on the purpose of the evaluation.

It is significant to learn the features and interactions of APIs and Mashups datasets for link prediction and recommendation procedures. There are various means for building datasets, and in API recommendation systems, the most effective way to do so is to crawl online repositories such as ProgrammableWeb. A properly established method for mashup recommendation should fully utilize the dataset to achieve the aims of the system. In Section 4, we will discuss some of the techniques selected for API recommendation models and how they use input data.

2.3.3. The Recommendation Model

The framework of API recommendation is designed to assist the creation of mashups by recommending a list of API candidates given descriptions of the mashup. Typically, the system is expected to take in a set of requirements from developers, and in return, the recommended mashup should be able to fulfil the purpose of their application. As stated, specifically, the main goal of mashup recommendation models is to recommend a set of Mashups (M), API compositions, of several APIs (A) to appease users' needs. For instance, the top restaurant mashup in Fig. 3 is built to search for restaurants with top reviews and better feedback as well as to identify the ones located close to the users. For mashup to provide the user with a list of restaurants, it includes web services like WordPress, Google Maps, Twitter, Yelp, and Web Search.

Developers provide a set of key terms or a requirements document, which includes terms that describe functionalities needed for their software development for the recommendation model. The model would exploit the specified terms in their approach to output a list of top API recommendations for mashup creation, Fig. 4. The applied techniques for API compositions differ in function, and each has benefits and drawbacks. The selection of the most appropriate one depends on the key purpose of the developer's model and should pay attention to the accuracy and diversity of recommendations.

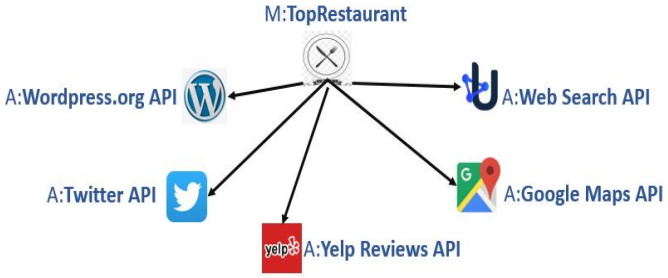

FIGURE 3. Top Restaurant Mashup Example

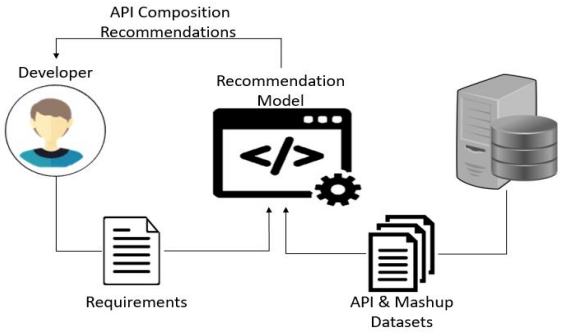

FIGURE 4. API Recommendation

3. RESEARCH METHODS

3.1. Search Process

This primary step is performed in an effort to collect the list of relevant studies and involves various tasks, such as identifying the primary data sources, constructing the search key terms, and utilizing the search procedures. These different databases, as in Table 1, are used to conduct the literature search and selection of papers within the scope of the article. These repositories were selected based on their comprehensiveness, quality control, subject specificity, and the ability of citation tracking. Furthermore, the advanced search functionalities offered by these databases will greatly facilitate the process of conducting searches.

TABLE 1: Databases Information

Database Name	Database URL
Google Scholar	https://scholar.google.com.au/
IEEE Xplore Digital Library	https://ieeexplore.ieee.org
ACM Digital Library	https://dl.acm.org/
ScienceDirect	https://www.sciencedirect.com/
Springer	https://link.springer.com/

Various search terms were used to search the existing literature from the databases. The selected key terms are designed to be relevant to the scope of the article. In order to capture the related academic

papers, the final query terms are constructed by using the Boolean operators, e.g. AND, OR (For example: "Web API recommendation", "APIs" OR "services" AND "mashup" AND "recommendation").

Furthermore, the period of the published papers is selected between 2014-2022, and the main focus was on recent publications within the last seven years. The initial search filtration process on the selected databases has yielded countless academic papers yet few are selected. The list of papers will be further filtered out in the next step based on the inclusion and exclusion to obtain the final selection outcome.

3.2. Data Filtering

In this stage, we mainly used the defined inclusion and exclusion criteria to lead the search procedure to select the most relevant papers, they are applied as follows:

- **Inclusion criteria:** The studies included in the literature review need to meet the following inclusion criteria:
  Criteria 1: The paper should discuss API recommendations for mashups.
  Criteria 2: It must have been published between 2014-2022.
- **Exclusion criteria:** If any study meets the following criteria, it will be excluded from the literature process:
  Criteria 1: The scientific document is written in a language other than English. =
  Criteria 2: It is published as a thesis, book, or book chapter.
  Criteria 3: The paper does not relate to web API or web service mashup recommendation systems.
  Criteria 4: It is a duplicate of any previous studies.

During the search process, it has been observed that certain articles occasionally utilize the terms "API" and "service" interchangeably. The output shortlisted articles are first scanned for titles and abstracts, then filtered to include only related papers and remove duplicates. The final number of studies resulted in more than 68 papers, and they are organized across distinct sections of the document.

3.3. Data Analysis

In the last few years, there has been an increase in the number of published research articles on API and mashup recommendations. In this paper, techniques are organized into four main components: Data Mining Techniques, Traditional Recommender Systems, Network Representation Learning, and Deep Learning Techniques. As Figure 5 expresses the distribution of the cited articles, it also indicates the popularity of traditional recommender systems. Deep learning approaches come in second on the list, and as noted here recently, they are becoming widely applicable.

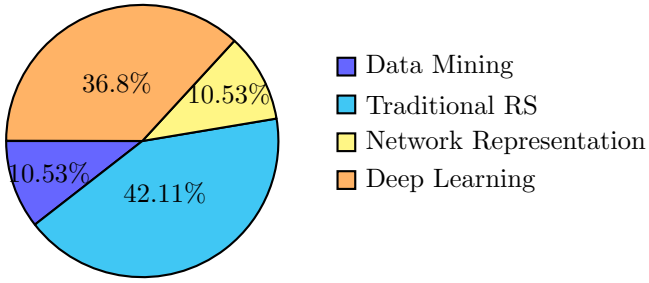

FIGURE 5. Studies Distribution across Sections

4. DISCUSSION OF WEB API RECOMMENDATION MODELS

In recent years, web-based APIs, also known as Application Programming Interfaces (APIs), have become an essential tool in software development. Yet, the increased number of Web APIs has added a new challenge for the developers to be able to choose the appropriate ones among the huge set of APIs to invoke and assist in the creation of new mashups. According to our quick analysis of the ProgrammableWeb site in 2021, Fig. 6, the shown number of APIs which was invoked in mashups by developers is still relevantly less than expected and despite all of the efforts spent so far, the challenge of efficiently utilizing APIs in mashups still exists. Various research on recommending the relevant set of web APIs and services have emerged in the field of service computing. The existing methods and research trends in this area can be distributed into numerous categories such as collaborative filtering, functional properties-based, content-based, and deep learning-aided recommendation systems.

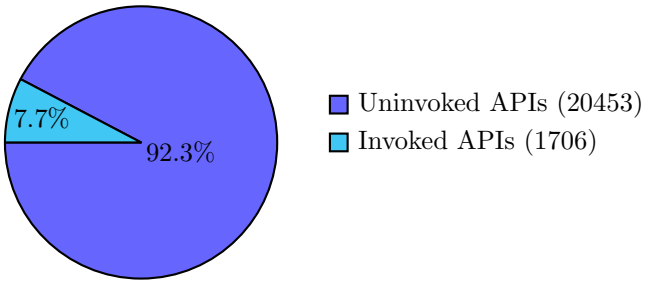

FIGURE 6. Percentages of Invoked and Univoked APIs in mashups(based on Programmableweb site analysis)

Several of these traditional systems are evolving to overpower issues like data sparsity. For instance, those who are functional-based APIs, which focus on the web services, and user queries functional-based similarity matching, can be limited by the short services' description while constructing mashups. Therefore, some have attempted to enrich the description using multiple data sources. Still, most of these methods fail to capture the essential features or most relevant function terms. As a result, it is not guaranteed whether

they can be applied in every situation.

Recently, content-based recommendation with deep learning techniques has attracted the attention of numerous researchers. They can learn the items' features automatically and therefore achieve further success with the recommendation. Yet, their approaches can't capture the elements' dependencies and assign distinct weights to these items.

Similarity computation solutions can be impractical in large-scale settings, and also the APIs description sparsity adds a new challenge. Therefore, further research has put forward other solutions like the semantic-based models for improving the recommendation in which it would examine the APIs' textual description for further analysis. Several have pursued the usage of APIs textual documents for constructing graphs to fulfill the requested queries in almost natural language. It helps easily with catching the interaction patterns between APIs for building proper mashups.

Thus, many have argued that the lack of semantic descriptions may have imposed difficulty in implementing these methods. They still tried to cope with the issue by developing approaches that can help capture the relevant details of web APIs and mashups to enrich these APIs' descriptions.

With the revolution and the emergence of many APIs recommendation systems mechanisms, it is essential to have a comprehensive review for further assessment and improvement. In the coming subsections, we will discuss the tactics of multiple recommendation methodologies alongside their benefits and drawbacks.

4.1. Data Mining Techniques

Machine learning (ML) is a subset of the Artificial Intelligence (AI) applications in which its methods automatically, with no human intervention, help computer programs access data and learn its common patterns to assist in making decisions and improve users' experience. Since ML algorithms are influenced by data, they have been widely applied in data science. They range from supervised, unsupervised, reinforcement learning, and so many others.

Data mining and analytical techniques have been positively influencing the process of Web APIs discovery and recommendation. The Association rule algorithm is a popular ML rule-based method that sets the model rules in accordance with previously identified relationships and correlations between variables. It has been adopted in various recommendation studies. For example, in Thung *et al.* [32] work, the association rule mining technique was employed as a component to extract APIs libraries' usage patterns and, therefore, assist in building a recommendation framework based on previous similar projects. The FP-growth-based association mining algorithm is a popular method that was employed by Tang *et al.* [33] as their approach to capturing the existing collaboration patterns in the stored APIs textual descriptions and tags for future mashup creation. Their work has also discussed the expansion of API tags as a potential solution to the sparsity problem.

Another common method is the clustering technique which tends to offer an efficient search mechanism by grouping APIs in different clusters using similarity measures and therefore helps reduce their number. Clusters of APIs can be built as topics similarly, e.g., Cao *et al.* [34] have implemented a domain-aware mashup service clustering method. The method models data topics by calculating the service topics' similarity to enhance the mashup creation process. Yet, their work does not perform well in a heterogeneous network information setting since they handle service documents independently.

A multi-description topic-based service clustering framework (MDT) was presented by Hu *et al.* [35]. Their technique tends to learn the topics from services' descriptions and composes clusters of the related ones. Therefore, it can help to discover the most topic-similar set of services to the input queries. An attempt to enhance traditional web service clustering algorithms' accuracy was proffered by Liang *et al.* [36] in their novel web service co-clustering approach that employs both services WSDL documents and tags, named WTO. In order to avoid tagging problems such as overly personalized and ambiguity, they have also brought forward two tag recommendation methods for enhancing the web service tagging process and data quality.

A lot of researchers have studied regression analysis as an approach to explore and model relationships between one or more variables to be used for mashup suggestions. Regression models are frequently used in predictive analysis to forecast users' items and mashup preferences. Zhao *et al.* [37] have proposed a regression analysis-based Web API recommendation method that extracts and merges trivial features such as mashup-API invocation, textual, API tags, etc... to estimate the relevance between mashups and APIs. The model further utilizes a learning-to-rank method to rank and recommend APIs and recommend APIs.

Traditional recommendation systems have failed to exploit the ever-growing and dynamic APIs data and cope with the various business demands. They need massive computational resources and time to achieve a considerable amount of accuracy. Consequently, new approaches have been studied to overcome the imposed challenges and incorporate further enhancement.

Transfer Learning, Fig. 7, is a methodology wherein knowledge acquired from solving a certain problem can be applied to solve another one. The information gained from recommending a couple of APIs according to the developers' needs can be used to recommend alike helpful APIs since many have found not to be invoked yet.

Recent work, i.e., Lei *et al.* [38] has introduced

TABLE 2: Data Mining Techniques Studies

Studies	Algorithm	Specifications
Thung *et al.* (2016)	Association Rule	APIs libraries Usage Patterns Extraction, Personalized Ranking Model
Tang *et al.* (2019)	Association Rule	FP-growth-based association algorithm, textual descriptions & tags
Cao *et al.* (2017)	Clustering Algorithm	Domain-aware mashup service clustering, topics similarity
Hu *et al.* (2019)	Clustering Algorithm	Multi-description topic-based service clustering framework (MDT), services descriptions
Liang *et al.* (2019)	Clustering Algorithm	Web service co-clustering approach, services WSDL documents and tags
Zhao *et al.* (2019)	Regression Analysis	Mashup-API data analysis, learning-to-rank method
Lei *et al.* (2020)	Transfer Learning	Latent Dirichlet Allocation (LDA) and word2vec techniques, matrix factorization, service descriptions
Liu *et al.* (2020)	Transfer Learning	Natural Language queries, Two-Steps Transfer Learning Steps, Deep Learning Model

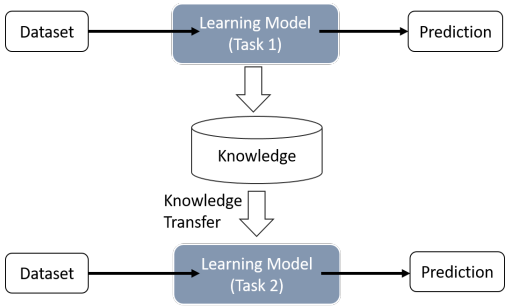

FIGURE 7. Transfer Learning Model

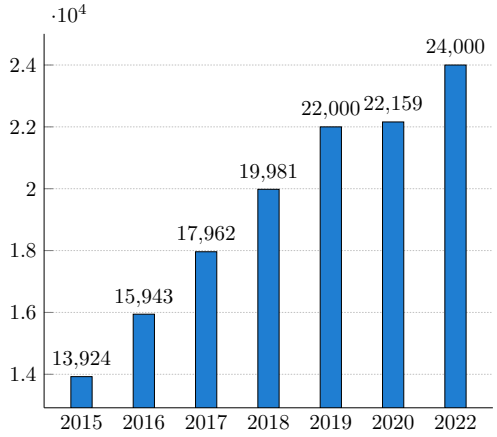

FIGURE 8. APIs Approximate Counts (based on Programmableweb Site)

transfer learning to help to alleviate the data scarcity challenge. They use word2vec and Latent Dirichlet Allocation (LDA) techniques to analyze the service descriptions for their content and location details interrelationship. External information, i.e., location here, with latent learned topics are used in the architecture to regularize MF for better results. Moreover, there were some attempts to build an intelligent system [39] that infuses natural language queries and transfer learning to train the Web API search and discovery model. In addition, it proposed a method to help collect useful information on APIs from their documentation as the model depends on the collected data, such as the Web API description.

4.2. Summary

Machine learning does support training systems based on previous experience to provide preferred recommendations. Its techniques, in Table 2, can be combined with others to gain further advantages in various aspects and help predict the user's best interests and therefore offer high accuracy recommendations [40]. In the era of big data, traditional ML technologies have fallen short of performing efficiently. Since APIs are skyrocketing Fig. 8, there was an urgent need for more robust techniques to handle the big rise of information. Though transfer learning has brought many advantages, it cannot be trusted to recommend APIs accurately in every scenario.

4.3. Traditional Recommender Systems

4.3.1. Collaborative Filtering (CF)

The collaborative filtering (CF) technique is a very common approach used by numerous recommender systems to predict potential users' interests based on neighbours' preferences, Fig. 9. Commonly, it can be noticed in movie recommendation applications where they tend to find users with a similar-like taste and suggests movies that appeal to them [42].

Its derived models can be either memory-based methods or model-based methods. The memory-based models are restricted by the accuracy of similar neighbors or item selection; therefore, they can be

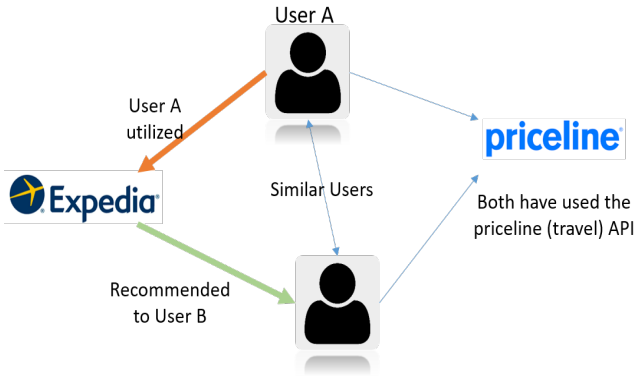

FIGURE 9. Collaborative Filtering Technique

limited by the high data sparsity challenge in historical records. They face various challenges when dealing with large sets of data. On the other hand, model-based methods do perform well when a sparsity problem exists. They basically utilize various machine learning algorithms to predict unrated items of users. They tend to use historical records and items' features to acquire latent representations and reduce data dimensions.

Matrix factorization (MF) is one of the most popular CF models. It has the ability to learn the latent features of both users and items from their stored historical interaction records. It is widely adopted by search engines and recommendation systems to generate an output. Yin *et al.* [3] developed a personalized holistic recommendation framework with a data collection model. It includes a joint of the matrix factorization with cognitive knowledge mining to study the hidden relationships among users and APIs and their types. The model does handle not only the historical records but also the content information of APIs. Similarly, Xu *et al.* [4] have proposed a joint matrix factorization technique that employs different types of information on both users and APIs for prediction.

Joining MF and popular clustering approach has been widely applied in research, i.e., in [43], they have pursued employing the matrix factorization (MF) along with the ICNC approach, named ICNC-MF, for Web APIs recommendation. It applies MF with the associated web APIs clusters and ranks them to produce the final results. ICNC-MF outperforms other baseline methods in terms of accuracy and diversity. Though this study does not overlook the least standard and unpopular APIs, unlike the previous studies, it still focuses on popularizing them with no such consideration for other factors. Moreover, this method utilizes the Pearson correlation coefficient (PCC) as its similarity function, which proved to not performs well when the mashups-APIs invocation matrix becomes sparse. Unlike traditional MF techniques, where neighborhood information is neglected and not considered with weights assigning, a novel model was implemented in [44] that integrates neighborhood information in the weighting mechanism and checks both the accuracy and diversity of recommendations.

The rapid growth of similar functional Web APIs has led researchers such as Cao *et al.* [45] who have attempted to enhance the mashup recommendation systems' diversity and accuracy. Their work presented the relational topic model (RTM) usage to mine the latent topics from APIs-mashups link relationships. It later utilizes them with the help of the factorization machine (FM) method to recommend top-k Web APIs for mashup development.

Moreover, Li *et al.* [46] have suggested integrating factors such as tag, topic, co-occurrence, and popularity in their model for mashup API recommendation. Their rational topic model (RTM) is used to output the mashups and APIs tags and topics for similarity calculation in-between Web APIs and mashups. API popularity is recognized in the model by information like historical invocation times and their category details. Information like mashups and API similarity, co-occurrence, and APIs popularity are fed to a Factorization Machine (FM) method to recommend top-k distinct and related APIs for a target mashup creation. Users' invocation records, similar to APIs and mashups, can be incorporated to enhance the APIs recommendation process. In [4, 47], they developed several APIs recommendation models based on a collaborative framework and a joint MF technique for information fusion to mine users' preferences over APIs and the in-between relationships.

Matrix factorization (MF) can have accurate results, but it is a shallow method that has low scalability. It can't learn the deeply hidden features of users and services. Its accuracy is limited by the richness of the input in which APIs description can be insufficient. Yao *et al.* [48] introduce a probabilistic matrix factorization (PMF) approach for mashup recommendation. It learns the influence of explicit similarities and implicit correlations of APIs on the co-invocations of APIs. The framework decomposes the API-mashup matrix into two low-dimensional matrices: API and mashup latent subsets. It unveils the latent APIs' implicit correlations from their co-invocation patterns using a designed latent variable model for MF regularization. It leverages the advantages of the MF technique and the APIs relations from historical mashup-API invocation records.

4.3.2. Content-based Filtering (CBF)

Content-based filtering recommendation model attempts to recommend similar items to the past interests of the users. As in our study, CBF does depend on the textual documents of the APIs and the descriptions of previously built mashups to build their future APIs pre-

diction model. There are various types of CBF models and will discuss some of them.

These types of CBF methods depend on matching the items, such as APIs, and extracted keywords with the input query for outputting the results. Keyword-based models play an important factor in the system's information filtering and facilitate item recommendation. These models focus mainly on similarity computing and comparison.

The context-aware WISeR framework proposed by Bianchini *et al.* [5] aimed at utilizing the multi-dimensional to model the APIs features such as categories, tags, functional features, and their usage history in mashups for keyword searching for the best set and ranking them afterwards. Their technique dynamically updates the dimensional attributes according to the developers' mashup selections.

Hao *et al.* [49] presented an automatic mashup query-based recommendation strategy that built upon the Targeted Reconstructing Service Descriptions (TRSD). TRSD is a technique that leverages the reconstruction of services description. First, it takes in the mashup descriptions and identifies the hidden information. Then, it calculates the similarity between previously developed mashup information and the mashup query input for leveraging services description information. Finally, the model will be able to recommend a ranked list of relevant services.

In an effort to utilize the deep networks for API mashups recommendation, Xiao *et al.* [50] have proposed a Deep Interest Network-based API Recommendation approach (DINRec) that captures the functional related semantic information and the composition relationship of mashups' APIs. It updates the mashup features vector while selecting relevant candidate APIs incrementally. Doc2sim mechanism is based on Doc2vec and cosine similarity that is used to extend the dataset for the deep network model training.

Zhou *et al.* [51] have attempted to improve existing APIs query-based recommendation methods by proposing a Boosting RecommendAtin with Implicit FeeDback (BRAID) that supports users' feedback. The model leverages learning-to-rank (LTR) techniques that provide a user's personalized ordered recommendation list of services based on his personal interaction history. Also, it integrates an active learning component to alleviate the "cold start" problem of feedback information. The study does not lay down a new recommendation but rather presents a boosting method for the existing approaches.

Semantic analysis approaches give the opportunity to explore the APIs and mashups of textual documents for similarity measurement. It can outperform traditional methods by adding the API semantic features knowledge to enhance the recommendation process, e.g. [52]. Their effectiveness relies heavily on the quality of the collected information [53].

For instance, the records of historical mashups can be studied along with the software developer's inputs to develop API recommendation techniques, such as [54], that identify the expected functional requirements and capture the potential composition of functionally satisfied and compatible APIs. Moreover, Gu *et al.* [55] developed a service packaging recommendation technique to recommend the best composition of services for mashup creation. It employs discourse analysis to study the mashup textual description and the semantic relevance between the services regarding their functionalities. Therefore, it incorporates the learned relationships and the mashup functional specifications input for a service set recommendation.

NLP techniques can be very useful and supportive when we deal with the APIs description or textual documents analysis in general for semantic-based recommendations. Lin *et al.* [56] developed an unsupervised framework Natural Language to service APIs (NL2API) to recommend services to users based on their input queries and services descriptions. NL2API is a bootstrap recommender that helps to capture in-between services relationships. It deals even with newly added services in a set of constructed communities of semantically similar services. Communities are built in a tree form in which nodes refer to their topics. It applies the Latent Semantic Index (LSI) technique for matching queries and existing communities and recommends top-k services.

Models have explored not only ranking the optimal recommended set but also the diversity of their items for developers to make a decision, i.e. Almarimi *et al.* [57] proposed a novel automatic approach SerFinder for service set recommendation. It runs the non-dominated sorting genetic algorithm (NSGA-II) as a services search technique. NSGA-II selects the optimal sets of APIs with regards to these objectives: their historical usage, functional matching to the expected mashup requirements, and functional diversity. The ranking component would use the output sets for recommending the common or most redundant services in these sets.

Some researchers have headed in the direction of improving the recommendation system in terms of best-capturing users' specifications, e.g. Jiang *et al.* [58] have implemented a novel semantic-based similarity approach, Service Discovery approach for Agile Mashup Development (SDAMD), that employed the user story concept of agile development to capture the user requirements. The method utilizes NLP to extract the three elements of the user's requirements and extract the three service attributes in an agile manner. The final step is to calculate the similarity between the service description and the search input and recommend the relevant services. Others tried to achieve a goal-driven mashup recommendation, such as [59], as they created a novel context-aware approach to attain the potential next service composition based on the under-construction mashups and selected services.

Zhang *et al.* [60] have implemented Recommendation

APIs by Stack Overflow posts and Java Packages (RASOP) that utilized word embedding techniques and question-and-answer information collected from Stack Overflow posts. LDA and LSA models are used to calculate the similarity of the extracted APIs topic word vectors and the previously assembled questions for a suitable API recommendation.

A similar study has studied the relationship between projects and their APIs, Thung *et al.* [61] developed a novel automated API recommendation framework called WebAPIRec. The model analyzes historical profiles of projects and their services into a personalized ranking model for project-specific web API ranking. It takes the project's textual description as input and utilizes learned information to find the most relevant web APIs. The approach ranks these APIs in descending order according to their relevancy and recommends the ranked list of services. A drawback is that this method does not recommend a composition of web APIs for the project when needed.

Probabilistic models are normally built based on the system's previously observed data with the goal of classifying its items into separate groups. These models employ knowledge acquired from previous events to filter the data and predict their classes. Therefore, they help greatly with multi-feature dataset classification, e.g., textual dataset.

For instance, Li *et al.* [62] attempted to assist mashup developers by presenting a probabilistic model to recommend the k-top of APIs. The model studies mashups-APIs' relationships via a relational topic model that helps obtain their latent relevant topics. Also, it incorporates the APIs' popularity to enhance the mashup recommendation process.

Various probabilistic models have endeavored to utilize the heterogeneous information network (HIN), a kind of directed graph that can model the dataset as in a network of nodes and edges with various defined types, as a helpful method to improve their recommendations. For example, Xie *et al.* [63] presented a mashup group preference-based service recommendation, GPSRec, on HIN. It employs the Bayesian personalized ranking algorithm to study the dense mashup interactions and therefore recommends personalized and ranked sets of services.

The importance of suggesting the best compatible mashup for applications during the design phase is to cut the development cost and save time. With that, some researchers have attempted to investigate the potential factors that influence the popularity of a mashup. Alshangiti *et al.* [64] have presented an approach that learns the influence of features such as functionality, novelty, efficient use of tags, selection of services, and the combination of selected services/tags in an early stage before the Bayesian model puts forth a confidence level for each of its predictions. They have drawn their focus on data provided by repositories such as ProgrammableWeb for analyzing popular mashups. The model aims at examining services from scratch at the designing level to save the time of developing shunned ones.

4.3.3. Hybrid Approaches

The merge of two distinct recommendation techniques, such as content-based and collaborative filtering, can significantly help to gain efficiencies and better results; this process is called a hybrid approach. These hybrid models can aid the merged methods in overcoming their shortcomings while working together in the same architecture to solve the problem. Numerous researchers have endeavored to solve some collaborative filtering challenges using various approaches. For instance, Jiang *et al.* [6] have developed an improved hybrid technique that combines word embedding and node embedding along with historical records of API invocation to generate output.

Traditional CF methods like matrix factorization are unable to capture the very sparse complex interactions between mashups and services. Hybrid approaches that take collaborative filtering and deep learning advantages to improve the precision of recommender systems have emerged recently. In [7], they have established an ensemble-based model engaging Word2vec similarity and Matrix Factorization techniques to obtain and select features for recommendations. Xiong *et al.* [65] also proposed a DL Hybrid Service Recommendation approach (DHSR) that takes on CF and the textual content, i.e., descriptions and tags of mashups and services. Historical mashups-services invocation interactions and functionality information are integrated by the DL model to learn complex mashups-services relationship features.

An innovative web service classification technique was proposed by Ye *et al.* [8]. They have explored the short services' functional description problems such as data sparsity for a solution. Their technique is based on implementing both the Wide learning and the Bi-LSTM models on the Web services' description documents inputs to predict their category. A linear regression algorithm is used in the framework as the final step to perform the service classification process. Their work can serve as a good basis for other web service discovery and recommendation systems.

An enhanced web services classification framework that exploits both the Bi-LSTM to learn the services feature representations and the topical attention mechanism to obtain their topic vector was proposed by Cao *et al.* [66]. In order to perform the final step of service classification, their work used the softmax neural network layer. Their work addresses function-based clustering approaches' challenges and offers an upgraded classification technique in terms of accuracy.

In an attempt to improve their recommendation system prediction, Xie *et al.* [67] approached utilizing the heterogeneous information network (HIN) of

mashups and services and their interactions in the prediction model. They proposed a novel factorization machine-based model, called FMRec, on HIN. It obtains the semantic similarity of selected meta paths and therefore builds the mashup-service matrices for MF to use with other sources of information to create a feature vector. The FMRec model automatically employs the process of learning the latent features of both mashups and services to predict the ratings of users on services for recommending the expected set of services.

Xie *et al.*, in their work [68], have suggested infusing both the DL and the HIN by adopting the Generative Adversarial Network (GAN) approaches to be able to study the data distribution and handle services that may not have been invoked by developers before. It presented an innovative method based on GAN and HIN, called HINGAN. First, it constructs a heterogeneous information network (HIN) using the semantic information of the mashups, services, their rich attribute, and various other sources. HIN has the power to express the mashup developer's preferences for services. Then, the approach samples numerous meaningful meta-paths from HIN to build similarity matrices of mashup services. Developers' required mashup attribute details are used as a condition in the GAN's adversarial training process. Using the learned vector, the model would recommend a list of services and create the expected mashup. The model does not only utilize the mashup and service information but also their historical invocation records. Yet, as an advantage, it does not require a vast set of these records.

Some users might not be able to describe their mashup requirements correctly for the best service recommendations. Therefore, there was a need for a comprehensive hybrid Open APIs discovery approach with regard to the user's preferences. Few have initiated the utilization of user stories from agile software development settings for robust data acquisition. For example, Jiang *et al.* [9] proposed the Hybrid Open API Selection Approach for Mashup development (HyOASAM) to discover open APIs that comply with the developers' requests. The system captures the mashup requirements from the offered user stories and extracts the three corresponding features matrix. Then, an FM model is employed to calculate the similarities or the association scores of mashups and public APIs to recommend the Top-N lists for mashup creation.

4.3.4. Summary

Traditional recommendation systems, see Table 3, are widely used in various industries to enhance user experiences and generate revenue. These systems rely on a range of algorithms to make personalized recommendations based on user-items preferences and behaviour.

Popular approaches like collaborative filtering are effective in identifying patterns among users. They have been used often, along with other techniques in recommendation systems implementation [69]. Traditional CF may not be robust and practical to deal with real-world problems such as the severe data sparsity challenge. Since it is dependent on historical records of user-item interactions, it may be affected by a cold-start problem when the information is unavailable at the time. Besides, it might lead to improper recommendations due to the provided inaccurate or inadequate service descriptions. In addition, CF-based system computation can become very expensive with the enormous growth of information over the Internet. Therefore, researchers have been tempted to investigate other techniques to bring forward further improvements to the recommendation systems.

Other traditional techniques, such as content-based filtering, relies on item characteristics to personalize suggestions. CBF models can be inflexible and limited in handling API extracted features and face a challenge in capturing their inter-dependencies. Unlike the CF methods, when recommending a set of APIs for mashup creation to satisfy the user's query, CBF methods might recommend entirely similar to those recommended before with no regard to other details like the compatibility of APIs. As a result, they will cause a problem for the newly added and the already existing APIs that haven't been invoked yet and affect the quality of the results. CBF approaches are frequently used to boost the performance of other techniques, such as the CF methods. They have been applied successfully in various domains to offer personalized recommendations [70].

Hybrid recommendation systems combine the two approaches to provide even more accurate recommendations by leveraging both user behaviour and item features. They successfully come in various combinations and have been used a lot nowadays to alleviate the disadvantages of standard techniques such as the cold-start issue [71, 72]. Though the hybrid models have properly combined several advantages of multiple techniques and have proven to be better than the baseline methods, they can be complex and expensive to implement.

Despite their effectiveness in generating relevant recommendations for individuals, traditional APIs recommender systems face several limitations, such as cold-start problems when new APIs or mashups are introduced. To address these limitations and provide better recommendations for API users and developers alike, several new approaches have emerged in recent years.

4.4. Network Representation Learning

Information networks have surfaced in recent years due to the need to capture complex relationships, yet they can become difficult to be analyzed. Network representation learning has been suggested by many analysts as a new paradigm to reconstruct network

TABLE 3: Traditional Recommender Systems

Studies	Algorithm	Specifications
Yin *et al.* (2020)	Collaborative Filtering	Matrix Factorization, Cognitive Knowledge Mining, Hidden Relationships Evaluation
Xu *et al.* (2021)	Collaborative Filtering	Joint Matrix Factorization Model, Similarity Computation, Neural Networks
Rahman *et al.* (2017)	Collaborative Filtering	Matrix Factorization, Ranked Clusters of Web APIs, Pearson Correlation Coefficient (PCC)
Wang *et al.* (2021)	Collaborative Filtering	Div_PreAPI, Mashup-API interactions, Matrix factorization
Cao *et al.* (2016)	Collaborative Filtering	Factorization Machine (FM), Relational Topic Model (RTM), APIs-mashups latent topics
Li *et al.* (2017)	Collaborative Filtering	Factorization Machine (FM), Relational Topic Model (RTM), latent topics, tags, historical invocation
Xu *et al.* (2021)	Collaborative Filtering	Joint matrix factorization, Doc2vec, Similarity computation methods
Yao *et al.* (2018)	Collaborative Filtering	Probabilistic Matrix Factorization (PMF), APIs implicit correlations, latent variable model, mashup-API invocation records
Bianchini *et al.* (2017)	Content-based Filtering	Context-aware WISeR framework, APIs features modelling, keyword searching
Hao *et al.* (2017)	Content-based Filtering	Automatic mashup query-based strategy, Targeted Reconstructing Service Descriptions (TRSD), mashup descriptions
Xiao *et al.* (2019)	Content-based Filtering	Deep Interest Network based API Recommendation approach (DINRec), mashups-APIs functional information, Doc2sim mechanism
Zhou *et al.* (2020)	Content-based Filtering	Boosting RecommendAtin with Implicit feeDback (BRAID), learning-to-rank (LTR) techniques, active learning
Gu *et al.* (2016)	Content-based Filtering	Service packaging recommendation technique, discourse analysis, mashup functional features
Qi *et al.* (2021)	Content-based Filtering	WAR(text), Compatibility-aware, text description-driven recommendation
Gu *et al.* (2016)	Content-based Filtering	Mashup textual description mining, Semantic relations analysis
Lin *et al.* (2018)	Content-based Filtering	Unsupervised framework Natural Language to service APIs (NL2API), bootstrap recommender, Latent Semantic Index (LSI) technique
Almarimi *et al.* (2019)	Content-based Filtering	SerFinder, non-dominated sorting genetic algorithm (NSGA-II), APIs optimal sets
Jiang *et al.* (2019)	Content-based Filtering	Service Discovery approach for Agile Mashup Development (SDAMD), semantic-based similarity approach, NLP techniques
Xie *et al.* (2022)	Content-based Filtering	Goal-driven and context-aware machine learning method, Representation learning
Zhang *et al.* (2019)	Content-based Filtering	Recommendation APIs by Stack Overflow posts and Java Packages (RASOP), LDA and LSA techniques, APIs topic vectors
Thung *et al.* (2017)	Content-based Filtering	WebAPIRec, ranked project-specific web APIs, projects textual description
Li *et al.* (2014)	Content-based Filtering	Relational topic model, mashups-APIs' latent relevant topics
Xie *et al.* (2019)	Content-based Filtering	Mashup group preference-based service recommendation (GPSRec), bayesian personalized ranking algorithm

Studies	Algorithm	Specifications
Alshangiti *et al.* (2020)	Content-based Filtering	Bayesian model, popular mashups data features analysis
Jiang *et al.* (2022)	Hybrid Approaches	WMD-NV, Historical invocation information, Word embedding, Node embedding
Chen *et al.* (2022)	Hybrid Approaches	Ensemble-based approach, Word2vec similarity, Matrix Factorization techniques, GBDT, GRU
Xiong *et al.* (2018)	Hybrid Approaches	Deep Learning Hybrid Service Recommendation (DHSR), CF, mashups-services historical invocation
Ye *et al.* (2019)	Hybrid Approaches	the Wide learning and the Bi-LSTM models, Web services description, Linear regression algorithm
Cao *et al.* (2019)	Hybrid Approaches	Bi-LSTM model, topical attention mechanism, service classification, function-based clustering approaches
Xie *et al.* (2018)	Hybrid Approaches	Heterogeneous information network (HIN), Factorization machine-based model (FMRec), mashup-service latent features
Xie *et al.* (2019)	Hybrid Approaches	HIN, Generative Adversarial Network (GAN), mashups and services semantic information & matrices
Jiang *et al.* (2020)	Hybrid Approaches	Hybrid Open API Selection Approach for Mashup development (HyOASAM), software agile development, FM model

information into a low-dimensional vector with regards to not losing any vital details [73].

The task of manually discovering the qualified and compatible cluster of web APIs from massive candidates is time-consuming, inefficient, and stressful. Qi *et al.* [74] present a weighted APIs correlation graph (W-ACG) to help learn the functions and compatibilities of APIs. Later on, they use the learned details to feed the Keywords-based and Compatibility-aware APIs Recommendation (K-CAR) model for recommending the function-qualified and compatibility-guaranteed set of APIs. K-CAR as a dynamic recommendation model can aid users in searching for APIs to satisfy their complex requirements with no need for prior knowledge of APIs structure. Then, the novel W-ACG model constructs a graph of APIs with their descriptive words as the nodes and the weight values as the edges. K-CAR heavily depends on historical API integration data to learn their compatibility, yet in a practical setup, the majority of them haven't been invoked by developers for building apps. Besides, this model does not scale well with the web APIs for the recommendation.

Qi *et al.* [75] proposed a data-driven approach called Web APIs Recommendation (WAR) that discovers, verifies, and selects web APIs based on users' search keywords. It helps programmers select the appropriate APIs without detailed knowledge of web APIs. The framework first builds a web API correlation graph of their compatibility from their vast existing data. Then, WAR aids with recommending a compatible and diverse set of APIs based on the typed keywords. It returns a sub-graph of APIs that fulfil the search.

Knowledge graphs have been widely discussed as they can bring forward fruitful facts that can enhance recommendation methods. Wang *et al.* [76] in their research have proposed an unsupervised method based on the mashup-API knowledge graph embedding and the dynamically deep random walk to adjust the results for developers' preferences.

Modeling API recommendation as a graph search problem has gained increased attention recently as several studies have focused on API compatibility and how to achieve it by treating it as a Steiner Tree search problem that needs to be addressed. A series of approaches, e.g., as that presented in [77, 78, 79, 80], have employed the usage APIs correlation graph-based approach and the Steiner Tree search problem to recommend a personalized set of functional-satisfactory APIs for APP developers. In [81], they have further enhanced the Steiner Tree search approach-based recommendation by incorporating API popularity into the equation to find the optimal set of APIs for mashups to ensure the diversity and quality of results.

4.4.1. Summary

Influenced by the deep learning ability to learn data features, network representation learning frameworks, Table 4, came to interest as they can obtain representation vectors from the data network. Therefore, they can efficiently assist significant tasks such as the network's nodes classification, clustering, and links prediction and display an overview of the network data and pattern recognition. Hence, data sparsity and limited scalability may impose a challenge for traditional network analysis [82].

TABLE 4: Network Representation Learning Studies

Studies	Technique	Specifications
Qi *et al.* (2019)	Weighted APIs correlation graph	Keywords based and Compatibility-aware APIs Recommendation (KCAR), historical APIs invocation
Qi *et al.* (2020)	API correlation graph	Web APIs Recommendation (WAR), data-driven approach
Wang *et al.* (2021)	Graph Embedding Technique, Deep Random Walk	Mashup-API co-invocation patterns, Knowledge Graph, entity bias procedure
Qi *et al.* (2021)	Steiner Tree search problem, Correlation graph-based approach (PC-WAR)	Web APIs correlation graph, Personalized and compatible Web APIs recommendation problem
Gong *et al.* (2022)	Steiner Tree search problem, Diversity-aware Web APIs recommendation approach	Keyword search technique, Determinantal point process (DPP)
Gong *et al.* (2020)	Steiner Tree search problem, Keywords-driven web APIs recommendation approach	Weighted web APIs-specific correlation graph, Keywords-driven and compatibility-aware multiple API group recommendation
Gong *et al.* (2021)	Steiner Tree search problem, DivCAR	Diversified and Compatible web APIs Recommendation approach, Random walk technique
Wu *et al.* (2022)	Steiner Tree search problem, PD-WACR	Popularity-aware and diverse method of web API compositions' recommendation, APIs' correlation graph

4.5. Deep Learning Techniques

Many researchers have attempted to tackle the CF common challenges, such as sparsity and its limited predictability. Several have investigated the usage of neural network (NN) and deep neural network (DNN) techniques Fig. 10 since they can automatically learn the nonlinear mashup-service interactions and the ability to represent the extracted features efficiently. Therefore, deep learning models were introduced in recommendation systems to solve some of the existing problems. They have been infused with many other recommendation techniques to make an effort to improve them and to handle the emerging high number of developed APIs.

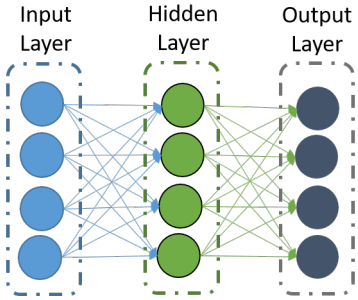

FIGURE 10. A Simple Deep Learning model

For instance, Zhang *et al.* [10] have expressed while creating mashups not only to consider the relevant APIs but also the ones that have high-order composition interactions in between. It employs the Doc2Vec mechanism for extracting the functionality features from the raw APIs descriptions as vectors. Then, it proposed to cluster the APIs with similar functions based on their feature vectors to match the target mashup description. It further utilized the Deep-FM, which takes advantage of using both techniques, the deep neural network factor and factorization machine, to learn these clusters' interaction information for a better API recommendation.

For recommender systems, mashup establishment may require adequate domain knowledge, yet inexperienced developers face the challenge of identifying sufficient knowledge to aid the services recommendation process. Chen *et al.* [84] proposed a keyword-based deep reinforcement learning model (K-DRSTS) that uses a service-keyword correlation graph to learn their relationships and the Steiner tree search algorithm for service discovery. Huang *et al.* [85] proposed Bi-Information source-based KnowledgE Recommendation BIKER, an API recommendation approach that tackles issues such as lexical and knowledge gaps. Their method employs word embedding techniques and incorporates both the Stack Overflow posts and API documentation to recommend a ranked top-k APIs list that matches the developer's program task.

In an effort to capture the developers' updated preferences and requirements, Ma *et al.* [11] implemented a deep learning model named DLISR. Their multi-round recommendation approach aims to learn the mashup interactions via historical records and recommend the appropriate composition of APIs. Zhang *et al.* [86] have implemented an offline and online framework where the offline deep neural network module learns the mashup-APIs relationship. The online component recommends

the APIs based on the requested mashup description.

With the successful prevalence of NN frameworks to effectively learn useful word representations, several scientists have tried to employ a wide range of NN techniques in their research. Though factorization machine models have shown an improved accuracy than traditional CF models, they still may lack capturing complex or different weight interactions. The integration of a deep neural network framework and attention mechanism with the factorization machine model, named the Neural and Attentional Factorization Machine for Web API Recommendation (NAFM) [87] was implemented to capture the non-linear interactions and the significance of their features.

Similarly, Cao *et al.* [88] have proposed the usage of the Attentional Factorization Machine recommendation method (AFMRec) for service recommendation. It employs the traditional matrix factorization of decomposing both the multi-dimensional features and their implicit cross-interactions. The model, as a start, tends to use the Doc2Vec technique to extract significant semantic latent features from the raw dataset and transform them into the model input format. Next, the model would utilize multi-dimensional information to predict the mashup-service ratings for service recommendations. Deep Factorization Machine models have displayed higher performance than traditional collaborative filtering ones. Yet, as these models ignore significant features and assign the same weights, an attention mechanism is employed to capture the varying importance of interactions, such as the NAFM model in [89]. Cao *et al.* [90] attempted to predict and rank web APIs for building their relationship network. They used the graph attention network (GAT) and the DeepFM models to classify APIs into clusters based on their functional features.

Nguyen *et al.* [12] have proposed a probabilistic matrix factorization (PMF) model called an attentional PMF model. It addresses one of the shortcomings of traditional PMF as it emphasizes studying the importance of assigning different weights or scores to the extracted latent feature interactions from the historical invocation since not all of them may provide the same significantly useful information. The attention mechanism in NN is employed in this model first to parametrize the attention scores and learn them before weighing the latent features of mashup-APIs invocations. In addition, the utilization of the implicit relationships between the APIs, along with their history of co-invocations, enhances the prediction model. Doc2Vec technique is used by the model to learn the document embeddings and generate the corresponding matrix. The contextual similarity and the APIs co-invocation are added to the model for regularization. It helps to reduce the PMF model overfitting and enhance the recommendation further. Moreover, Nguyen, in their paper [91], employed the Attentional PMF (AMF) to set weights and determine the significance of latent features of mashups and APIs for outperforming the traditional PMF.

The integration of content-based recommendation with deep learning was presented by Shi *et al.* [92] as a solution to the service description's data sparsity challenge. They proposed a novel probabilistic model to expand service descriptions at a sentence level to relieve the feature sparsity gap problem. Additionally, it utilized a Long Short-Term Memory (LSTM) model-based deep architecture with two attention mechanisms for recommending top-N ranks. A functional attention mechanism that focuses on the most decisive features, which may affect the recommendation accuracy, and a contextual attention mechanism to select services to match the user's specific requirements. Previous approaches fell short of weighing the significant features found in the descriptions, which this paper has taken into consideration.

Shi *et al.* [93] have also presented a model based on Bi-LSTM networks to obtain the services' functional features and the functional requirements of mashups. Their model also consists of an attention mechanism that utilizes service tags to assign attention scores to the different service description words.

The development of new mashups has caused the cold-start challenge. Since deep neural representation learning is powerful with hidden features extraction, Ma *et al.* [94] have implemented a multiplex interaction-oriented service recommendation mechanism (MISR). The model studies three types of interactions between mashups and services, which are content interaction, implicit neighbor interaction, and explicit neighbor interaction. These interactions are incorporated into a DNN model to predict the probability of candidate services invocation in a new mashup. A novel mashup-oriented recommendation technique was presented by Wu *et al.* [95] named neural framework (MTFM). Their method aims to model the feature interactions between mashups and APIs and generate API recommendations matching mashup requirements. They further utilized other auxiliary information, such as metadata, to develop a metric for compatibility evaluation.

Early-on keyword-based API discovery approaches have long suffered the term-mismatch problem due to the differences in user queries and service descriptions. With that, few have attempted to incorporate alternative feature learning methods that can achieve better results. For instance, Lizarralde *et al.* [96] introduced the use of Variational Autoencoders neural networks as a way of restricting the encoded representation of learned latent features. Autoencoders, as a type of NN, have had a significant contribution by reducing input dimensions and therefore helping with learning data hidden features and modeling complex data relationships.

The application of Stacked AutoEncoders (SAE), a deep feedforward network model, for predicting the future mashups services compositions and substitutions

interactions was proposed by Labbaci *et al.* [97]. The goal of using SAE is to learn the latent service characteristics such as text description, categories, constraints, and others and use them with previous interactions to predict new ones. It helps to alleviate the cold-start problem that traditional recommendation approaches suffer from, as well as can be relied on for identifying compatible services for potential mashup development. Interactions are constructed in a service network graph with services as the nodes and the composition and substitution relationships.

Moreover, Bai *et al.* [98] have presented a deep learning framework for long-tail services recommendations (DLTSR). The approach adopts Stacked Denoising AutoEncoders (SDAE) as the main component for effective feature extraction. The procedure does utilize the hot services invocation records and the modeled developer's preferences for SDAE outputs regularization.

Studies like [99, 100] Knowledge Graphs have been used as auxiliary information to alleviate user-service interactions and matrix sparsity. Dang *et al.* [99] presented a deep knowledge-aware method that utilizes textual contents and the knowledge graph to capture the complex relations between mashups and web services in a setting of a sparse invocation matrix. A graph attention network model was developed by Li *et al.* [100] (WSR-MGAT) to fully exploit the knowledge graph meta-path information to improve the accuracy and interpretability of recommendations. Yu *et al.* [101] have also attempted to handle the data sparsity by implementing an API Knowledge Graph and Intent Network based Mashup-Oriented API recommendation method (AKGIN) that models the mashup intention through the mashup-API high-order interactions.

The deep learning-based open API recommendation (DLOAR) approach proposed by Wang *et al.* [102] aimed to help developers find the most relevant and high-quality open APIs for their composition application development. It uses a combination of hierarchical density-based spatial clustering, TextRank algorithm, and neural network-based similarity calculation to recommend the best APIs. It outperforms the baseline approaches in terms of precision, recall, and F1-measure.

WANG *et al.* [103] presented a novel mashup creation method, named the Functional and Structural Fusion Model (FSFM) to address the challenges faced by current Web API recommendation methods such as relying solely on API information. The methodology involves a structural component for capturing the latent interactions between mashups and APIs in a heterogeneous network and a functional semantic component to help generate the embedding vectors of mashups and APIs to fuse them later and recommend a list of candidate APIs for mashup developers. The model is evaluated against different baselines, including traditional Collaborative Filtering (CF) methods and Neural Collaborative Filtering (NCF). The results show that the FSFM method outperforms other methods. However, it is worth noting that the proposed model may have limitations in terms of scalability and computational complexity, as it involves multiple components and the integration of different types of features.

He *et al.* [104] proposed a probabilistic generative model called Binary-API Topic (BAT) to help developers select appropriate APIs for their mashup projects. The recommendation model is built upon mashup data and API co-occurrence patterns to learn sparse interactions between mashups and APIs. Its objective is to extract binary APIs and predict unknown pairwise interactions, improving user satisfaction and promoting more popular web applications. The results show that BAT yields a higher-quality recommendation list. Though the model attempts to alleviate the severe sparsity issue of individual mashups, the global mashup repository is required to learn the topic distribution across all mashups.

Yu *et al.* [105] focused on handling the cold start problem in the mashup creation setting by proposing a novel approach named SRMG for web service recommendation for mashup. They employ the Bidirectional Encoder Representations from Transformers (BERT) to discover any functionally similar mashups and Graph Generative Adversarial Networks (GraphGAN) to obtain the mashups and services representation vectors based on the historical invocation records. Their model learns the mashup preferences for each service to derive the list of service candidates that match the new mashup preference.

The objective of works such as Xiao *et al.* [106] is to propose a new method for web API recommendation that effectively reflects the relationship between mashups and APIs. The proposed method, called the Hypergraph Convolutional Network for Web API Recommendation (MRHN), uses motifs to extract a hypergraph structure and a hypergraph convolutional network to obtain feature embeddings of mashups. The method also uses a channel error attention mechanism to adjust the weight of each channel and graph convolution to obtain a representation vector that contains mashup-API calling information. The results of the experimental analysis and evaluation show that the MRHN method outperforms existing methods in terms of accuracy. Wang *et al.* [107] have also employed a motif-based attention mechanism in their Motif-based Graph Attention Network for service recommendation (MGSR) approach to capture high-order interactions in the Mashup-API interaction bipartite graph along with a collaborative filtering model to derive the recommendations. Moreover, Zheng *et al.* [108] have developed a Hierarchical Motif-based Graph attention network for Service Recommendation (H-MGSR) that applies node-level and motif-level attention mechanisms to help capture the significance of different motifs and their impact on the accuracy of service recommendations.

TABLE 5: Deep Learning Techniques

Studies	Technique	Specifications
Zhang *et al.* (2018)	Deep-FM	Doc2Vec mechanism, APIs functionality-based clustering
Chen *et al.* (2022)	K-DRSTS, Steiner tree search problem	Keyword-based Deep Reinforced Steiner Tree Search, Service-keyword correlation graph
Huang *et al.* (2018)	Deep Learning Technique	Bi-Information source-based KnowledgE Recommendation (BIKER), word embedding techniques, Stack Overflow posts and API documentation
Ma *et al.* (2022)	DLISR, Hybrid model	Service bundle recommendation framework based on deep learning, HISR
Zhang *et al.* (2021)	Deep recommendation framework	Feature extraction, Matrix decomposition, Neural network learning mapping
Kang *et al.* (2020)	Neural and Attentional Factorization Machine (NAFM)	Classical FM, Multi-dimensional Features Incorporation
Cao *et al.* (2019)	Attentional Factorization Machine	Doc2Vec technique, services semantic latent features
Kang *et al.* (2021)	NAFM, Novel neural network architecture	Hybrid factorization machine model, Attention mechanism
Cao *et al.* (2022)	GAT, DeepFM	Graph attention representation, Web API relationship network
Nguyen *et al.* (2020)	Attentional probabilistic matrix factorization (PMF)	Doc2Vec technique, APIs co-invocation data
Nguyen *et al.* (2021)	Attentional PMF Model (AMF)	Matrix factorization, Neural attentional network, Doc2Vec
Shi *et al.* (2018)	probabilistic model with LSTM and attention	Novel probabilistic model, Two attention mechanisms
Shi *et al.* (2019)	Bi-LSTM networks with attention	Bi-LSTM networks, attention mechanism, services tags
Ma *et al.* (2020)	Deep Learning Techniques	Multiplex interaction-oriented service recommendation mechanism (MISR), mashups-services interactions
Wu *et al.* (2021)	MTFM, MTFM++	multi-model fusion and multi-task learning for Mashup-oriented Web API recommendation
Lizarralde *et al.* (2020)	Variational Autoencoders	Variational Autoencoders neural networks, complex data relationships modelling
Labbaci *et al.* (2017)	Stacked AutoEncoders (SAE)	latent service characteristics, service network graph
Bai *et al.* (2017)	Stacked Denoising AutoEncoders (SDAE)	Deep learning framework for long-tail services recommendations (DLTSR)
Dang *et al.* (2021)	Deep knowledge-aware approach	Knowledge Graph, Knowledge Representation, Mashup-Service relationships learning
Li *et al.* (2022)	WSR-MGAT	Metapath-guided Graph Attention Network Model, Knowledge Graph, Novel distance-aware-based path sampling strategy
Yu *et al.* (2023)	AKGIN	Knowledge Graph, Intent Network, LDA
Wang *et al.* (2023)	DLOAR	Hierarchical Density-based Spatial Clustering, TextRank algorithm, and Neural Network-based similarity calculation
Wang *et al.* (2023)	FSFM	Structural Component, Functional Semantic Component, Embeddings Fusion
He *et al.* (2023)	BAT	Binary API Topic Extraction, API co-occurance patterns learning
Yu *et al.* (2023)	SRMG	Bidirectional Encoder Representations from Transformers, Graph Generative Adversarial Networks, Mashup-Service preferences

Studies	Technique	Specifications
Xiao *et al.* (2023)	MRHN	Motifs, Hypergraph Convolutional Network, Attention mechanism
Wang *et al.* (2023)	MGSR	Motif-based Attention Mechanism, Mashup-API Interaction Bipartite Graph, Collaborative Filtering Model
Zheng *et al.* (2023)	H-MGSR	Node-level Attention Mechanism, Motif-level Attention Mechanism

4.5.1. Summary

Deep Learning-based recommender models have achieved significant advances in recent years in their flexibility, effectiveness, and high-quality output. Their techniques in Table 5 can assist with learning users' preferences, item characteristics, and mashup-item interactions. They have garnered considerable interest, and their successful widespread has influenced their adoption in many real-world problems, i.e., API mashup recommendations. However, deep learning models are still suffering limitations and drawbacks that can affect their productivity and accuracy, such as interpretability and the rich dataset requirement. Yet, this field is still flourishing and has shown signs of improvement.

5. FUTURE DIRECTIONS

In previous sections, we discussed how recommendation techniques could be used to create API mashups. As recommendation techniques can have limitations or be enhanced, there are several interesting areas of research in recommendation systems that can be considered to be employed in API recommendation to facilitate mashup creation. This section discusses a few of these open research areas and active topics in recommendation systems. We will tackle a couple of innovative recommendation systems, such as on-the-fly recommendation and explainable systems, that have attracted researchers to the community.

5.1. Bootstrapping Cold-start Items

New recommender systems may fail to provide sufficient and helpful recommendations because of their limited datasets. Systematic bootstrapping cold-start items is a technique used in recommender systems to handle the problem of recommending items with little or no historical interaction data available, also known as "cold-start" items. The essence of bootstrapping recommenders is to enrich the inadequate history of interactions and hence accelerate these models.

A data-centric bootstrapping approach was developed by [110] to support the launched system with new user interactions and improve its model performance. Their machine learning framework involves building an initial model based on metadata or auxiliary information about the item, such as genre, director, actors, year released, etc., rather than relying solely on user-item interaction data. For instance, some research like [109, 111] have pushed forward the implementation of information elicitation methods based on conducting user interviews for assembling the new user preferences data. These user interviews are gathered following carefully selected seed-set items with various features.

Many deep reinforcement learning models were utilized by many system designers such as Lee *et al.* [112], where they applied two neural networks to learn from each other and to identify the potential positive user-item interactions. They aimed to overcome some of the drawbacks of one-class collaborative filtering (OCCF) and increase the robustness of bootstrapping techniques. As the initial stage of recommender systems may be affected by the lack of information required for generating high-quality recommendations, bootstrapping recommender models were established to fill the gaps in the dataset. They have gained the interest of system developers to deal with the cold-start problem many recommenders encounter.

Bootstrapping Cold-start Items is a useful technique for addressing cold-start issues in recommenders when generating recommendations in recommender systems by leveraging additional information beyond the user-item interactions. It has become an essential part of recommendation systems' strategies as it significantly improves their accuracy and helps provide relevant and timely recommendations. In the article, there were several studies have incorporated bootstrapping techniques to enrich their dataset, yet it is still a new area of research that can be investigated.

5.2. Leveraging External Side-information

Some researchers have done pioneer work considering the rich side information like social network data (e.g., social comments or tags on APIs) to support API or service recommendations. Social network-based approaches aim to personalize the recommendation results; however, processing large-scale data can be time-consuming. Network representation learning approaches have recently surfaced, and researchers have been interested as they can be beneficial. Wu *et al.* [113], for example, have discussed a hybrid network representation learning technique for service recommendation based on integrating the users' co-tag and social networks. Their approach brings further improvement as it can help learn users' preferences and handle big information networks.

Clustering-based recommendation techniques have emerged to be commonly used for service and API recommendations. Yet, they could be evolved and enhanced to include user-service traditional relations and other data sources. Information like temporal, spatial, and others can affect users' preferences; therefore, it can be used as auxiliary information in recommendation systems. Mezni *et al.* [114] have proposed a context-aware service recommendation technique that focuses on the temporal information and the K-means clustering method to suggest the top-rated list of services.

Wang *et al.* [115] developed a novel Diversified Recommendation method based on LSH (DivRec_LSH) as an attempt to enhance the traditional CF-based recommendation solution diversity as well as its accuracy for users' satisfaction. The architecture makes use of historical records and the efficient search technique Locality-Sensitive Hashing (LSH). The model emphasizes that the recommended items for users need not be redundant. It experimented on a movie data set, Movie-Lens, for efficiency results. (related but not quietly for web services)

The implicit invocation feedback data of APIs can enhance the recommendation system's accuracy. Contextual information, such as geographical location, can significantly impact API invocations. With the help of side information, API recommendation systems can be enhanced and their output can be refined even further. For example, in Botangen *et al.* [116] work they have suggested an approach that calculates preference scores from geographical location information and API functionality descriptions. Afterwards, they incorporate them into their matrix factorization recommendation-based approach for generating favored mashup-API recommendations.

5.3. On-the-fly Recommendation

The rapid changes in users' interests or data patterns can pose a critical challenge for recommendation models. Similarly, for multiple domains, the relevance and the order of information may drop over time, which adds another challenge. For instance, articles in the news or fashion domains are continuously updated based on shifting trends.

Since the requirements of businesses and customers can be modified at any time, recommendation systems should also be flexible and configurable at runtime. In [117], a sequential pattern mining approach was developed to generate updated news recommendations. In their approach, the ongoing user session information is analyzed for pattern extraction and compared with the user clickstream's real-time inputs for the recommendation. Wang *et al.* [118] has introduced a new family of online multi-task collaborative filtering (OMTCF) to improve the issues of existing online collaborative filtering algorithms. It applies the principles of online multi-task learning and collaborative filtering techniques to learn the model accurately and update the relevant data based on the matrix of user interaction.

In a different setting, Safran *et al.* [119] have implemented two novel algorithms to help generate on-the-fly recommendations in crowdsourcing systems. They have investigated the characteristics and relationships of task/worker and utilized the "categories" key factor to accelerate the recommendation model. As online recommender systems, e.g. API recommendation systems for mashups, efficiency and scalability can be enhanced, the development research of the on-the-fly recommender systems is still ongoing.

5.4. Explainable Recommendation

The current personalized recommendation techniques may provide good predictions but cannot give proper justifications or feedback which can help users understand the results and guarantee their satisfaction. Therefore, a new trend of recommendation models, called Explainable recommendation models, has arisen to generate high-quality recommendations and help users understand the algorithms' insight, Fig. 11. Unlike others, these types of models bring many benefits as they improve the effectiveness, trustworthiness, and user stratification of recommendation systems. Explainable recommendation systems can be either model-intrinsic, interpretable models whose mechanisms provide the results and the explanations together to the users, or model-agnostic, or *post hoc* models, a BlackBox-based recommendation mechanism that generates explanations based on the recorded decisions [120].

The aim of the explainable recommendation models is to involve humans in the process of developing useful and agreeable recommendations. It is a very interesting and promising research area for API recommendation systems researchers because it can help them analyze users' behavior and preferences, and give an explanation for mashup developers to help make a decision of whether the recommended mashup is useful or not.

5.5. User Tools

The vast amount of available application tools and features led to the need for recommender systems to help facilitate the software development process. A few researchers endeavoured to implement recommenders that act in a user's role and recommend tools for another user in the system. Brown *et al.* [121] have conducted a study to explore the nature of user-to-user recommendations via data analysis and observing real-world cases of users recommending tools to others. Again, the inadequate amount of previous users interactions data at the initiation phase of the system can bring the traditional issue of cold-start,

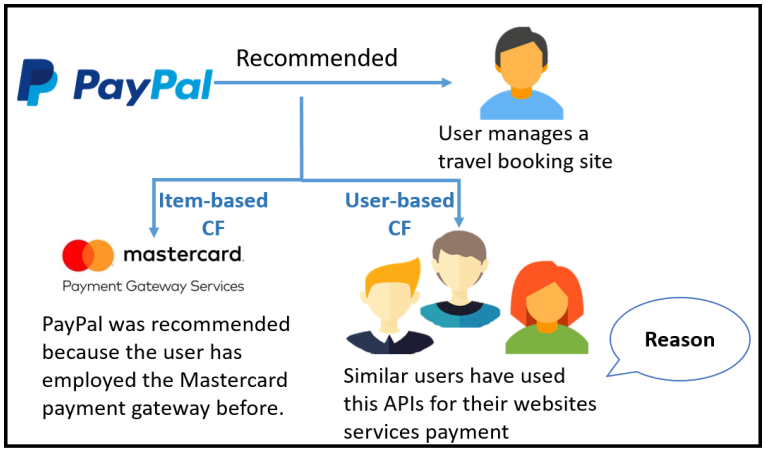

FIGURE 11. Explainable Recommendation Model Examples

which it needs to handle. It is challenging to picture user interests fully as users behave diversely under different circumstances. In addition, the automation of tool recommendations may also have to meet a defined set of metrics based on the characteristics of the project. Research is still further examining the effectiveness of relevant factors such as user experience, length, task difficulty, and others.

6. CONCLUSIONS

Web API recommendation for mashup creation has become a trending research topic for its benefits in the service computing field. In this survey, we briefly inspected the characteristics of recommender systems and discussed the significant emergence of Web APIs recommendation systems and their advancement. Moreover, we investigated the current state-of-the-art solutions and presented a review of their techniques for mashup creation. We discussed the benefits and limitations of popular traditional recommendation methods, such as collaborative-based filtering and content-based filtering techniques. Besides, this work examined other models like network representation learning and hybrid approaches. As deep learning techniques attracted much interest, we introduced various developed systems to identify their key points and challenges. Finally, we suggested a couple of potential future research directions for further analysis. Nowadays, though many studies have attempted to offer reliable approaches for accurate mashup development, it is still believed to have room for improvement.

DATA AVAILABILITY STATEMENT

The data underlying this article will be available from the corresponding author, [HA], upon reasonable request.

REFERENCES

[1] Tan, W., Fan, Y., Ghoneim, A., Hossain, M. A., & Dustdar, S. (2016). From the service-oriented architecture to the web api economy. *IEEE Internet Computing*, *20*(4), 64–68.

[2] Lamothe, M., & Shang, W. (2020). When apis are intentionally bypassed: An exploratory study of api workarounds. *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE), Seoul, South Korea, 27 June-19 July*, pp. 912–924.

[3] Yin, Y., Huang, Q., Gao, H., & Xu, Y. (2020). Personalized apis recommendation with cognitive knowledge mining for industrial systems. *IEEE Transactions on Industrial Informatics*, *17*(9), 6153–6161.

[4] Xu, Y., Wu, Y., Gao, H., Song, S., Yin, Y., & Xiao, X. (2021). Collaborative apis recommendation for artificial intelligence of things with information fusion. *Future Generation Computer Systems*, *125*, 471–479.

[5] Bianchini, D., Antonellis, V. D., & Melchiori, M. (2017). Wiser: A multi-dimensional framework for searching and ranking web apis. *ACM Transactions on the Web (TWEB)*, *11*(3), 1–32.

[6] Jiang, B., Li, H., Yang, J., Qin, Y., Wang, L., & Pan, W. (2022). Web service recommendation based on word embedding and node embedding. *Mobile Information Systems*, *2022*.

[7] Chen, J., Wang, Y., Huang, Q., Jiang, B., & Liu, P. (2022). Open apis recommendation with an ensemble-based multi-feature model. *Expert Systems with Applications*, *196*, 116574.

[8] Ye, H., Cao, B., Peng, Z., Chen, T., Wen, Y., & Liu, J. (2019). Web services classification based on wide & bi-lstm model. *IEEE Access*, *7*, 43697–43706.

[9] Jiang, B., Liu, P., Wang, Y., & Chen, Y. (2020). Hyoasam: A hybrid open api

selection approach for mashup development. *Mathematical Problems in Engineering*, *2020*.

[10] Zhang, X., Liu, J., Cao, B., Xiao, Q., & Wen, Y. (2018). Web service recommendation via combining doc2vec-based functionality clustering and deepfm-based score prediction. *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom), Melbourne, VIC, Australia, 11-13 December*, pp. 509–516.

[11] Ma, Y., Geng, X., Wang, J., He, K., & Athanasopoulos, D. (2022). Deep learning framework for multi-round service bundle recommendation in iterative mashup development. *CAAI Transactions on Intelligence Technology*.

[12] Nguyen, M., Yu, J., Bai, Q., Yongchareon, S., & Han, Y. (2020). Attentional matrix factorization with document-context awareness and implicit api relationship for service recommendation. *Proceedings of the Australasian Computer Science Week Multiconference, Melbourne, VIC, Australia, 4-6 February*, pp. 1–10.

[13] Xue, Q., Liu, L., Chen, W., & Chuah, M. C. (2017). Automatic generation and recommendation for api mashups. *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 18-21 December*, pp. 119–124.

[14] Xu, G., Lian, S., & Tang, M. (2023). Web api service recommendation for mashup creation. *International Journal of Computational Science and Engineering*, *26*(1), 45–53.

[15] Halili, F., Ramadani, E., et al. (2018). Web services: A comparison of soap and rest services. *Modern Applied Science*, *12*(3), 175.

[16] Altexsoft. (2022). What is api: Definition, types, specifications, documentation [Available at https : / / www . altexsoft . com / blog / engineering / what - is - api - definition - types - specifications - documentation/, Accessed on 2022/11/05].

[17] DevExchange, C. O. (2017). Infographic - a history of modern api mashups [Available at https : / / medium . com / capital - one - tech / infographic- a- history- of- modern- api- mashups- 9476dad7685b].

[18] Zhang, Y., Chen, R., Tang, J., Stewart, W. F., & Sun, J. (2017). Leap: Learning to prescribe effective and safe treatment combinations for multimorbidity. *proceedings of the 23rd ACM SIGKDD international conference on knowledge Discovery and data Mining, Halifax, NS, Canada, 13-17 August*, pp. 1315–1324.

[19] Wang, L., Zhang, W., He, X., & Zha, H. (2018). Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, London, UK, 19-23 August*, pp. 2447–2456.

[20] Wang, S., Ren, P., Chen, Z., Ren, Z., Ma, J., & de Rijke, M. (2019). Order-free medicine combination prediction with graph convolutional reinforcement learning. *Proceedings of the 28th ACM international conference on information and knowledge management, Beijing, China, 3-7 November*, pp. 1623–1632.

[21] Wang, M., Liu, M., Liu, J., Wang, S., Long, G., & Qian, B. (2017). Safe medicine recommendation via medical knowledge graph embedding. *ArXiv*, *abs/1710.05980*. https://api.semanticscholar.org/CorpusID:3746317

[22] Symeonidis, P., Chairistanidis, S., & Zanker, M. (2022). Safe, effective and explainable drug recommendation based on medical data integration. *User Modeling and User-Adapted Interaction*, *32*(5), 999–1018.

[23] Symeonidis, P., Kostoulas, T., Danilatou, V., Andras, C., & Chairistanidis, S. (2022). Mortality prediction and safe drug recommendation for critically-ill patients. *2022 IEEE 22nd International Conference on Bioinformatics and Bioengineering (BIBE), Taichung, Taiwan, 7-9 November*, pp. 79–84.

[24] Symeonidis, P., Manitaras, G., & Zanker, M. (2023). Accurate and safe drug recommendations based on singular value decomposition. *2023 IEEE 36th International Symposium on Computer-Based Medical Systems (CBMS), L'Aquila, Italy, 22-24 June*, pp. 163–168.

[25] Symeonidis, P., Bellinazzi, L., Berbague, C., & Zanker, M. (2023). Safe and effective recommendation of drug combinations based on matrix co-factorization. *2023 IEEE 36th International Symposium on Computer-Based Medical Systems (CBMS), L'Aquila, Italy, 22-24 June*, pp. 634–639.

[26] Shang, J., Xiao, C., Ma, T., Li, H., & Sun, J. (2019). Gamenet: Graph augmented memory networks for recommending medication combination. *proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, Hawaii, USA, 27 January – 01 February*, *33*(01), pp. 1126–1133.

[27] Symeonidis, P., Chaltsev, D., Berbague, C., & Zanker, M. (2022). Sequence-aware news recommendations by combining intra-with inter-session user information. *Information Retrieval Journal*, *25*(4), 461–480.

[28] Najmani, K., El habib, B., Sael, N., & Zellou, A. (2019). A comparative study on

recommender systems approaches. *Proceedings of the 4th International Conference on Big Data and Internet of Things, Rabat, Morocco, 23-24 October*, pp. 1–5.

[29] Mohamed, M. H., Khafagy, M. H., & Ibrahim, M. H. (2019). Recommender systems challenges and solutions survey. *2019 International Conference on Innovative Trends in Computer Engineering (ITCE), Aswan, Egypt, 2-4 February*, pp. 149–155.

[30] Felfernig, A., Ninaus, G., Grabner, H., Reinfrank, F., Weninger, L., Pagano, D., & Maalej, W. (2013). An overview of recommender systems in requirements engineering. *Managing requirements knowledge*, 315–332.

[31] Peng, Y., Li, S., Gu, W., Li, Y., Wang, W., Gao, C., & Lyu, M. R. (2022). Revisiting, benchmarking and exploring api recommendation: How far are we? *IEEE Transactions on Software Engineering*, *49*(4), 1876–1897.

[32] Thung, F. (2016). Api recommendation system for software development. *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE), Singapore, Singapore, 3-7 September*, pp. 896–899.

[33] Tang, M., Xia, Y., Tang, B., Zhou, Y., Cao, B., & Hu, R. (2019). Mining collaboration patterns between apis for mashup creation in web of things. *IEEE Access*, *7*, 14206–14215.

[34] Cao, B., Liu, X. F., Liu, J., & Tang, M. (2017). Domain-aware mashup service clustering based on lda topic model from multiple data sources. *Information and Software Technology*, *90*, 40–54.

[35] Hu, R., Chen, J., Liu, J., & Nian, Q. (2019). Mdt: A multi-description topic based clustering approach for composite-service discovery. *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Zhangjiajie, China, 10-12 August*, pp. 130–137.

[36] Liang, T., Chen, Y., Gao, W., Chen, M., Zheng, M., & Wu, J. (2019). Exploiting user tagging for web service co-clustering. *IEEE Access*, *7*, 168981–168993.

[37] Zhao, H., Wang, J., Zhou, Q., Wang, X., & Wu, H. (2019). Web api recommendation with features ensemble and learning-to-rank. *CCF Conference on Big Data, Wuhan, China, 26–28 September*, pp. 406–419.

[38] Lei, C., Dai, H., Yu, Z., & Li, R. (2020). A service recommendation algorithm with the transfer learning based matrix factorization to improve cloud security. *Information Sciences*, *513*, 98–111.

[39] Liu, L., Bahrami, M., Park, J., & Chen, W.-P. (2020). Web api search: Discover web api and its endpoint with natural language queries. *International Conference on Web Services, Honolulu, HI, USA, 18–20 September*, pp. 96–113.

[40] Das, D., Sahoo, L., & Datta, S. (2017). A survey on recommendation system. *International Journal of Computer Applications*, *160*(7).

[41] Simpson, J. (2022). 20 impressive api economy statistics: Nordic apis —. https://nordicapis.com/20-impressive-api-economy-statistics/

[42] Sharma, M., & Mann, S. (2013). A survey of recommender systems: Approaches and limitations. *International journal of innovations in engineering and technology*, *2*(2), 8–14.

[43] Rahman, M. M., Liu, X., & Cao, B. (2017). Web api recommendation for mashup development using matrix factorization on integrated content and network-based service clustering. *2017 IEEE International Conference on Services Computing (SCC), Honolulu, HI, USA, 25-30 June*, pp. 225–232.

[44] Wang, F., Wang, L., Li, G., Wang, Y., Lv, C., & Qi, L. (2022). Edge-cloud-enabled matrix factorization for diversified apis recommendation in mashup creation. *World Wide Web*, *25*(5), 1809–1829.

[45] Cao, B., Shi, M., Liu, X. F., Liu, J., & Tang, M. (2016). Using relational topic model and factorization machines to recommend web apis for mashup creation. *Asia-Pacific Services Computing Conference, Zhangjiajie, China, 16-18 November*, pp. 391–407.

[46] Li, H., Liu, J., Cao, B., Tang, M., Liu, X., & Li, B. (2017). Integrating tag, topic, co-occurrence, and popularity to recommend web apis for mashup creation. *2017 IEEE International Conference on Services Computing (SCC), Honolulu, HI, USA, 25-30 June*, pp. 84–91.

[47] Xu, Y., Zhang, H., Gao, H., Song, S., Yin, Y., Hei, L., Ding, Y., & Barroso, R. J. D. (2021). Preference discovery from wireless social media data in apis recommendation. *Wireless Networks*, *27*(5), 3441–3451.

[48] Yao, L., Wang, X., Sheng, Q. Z., Benatallah, B., & Huang, C. (2018). Mashup recommendation by regularizing matrix factorization with api co-invocations. *IEEE Transactions on Services Computing*.

[49] Hao, Y., Fan, Y., Tan, W., & Zhang, J. (2017). Service recommendation based on targeted reconstruction of service descriptions. *2017 IEEE International Conference on Web Services (ICWS), Honolulu, HI, USA, 25-30 June*, pp. 285–292.

[50] Xiao, Y., Liu, J., Hu, R., Cao, B., & Cao, Y. (2019). Dinrec: Deep interest network based api recommendation approach for mashup creation. *International Conference on Web Information Systems Engineering, Hong Kong, China, 19–22 January*, pp. 179–193.

[51] Zhou, Y., Yang, X., Chen, T., Huang, Z., Ma, X., & Gall, H. (2020). Boosting api recommendation with implicit feedback. *arXiv preprint arXiv:2002.01264*.

[52] Gu, Q., Cao, J., & Liu, Y. (2021). Csbr: A compositional semantics-based service bundle recommendation approach for mashup development. *IEEE Transactions on Services Computing*.

[53] Ali, G., & ElKorany, A. (2014). Semantic-based collaborative filtering for enhancing recommendation. *KEOD, Rome, Italy, 21-24 October*, pp. 176–185.

[54] Qi, L., Song, H., Zhang, X., Srivastava, G., Xu, X., & Yu, S. (2021). Compatibility-aware web api recommendation for mashup creation via textual description mining. *ACM Transactions on Multimidia Computing Communications and Applications*, *17*(1s), 1–19.

[55] Gu, Q., Cao, J., & Peng, Q. (2016). Service package recommendation for mashup creation via mashup textual description mining. *2016 IEEE International Conference on Web Services (ICWS), San Francisco, CA, USA, 27 June - 02 July*, pp. 452–459.

[56] Lin, C., Kalia, A., Xiao, J., Vukovic, M., & Anerousis, N. (2018). Nl2api: A framework for bootstrapping service recommendation using natural language queries. *2018 IEEE International Conference on Web Services (ICWS), San Francisco, CA, USA, 2-7 July*, pp. 235–242.

[57] Almarimi, N., Ouni, A., Bouktif, S., Mkaouer, M. W., Kula, R. G., & Saied, M. A. (2019). Web service api recommendation for automated mashup creation using multi-objective evolutionary search. *Applied Soft Computing*, *85*, 105830.

[58] Jiang, B., Chen, Y., Wang, Y., & Liu, P. (2019). Service discovery method for agile mashup development. *CCF Conference on Computer Supported Cooperative Work and Social Computing, Kunming, China, 16-18 August*, pp. 30–49.

[59] Xie, X., Zhang, J., Ramachandran, R., Lee, T. J., & Lee, S. (2022). Goal-driven context-aware next service recommendation for mashup composition. *arXiv preprint arXiv:2210.14127*.

[60] Zhang, B., Sheng, L., Jin, L., & Wen, W. (2019). Rasop: An api recommendation method based on word embedding technology. *International Symposium on Intelligence Computation and Applications, Guangzhou, China, 16-17 November*, pp. 281–295.

[61] Thung, F., Oentaryo, R. J., Lo, D., & Tian, Y. (2017). Webapirec: Recommending web apis to software projects via personalized ranking. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *1*(3), 145–156.

[62] Li, C., Zhang, R., Huai, J., & Sun, H. (2014). A novel approach for api recommendation in mashup development. *2014 IEEE International Conference on Web Services, Anchorage, AK, USA, 27 June - 02 July*, pp. 289–296.

[63] Xie, F., Chen, L., Lin, D., Zheng, Z., & Lin, X. (2019). Personalized service recommendation with mashup group preference in heterogeneous information network. *IEEE Access*, *7*, 16155–16167.

[64] Alshangiti, M., Shi, W., Liu, X., & Yu, Q. (2020). A bayesian learning model for design-phase service mashup popularity prediction. *Expert Systems with Applications*, *149*, 113231.

[65] Xiong, R., Wang, J., Zhang, N., & Ma, Y. (2018). Deep hybrid collaborative filtering for web service recommendation. *Expert systems with Applications*, *110*, 191–205.

[66] Cao, Y., Liu, J., Cao, B., Shi, M., Wen, Y., & Peng, Z. (2019). Web services classification with topical attention based bi-lstm. *International Conference on Collaborative Computing: Networking, Applications and Worksharing, London, UK, 19-22 August*, pp. 394–407.

[67] Xie, F., Chen, L., Ye, Y., Zheng, Z., & Lin, X. (2018). Factorization machine based service recommendation on heterogeneous information networks. *2018 IEEE International Conference on Web Services (ICWS), San Francisco, CA, USA, 2-7 July*, pp. 115–122.

[68] Xie, F., Li, S., Chen, L., Xu, Y., & Zheng, Z. (2019). Generative adversarial network based service recommendation in heterogeneous information networks. *2019 IEEE International Conference on Web Services (ICWS), Milan, Italy, 8-13 July*, pp. 265–272.

[69] Thorat, P. B., Goudar, R., & Barve, S. (2015). Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications*, *110*(4), 31–36.

[70] Lops, P., Jannach, D., Musto, C., Bogers, T., & Koolen, M. (2019). Trends in content-based recommendation. *User Modeling and User-Adapted Interaction*, *29*(2), 239–249.

[71] Bai, X., Wang, M., Lee, I., Yang, Z., Kong, X., & Xia, F. (2019). Scientific paper recommendation: A survey. *IEEE Access*, *7*, 9324–9339.

[72] Saraswathi, K., Saravanan, B., Suresh, Y., Senthilkumar, J., et al. (2017). Survey: A hybrid approach to solve cold-start problem in online recommendation system. *Proceedings of the International Conference on Intelligent*

Computing Systems (ICICS 2017–Dec 15th-16th 2017) organized by Sona College of Technology, Salem, Tamilnadu, India.

[73] Zhang, D., Yin, J., Zhu, X., & Zhang, C. (2018). Network representation learning: A survey. *IEEE transactions on Big Data*, *6*(1), 3–28.

[74] Qi, L., He, Q., Chen, F., Dou, W., Wan, S., Zhang, X., & Xu, X. (2019). Finding all you need: Web apis recommendation in web of things through keywords search. *IEEE Transactions on Computational Social Systems*, *6*(5), 1063–1072.

[75] Qi, L., He, Q., Chen, F., Zhang, X., Dou, W., & Ni, Q. (2020). Data-driven web apis recommendation for building web applications. *IEEE Transactions on Big Data*.

[76] Wang, X., Liu, X., Liu, J., Chen, X., & Wu, H. (2021). A novel knowledge graph embedding based api recommendation method for mashup development. *World Wide Web*, *24*(3), 869–894.

[77] Qi, L., Lin, W., Zhang, X., Dou, W., Xu, X., & Chen, J. (2022). A correlation graph based approach for personalized and compatible web apis recommendation in mobile app development. *IEEE Transactions on Knowledge and Data Engineering*.

[78] Gong, W., Zhang, X., Chen, Y., He, Q., Beheshti, A., Xu, X., Yan, C., & Qi, L. (2022). Dawar: Diversity-aware web apis recommendation for mashup creation based on correlation graph. *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, 11-15 July*, pp. 395–404.

[79] Gong, W., Lv, C., Duan, Y., Liu, Z., Khosravi, M. R., Qi, L., & Dou, W. (2021). Keywords-driven web apis group recommendation for automatic app service creation process. *Software: Practice and Experience*, *51*(11), 2337–2354.

[80] Gong, W., Wu, H., Wang, X., Zhang, X., Wang, Y., Chen, Y., & Khosravi, M. R. (2021). Diversified and compatible web apis recommendation in iot. *arXiv preprint arXiv:2107.10538*.

[81] Wu, S., Shen, S., Xu, X., Chen, Y., Zhou, X., Liu, D., Xue, X., & Qi, L. (2022). Popularity-aware and diverse web apis recommendation based on correlation graph. *IEEE Transactions on Computational Social Systems*.

[82] Li, B., & Pi, D. (2020). Network representation learning: A systematic literature review. *Neural Computing and Applications*, 1–33.

[83] Mahapatra, S. (2020). Why deep learning over traditional machine learning? [Available at https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063].

[84] Chen, H., Wu, H., Li, J., Wang, X., & Zhang, L. (2022). Keyword-driven service recommendation via deep reinforced steiner tree search. *IEEE Transactions on Industrial Informatics*.

[85] Huang, Q., Xia, X., Xing, Z., Lo, D., & Wang, X. (2018). Api method recommendation without worrying about the task-api knowledge gap. *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE), Montpellier, France, 3-7 September*, pp. 293–304.

[86] Zhang, Y., Su, J., & Chen, S. (2021). A deep recommendation framework for completely new users in mashup creation. *International Conference on Collaborative Computing: Networking, Applications and Worksharing, Shanghai, China, 16–18 October*, pp. 550–566.

[87] Kang, G., Liu, J., Cao, B., & Cao, M. (2020). Nafm: Neural and attentional factorization machine for web api recommendation. *2020 IEEE international conference on web services (ICWS), Beijing, China, 19-23 October*, pp. 330–337.

[88] Cao, Y., Liu, J., Shi, M., Cao, B., Chen, T., & Wen, Y. (2019). Service recommendation based on attentional factorization machine. *2019 IEEE International Conference on Services Computing (SCC), Milan, Italy, 8-13 July*, pp. 189–196.

[89] Kang, G., Liu, J., Xiao, Y., Cao, B., Xu, Y., & Cao, M. (2021). Neural and attentional factorization machine-based web api recommendation for mashup development. *IEEE Transactions on Network and Service Management*, *18*(4), 4183–4196.

[90] Cao, B., Peng, M., Qing, Y., Liu, J., Kang, G., Li, B., & Fletcher, K. K. (2022). Web api recommendation via combining graph attention representation and deep factorization machines quality prediction. *Concurrency and Computation: Practice and Experience*, *34*(21), e7069.

[91] Nguyen, M., Yu, J., Nguyen, T., & Han, Y. (2021). Attentional matrix factorization with context and co-invocation for service recommendation. *Expert Systems with Applications*, *186*, 115698.

[92] Shi, M., Liu, J., et al. (2018). Functional and contextual attention-based lstm for service recommendation in mashup creation. *IEEE Transactions on Parallel and Distributed Systems*, *30*(5), 1077–1090.

[93] Shi, M., Tang, Y., & Liu, J. (2019). Ta-blstm: Tag attention-based bidirectional long short-term memory for service recommendation in mashup creation. *2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14-19 July*, pp. 1–8.

[94] Ma, Y., Geng, X., & Wang, J. (2020). A deep neural network with multiplex interactions for cold-start service recommendation. *IEEE Transactions on Engineering Management*.

[95] Wu, H., Duan, Y., Yue, K., & Zhang, L. (2021). Mashup-oriented web api recommendation via multi-model fusion and multi-task learning. *IEEE Transactions on Services Computing*.

[96] Lizarralde, I., Mateos, C., Zunino, A., Majchrzak, T. A., & Grønli, T.-M. (2020). Discovering web services in social web service repositories using deep variational autoencoders. *Information Processing & Management*, *57*(4), 102231.

[97] Labbaci, H., Medjahed, B., Binzagr, F., & Aklouf, Y. (2017). A deep learning approach for web service interactions. *Proceedings of the International Conference on Web Intelligence, Leipzig, Germany, 23-26 August*, pp. 848–854.

[98] Bai, B., Fan, Y., Tan, W., & Zhang, J. (2017). Dltsr: A deep learning framework for recommendation of long-tail web services. *IEEE Transactions on Services Computing*.

[99] Dang, D., Chen, C., Li, H., Yan, R., Guo, Z., & Wang, X. (2021). Deep knowledge-aware framework for web service recommendation. *The Journal of Supercomputing*, *77*(12), 14280–14304.

[100] Li, X., Zhang, X., Wang, P., & Cao, Z. (2022). Web services recommendation based on metapath-guided graph attention network. *The Journal of Supercomputing*, *78*(10), 12621–12647.

[101] Yu, C., Hu, R., & Wang, B. (2023). Akgin: An api knowledge graph and intent network based mashup-oriented api recommendation method. *2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Rio de Janeiro, Brazil, 24-26 May*, pp. 261–266.

[102] Wang, Y., Chen, J., Huang, Q., Xia, X., & Jiang, B. (2023). Deep learning-based open api recommendation for mashup development. *Science China Information Sciences*, *66*(7), 1–18.

[103] Wang, X., Xi, M., & Yin, J. (2023). Functional and structural fusion based web api recommendations in heterogeneous networks. *2023 IEEE International Conference on Web Services (ICWS), Chicago, IL, USA, 2-8 July*, pp. 91–96.

[104] He, P., Liu, L., You, D., Shen, L., & Chen, Z. (2023). Bat: Mining binary-api topic for multi-service application development. *2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Rio de Janeiro, Brazil, 24-26 May*, pp. 745–750.

[105] Yu, T., Yu, D., Wang, D., & Hu, X. (2023). Web service recommendation for mashup creation based on graph network. *The Journal of Supercomputing*, *79*(8), 8993–9020.

[106] Xiao, G., Fei, J., Li, D., Wang, C., Cheng, Z., & Lu, J. (2023). Mrhn: Hypergraph convolutional network for web api recommendation. *2023 IEEE 24th International Conference on Information Reuse and Integration for Data Science (IRI), Bellevue, WA, USA, 4-6 August*, pp. 179–184.

[107] Wang, G., Yu, J., Nguyen, M., Zhang, Y., Yongchareon, S., & Han, Y. (2023). Motif-based graph attentional neural network for web service recommendation. *Knowledge-Based Systems*, *269*, 110512.

[108] Zheng, X., Wang, G., Zhang, J., Zhang, Y., Wang, N., Yu, J., & Han, Y. (2023). H-mgsr: A hierarchical motif-based graph attention neural network for service recommendation. *2023 IEEE International Conference on Web Services (ICWS), Chicago, IL, USA, 2-8 July*, pp. 553–562.

[109] Golbandi, N., Koren, Y., & Lempel, R. (2010). On bootstrapping recommender systems. *Proceedings of the 19th ACM international conference on Information and knowledge management, Toronto, ON, Canada, 26-30 October*, pp. 1805–1808.

[110] Goldman, S., Kuzmin, D., Rendle, S., Zhang, L., Alzantot, M., Apte, A., Joshi, A., Kesari, A., Ontanon, S., Subbiah, A., et al. (2022). Bootstrapping interactive recommender systems.

[111] Golbandi, N., Koren, Y., & Lempel, R. (2011). Adaptive bootstrapping of recommender systems using decision trees. *Proceedings of the fourth ACM international conference on Web search and data mining, Hong Kong, China, 9-12 February*, pp. 595–604.

[112] Lee, D., Kang, S., Ju, H., Park, C., & Yu, H. (2021). Bootstrapping user and item representations for one-class collaborative filtering. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 11-15 July*, pp. 317–326.

[113] Wu, H., Zhang, H., He, P., Zeng, C., & Zhang, Y. (2019). A hybrid approach to service recommendation based on network representation learning. *IEEE Access*, *7*, 60242–60254.

[114] Mezni, H., Arab, S. A., Benslimane, D., & Benouaret, K. (2020). An evolutionary clustering approach based on temporal aspects for context-aware service recommendation. *Journal of Ambient Intelligence and Humanized Computing*, *11*(1), 119–138.

[115] Wang, L., Zhang, X., Wang, R., Yan, C., Kou, H., & Qi, L. (2020). Diversified service recommendation with high accuracy and efficiency. *Knowledge-Based Systems*, 106196.

[116] Botangen, K. A., Yu, J., Yongchareon, S., Yang, L., & Sheng, Q. Z. (2019). Integrating geographical and functional relevance to implicit data for web service recommendation. *International Conference on Service-Oriented Comput-*

ing, Toulouse, France, 28–31 October, pp. 53–57.

[117] Karimi, M., Cule, B., & Goethals, B. (2019). On-the-fly news recommendation using sequential patterns. *INRA@ RecSys, Copenhagen, Denmark, 16-20 September*, pp. 29–34.

[118] Wang, J., Hoi, S. C., Zhao, P., & Liu, Z.-Y. (2013). Online multi-task collaborative filtering for on-the-fly recommender systems. *Proceedings of the 7th ACM conference on Recommender systems, Hong Kong, China, 12-16 October*, pp. 237–244.

[119] Safran, M., & Che, D. (2017). Real-time recommendation algorithms for crowdsourcing systems. *Applied Computing and Informatics*, *13*(1), 47–56.

[120] Zhang, Y., & Chen, X. (2018). Explainable recommendation: A survey and new perspectives. *arXiv preprint arXiv:1804.11192*.

[121] Brown, C., Middleton, J., Sharma, E., & Murphy-Hill, E. (2017). How software users recommend tools to each other. *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), Raleigh, NC, USA, 11-14 October*, pp. 129–137.