CONSTRUCTING EVOLVING WEB SERVICE SOCIAL NETWORKS FOR WEB SERVICE DISCOVERY

A THESIS SUBMITTED TO AUCKLAND UNIVERSITY OF TECHNOLOGY
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Supervisors

Associate Professor Jian Yu

Dr Ji Ruan

October 2021

By

Olayinka Adeleye

School of Engineering, Computer and Mathematical Sciences

Abstract

Web services have been one of the major drivers of distributed service economy, supporting businesses on global scale. They enable cross-organisational functionality integration over the Web and thus are the foundation of modern distributed service-based systems. However, despite the rapid and continual increase of Web services available on the internet, the discovery and uptake of appropriate Web services by businesses on a Web scale is still a great challenge. The reasons for this meager uptake include (i) *isolation of Web services in their ecosystem* (ii) *poor scaling mechanism for Web service ecosystem*, (iii) *the lack of social relationships among related Web services* and (iv) *inadequate semantic information for facilitating semantic-oriented Web service discovery, and the vocabulary gaps between the service functional descriptions given by service providers and the user's Web service query*. To fill these research gaps, there is a need for a Web service discovery framework which can easily be scaled, and can simulate social interactions between Web services based on their social, functional and non-functional attributes, and in turn improve Web services discoverability. This thesis aims to contribute to the service computing domain by developing a service discovery framework that can assist Web service consumers including corporations in fulfilling their ever increasing service needs, and help them benefit maximally from the advancements of Web service technology. In particular, the thesis has addressed the Web service discovery challenges from the *complex network* perspective by constructing evolving Web service social networks which enables social links formation among Web

services based on well-defined complex network theoretical procedures. This will allow the integration of Web service ecosystem properties such as service sociability and functionality, and Web service network properties into the discovery framework, and thereby help in enhancing service discoverability. The thesis follows three key pathways in addressing the Web service discovery challenges mentioned above.

In the first pathway, a critical study of Web service ecosystem which involves investigating the underlying mechanisms that drive the evolution and interactions of Web services in their ecosystem is conducted. This is important to understand the structure of the Web service ecosystem, the evolving properties that characterized its continual growth and nature of relationships or interactions withing the system. This is achieved by using network analysis approach to study the social interactions of existing Web services and their compositions. The study analysed the dynamical properties of a typical Web service ecosystem, and investigated the popularity distribution of Web services in the ecosystem in order to get clear insight into the social interactions among Web services. Distributions of different interaction patterns that appear in form of network *motifs* were also analysed . In addition, various topological properties similar to the ones that characterised most evolving real world network systems are measured in the service ecosystem. Then, the key attraction dimensions, specifically *preferential attachment* and *similarity* within the Web service system are quantified.

In the second pathway, the challenge of Web service *isolation* and its effect on service discoverability are addressed. Based on the results of the analysis conducted in first pathway, two unique approaches for constructing evolving Web service social networks, which follow the *Barabási-Albert* and the *Popularity-Similarity Optimization* complex network theoretical procedures have been proposed. Both approaches enable simulation of social links and the incorporation of social properties such as *popularity*, and topological properties into the Web service discovery framework. For *Barabási-Albert* based approach, the network is built solely on the principle of *growth* which is driven by

popularity attractiveness , and for the *Popularity-Similarity Optimization* model, certain trade-offs which exist between the two Web services attraction dimensions including *popularity* and *similarity* are exploited to facilitate link formation between the services.

Finally, in the third pathway, an evolving complex network-based Web service discovery service that exploits functional, social and network properties to find, select and rank Web services was proposed. The discovery service employs a novel motif-based page-rank feature with Google custom service to facilitate node ranking based on the network patterns, functional descriptions and popularity information of the Web services. The effectiveness of the proposed discovery method has been demonstrated by conducting extensive experiments on a real-world dataset crawled from *Programmableweb.com*.

Contents

List of Tables

List of Figures

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a university or other institution of higher learning.

_____

Signature of candidate

Publications

Adeleye, O., Yu, J., Wang, G., Yongchareon, S. (2021). Constructing and Evaluating Evolving Web-API Networks-A Complex Network Perspective. IEEE Transactions on Services Computing.

Adeleye, O., Yu, J., Yongchareon, S., Han, Y., Sheng, Q. (2020). Complex Network-Based Web Service for Web-API Discovery. In Proceedings of the Australasian Computer Science Week Multiconference (pp. 1-10).

Adeleye, O., Yu, J., Ruan, J., Sheng, Q. Z. (2020). Evaluating Random Walk-Based Network Embeddings for Web Service Applications. In Australasian Database Conference (pp. 198-205).

Adeleye, O., Yu, J., Yongchareon, S., Han, Y. (2018). Constructing and evaluating an evolving web-API network for service discovery. In International Conference on Service-Oriented Computing (pp. 603-617). Springer, Cham.

Adeleye, O., Yu, J., Yongchareon, S., Sheng, Q. Z., Yang, L. H. (2019). A fitness-based evolving network for web-APIs discovery. In Proceedings of the australasian computer science week multiconference (pp. 1-10).

Adeleye, O., Yu, J., Yongchareon, S. (2021). An Evolving Complex Network-Based Web Service for Web-API Discovery. World Wide Web, Internet and Information System Journal. (*Submitted 2021*).

Acknowledgements

I would like to thank everyone who contributed to the successful completion of my doctoral study. I would have never accomplished this thesis without the love and support of many people. I would like to express my appreciation and gratitude to my primary supervisor, Associate Professor Jian Yu, who offered valuable support, encouragement and understanding throughout the course of my PhD journey. His enthusiasm inspired me greatly to go extra mile in my research and I will forever be grateful to him. In addition, I would like to thank him and also acknowledge the support of the School of Engineering, Computer, and Mathematical Sciences (SECMS) for funding my publications, conferences, and providing the PhD Fees Scholarship and teaching opportunities for me in the school.

I would also like to express my sincere gratitude to my second supervisor Dr Ji Ruan for his support, guidance and motivation. Dr Ji provided me with feedback and insightful suggestions throughout this long journey. I am also thankful to my co-authors Professor Guiling Wang, Professor Quan Z. Sheng, Dr Sira Yongchareon and Professor Yanbo Han for their valuable contribution in various aspects of my research.

I thank all my research colleagues and friends in the Service and Cloud Computing Research Lab (SCCRL) and WT Level 7 building, especially Naga Kunchala, Pinal Shah, Khavee Botangen, Herman Wandabwa, Alan Zhang, Ashad Khan and Bryce Anthony for the friendship and support. I would also like to say thank you to my Pastor and guardian, Pastor Ayo Martins and his wife for their prayers and support. Finally, I would like to express my deepest gratitude to my immediate family for their exceptional love and support throughout the course of my PhD. I am especially grateful to my mother Mrs Elizabeth Modupe Adeleye for her prayers and support, and my lovely wife Alverty Stephanie for her love, care , support and sacrifice, and my siblings Temiloluwa, Feranmi and Victor Adeleye for being my source of inspiration during this journey.

To all of you, I say many thanks, *E se pupo!*

Dedication

With great love and respect, I dedicate this work to my beloved Wife *Mrs Alverty Stephanie Olayinka* and my wonderful mother *Mrs Elizabeth Modupe Adeleye* and my siblings : Temiloluwa, Feranmi and Victor.

Chapter 1

Introduction

Service-Oriented Architecture (SOA) and its leading implementation technology called Web services have changed the way software engineers design, develop and maintain enterprise applications (Maamar, Hacid & Huhns, 2011). SOA brings about a paradigm shift in software development and has revolutionized information system development processes, transforming them from component-based to service-based (Xu, Cao, Hu, Wang & Li, 2013). This service-based paradigm enables improvement in the time, cost and effort required to build service-based systems and encourages reusability (He, Yan, Jin & Yang, 2014). Central to this computing evolution is the *Web service*, which is characterized as loosely-coupled, self-contained, self-describing, Web accessible, modular programming functions that can be published, discovered and invoked across the Web (Rostami, Kheirkhah & Jalali, 2013; M. P. Papazoglou, Traverso, Dustdar & Leymann, 2007). Web services enable cross-organizational functionality integration over the Web and make functional building blocks accessible over standard Internet protocols, independent of platforms and programming languages (Z. Wu, Deng & Wu, 2014; W. Chen, Paik & Hung, 2015a). They have been one of the major drivers of distributed service economy, supporting enterprises on global scale. Businesses can now dynamically compose Web services to perform complex transactions with minimal programming.

Modern Web services with features such as *RESTful architecture*, *JSON data*, and/or *JavaScript interface* are usually called *Web-APIs* in order to distinguish them from the traditional SOAP-based Web services; and multiple Web-APIs can be quickly composed into a Web-page or application called *composite service* or $mashup$ (Benslimane, Dustdar & Sheth, 2008). This process shortens software development life cycle and results into the formation of the so-called *Web service ecosystem* (Huang, Fan & Tan, 2012a; Barros & Dumas, 2006a; Lyu et al., 2014), where new services emerge, some old ones perish, and service vendors and developers collaborate to develop innovative software solutions. Companies such as *Google*, *Amazon*, *Twitter* and *Facebook* have encapsulated some of their functionalities as Web-APIs for easy consumption and advertised or published them via Web service registry like *ProgrammableWeb.com*. Many real-world applications such as online shopping, weather forecast, social media and disaster prevention (Kavitha & Anuvelavan, 2015; Lee, Niko, Hwang, Park & Kim, 2011) invoke Web services via accessible endpoints to implement their functionalities. Consequently, there are tens of thousands Web services currently available on the internet and the number continues to grow rapidly.

To build a service-based system, three key stages are generally involved (N. Zhang et al., 2018; He et al., 2014), including: (i) *system planning*, where requirements and tasks required to implement the system functionalities are determined, (ii) *service discovery*, where the discover and retrieval of set of candidate that can potentially meet user's functional requirements from the service registry, and (iii) *service selection*, which involves selection of highly, functional relevant service from the retrieved candidate services. Over the years, the service discovery stage have become a very challenging stage due to the continual, rapid increase in the number and diversity of Web services, with myriads of semantically similar service functionalities in the service ecosystem (W. Chen, Paik & Hung, 2015b; J. Wang, Gao, Ma, He & Hung, 2017; He et al., 2017, 2014).

1.1 Web Service Discovery for Modern Service-Based Systems

This section describes Web service discovery process and the state-of-the-art. In particular, it explains service discovery processing steps with the requirements imposed on each of the steps, other related concepts and challenges.

1.1.1 Definition

At an abstract level, Web service discovery is defined as process of finding and obtaining appropriate set of services which can potentially fulfil specific user's requirements (Metrouh & Mokhati, 2013). The main essence of this process is to retrieve the service description documents that satisfy the service requesters' (service consumer who requests for services) queries from central or distributed service repository (a database of service description records). This process is often referred to as a service by itself – *discovery service* (Sukkar, 2010). Following the discovery of set of candidate services is the selection process, where the most suitable service(s) for the service consumer's request is selected and then invoked. Figure 1.1 shows the main components of a Web service discovery process. Service consumer initiates the service discovery process by formulating a *service request* in the form of query in order to find appropriate candidate services in the Web service repository or registry. If the service which will be invoked is already known beforehand, the requester directly binds the service, otherwise, a *service broker* (usually provides the service registry where the services are published) may be involved in the invocation process. After service discovery, binding, invocation and execution follow. The *service request* is usually a keyword-based description of service requirements (which could include functional and non-functional capabilities of Web service) by the consumer. Service producers publish

the functional descriptions of their services when advertising the services through a *service broker* on the registry. Matching engines are usually added to the Web service registry to capture the consumers' *service requests* and match them with the service advertisement records available in the registry.

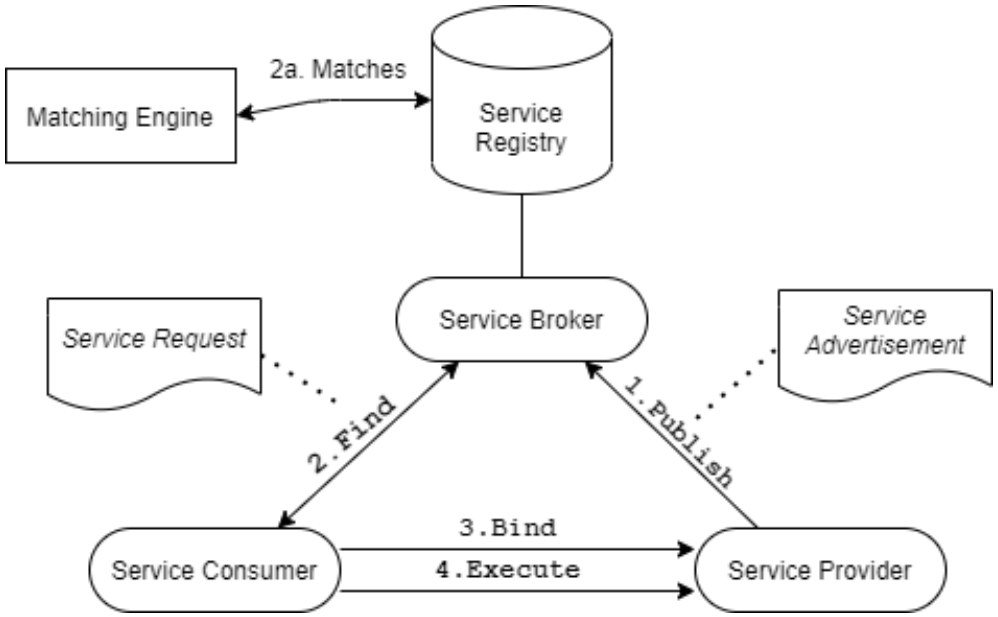


Figure 1.1: Service Discovery Referenced Architecture (Dillon et al., 2007; Sukkar, 2010; Schulte, 2010)

1.1.2 Challenges

The following are the basic concepts and related challenges that affect the Web service discovery process based on Figure 1.1:

- *Service Registry:* The service registry is a search-able directory where service descriptions can be published and searched. It enables the matchmaking, discovery, ranking and selection process, and could be in two forms: the *document-based* registry, which allows providers to publish Web service information by storing XML-based service documents including business profiles and technical specification, and the *metadata-based* service registry, which captures attributes of the

service (M. Papazoglou, 2008). Web service registry could evolve into a logical collection of Web services commonly called *Web service ecosystem* (Barros & Dumas, 2006b), where some new services emerge, some old ones perish and service vendors and developers collaborate to develop innovative software solutions. A typical representation of Web service ecosystem is the *ProgrammableWeb* [1], which is currently the largest online Web-API directory , with over 23000 RestFul Web services belonging to more than 400 predefined categories and over 7,000 service compositions (mashups) as at June 2020. The perishing of some existing Web services and the emergence of new ones coupled with their dynamic collaborations drive the evolution of the service ecosystem over time (Huang, Fan & Tan, 2012b).

For most Web service ecosystems, one of the main issues is the *isolation* of Web services, which in turn limits their discoverability (W. Chen, Paik & Yen, 2017; Huang et al., 2012a; Maamar, Hacid & Huhns, 2011). Web service registries like *ProgrammableWeb* and *Mashape* [2] consider Web service ecosystem as *isolated* functional islands, where services are registered by diverse providers independently and progressively without considering relevant dynamic information or continuous social interactions that exist among the services which could influence their discovery. For instance, in ProgrammableWeb, Web-APIs have categories, and several Web-APIs can be invoked in a mashup, however, there is no *direct* connection between two Web-APIs. The reason behind this is that Web-APIs in ProgrammableWeb registry are registered by diverse service providers independently over time, and the connections or social relationships between Web-APIs are never directly created or defined. Moreover, the current state of service-oriented architecture does not sustain the advantages of Web services as it

[1] https://www.programmableweb.com/
[2] https://rapidapi.com

limits the exposure and uptake of Web services, especially those complying with the SOA paradigm. Current SOA-based applications are designed independently and primarily for closed environment, hence, several challenges such as how to discover the Web services, where to advertise them for high quality and immediate exposure, and how to recommend appropriate services for composition are yet to be resolved (Maamar, Hacid & Huhns, 2011). A handful of Web services are related based on their *co-occurrences* in service compositions, however, a large number of services which are not involved in any compositions and therefore cannot be discovered through their social interactions or by following social links in the service environment. For instance, in programmableWeb, as at June 2020, only about 1525 Web-APIs have ever been invoked in mashups (less than 11% of the total Web-API in the repository) (Adeleye, Yu, Yongchareon & Han, 2018). According to Duan and Tian (2017), 75% of the Web services currently published on the internet have not been discovered or invoked . Existing Web service discovery approaches mainly exploit Web services functional descriptions and ignore internal social relationship that exist among these services which could influence their discoverability (W. Chen et al., 2015a). Some existing works (Maamar, Faci et al., 2011; Maamar, Hacid & Huhns, 2011; W. Chen & Paik, 2013; W. Chen et al., 2015b; Metrouh & Mokhati, 2013) have emphasised the significance of service social activities for enhancing service discoverability and improving users interactions within Web service discovery system. Since Web services are primarily developed to be composed with other Web services, it important to capture the history of their social interactions with peer services, and use this information to facilitate their discovery. The social interaction information can provide significant insights into the drivers and *micro-level* dynamics of service interactions including the properties that stimulate the evolution of the Web service ecosystem (W. Chen et al., 2015a; Huang et al., 2012a).

- *Service Advertisement:* involves the publication of Web services by offering and exposing their attributes including the non-functional and the functional capabilities on a service registry using textual descriptions. The service description could be more or less formal and comprehensive, depending on the type of the registry used and the expressiveness of the description.

  One of the key factors that affects the discovery process is the ability of service providers to describe in detail the capabilities of their services in the description. Since most service discovery solutions mainly rely on the initial descriptions written and published by service providers, the lack of comprehensiveness in describing the potential application scenario of the services , and the failure to emphasize the applicability targeted for specific requirements in the initial service descriptions provided by the service manufacturer or provider, could limit the overall performance of discovery system (Zhong, Fan, Tan & Zhang, 2016; Schulte, 2010). It is worth nothing that applicability also *evolves* with time, that is, certain services that used to be best match for a particular usage scenario may not be the most suitable again as time goes by, while others may be better suited for the scenario.

- *Service Request:* Service requests are formulated in service queries, where the declaration and descriptions of the service requirements are made by the service requester regarding the functional, non-functional and technical service capabilities (Schulte, 2010). The ability of the service consumer to clearly describe their requirements could also affect the overall discovery process. However, some consumers may not know the right expression or keywords to use when searching for a service.

- *Matching Engines:* These are usually integrated with the service registries with the main function of matching the service profiles in the registry with the service

request of the user. The matching process involves a pairwise comparison of a service function description as published by the provider and the service request. Numerical values that express how similar the descriptions and request are often generated during the process. The engines could be a *syntactic-based* or *semantic-based* or an *hybrid* matching engine (Cardoso, 2007).

A large body of research have been committed into addressing various challenges related to Web service discovery and recommendation. Most of these works have focused mainly on exploiting Web services descriptions for enhancing their discoverability, and they can be classified into two large groups based on the discovery methods used: The *syntactic-oriented* approaches (He et al., 2014; Dong, Halevy, Madhavan, Nemes & Zhang, 2004; Halevy, Nemes, Dong, Madhavan & Zhang, 2004) and the *semantic-aware* approaches (J. Wang et al., 2017; F. Chen, Lu, Wu & Li, 2017; Y. Wang, Lin, Wu & Zhang, 2017; Rodriguez-Mier, Pedrinaci, Lama & Mucientes, 2015; Naim, Aznag, Quafafou & Durand, 2016; Aznag, Quafafou & Jarir, 2014; Roman, Kopeckỳ, Vitvar, Domingue & Fensel, 2015; Lu, Cai, Che & Lu, 2016; Garriga et al., 2018). The *syntactic-oriented* approaches find and retrieve Web services by matching the keywords in the service functional descriptions with those of service requester queries using the various information retrieval techniques including Term Frequency-Inverse Document Frequency (TF-IDF) technique and Vector Space Model (VSM) (He et al., 2014, 2017; Manning, Raghavan et al., 2008). Various extensions of the syntactic approaches exist, including the integration of information retrieval techniques with clustering algorithms (Dong et al., 2004; Cong, Fernandez, Billhardt & Lujak, 2015), and structural matching of Web Service Definition Language (WSDL) (Y. Wang & Stroulia, 2003). For the *semantic-aware* approaches, semantic similarities between services descriptions and service requesters' queries are used to discover services related to user's requirements. While some of these approaches explore ontological-based semantics

WSDL like OWL-S and SAWSDL with logic-based reasoning algorithms to discover Web services, others utilize latent topic models such as Latent Dirichlet Allocation (LDA) (Cassar, Barnaghi & Moessner, 2013)), Bi-term Topic Models (BTM) (Y. Wang et al., 2017), and leverage the semantic relationships among words using tools like *WordNet* (F. Chen, Lu et al., 2017; Lu et al., 2016).

Despite various advancements of Web service discovery approaches, the discovery and uptake of Web services by businesses or other service consumers are still significantly less than initially anticipated. Recent studies (Maamar, Faci et al., 2011; Duan & Tian, 2017; Huang et al., 2012a; W. Chen & Paik, 2013; Lizarralde, Mateos, Rodriguez & Zunino, 2019; Jiang, Lee & Hu, 2012) show that most services published on the Web have not been discovered or invoked. Only very few Web services on the internet have been discovered, composed or invoked. For instance, while there are hundred of thousands Web services currently available in different Web service repositories with trillions of Web pages on the Web, the number of publicly available Web services in a service composition system is less than 8000. This meager result does not only limits the Web service ecosystem but also leads to continuous, vicious circle of Web service creation, publication, and composition by service provider without an effective discovery solution for consumers to find existing services (W. Chen et al., 2015a).

1.2 Social Web Service Discovery

In this section, the motivations and challenges of using Web service social network to facilitate service discovery is discussed.

1.2.1 Web Service Sociability

Web service sociability is one of the properties of Web services which describe the ability of a service to interact well with other related services, and it is usually captured

by a network model known as *Web service social network* (L. Chen, Wang, Yu, Zheng & Wu, 2013). Web services interact with each other through composition, collaboration and substitution operations, and their functionalities with the respective Quality of Services (QoS) are interdependent with each other. Thus, the discovery and subsequent engagements of Web services become social activities similar to the collaboration and competition activities found and supported in conventional social network. When "*Socialized*", Web services can provide insight into their historical behaviors and how they have been consumed in the past. Service request and service advertisements could be seen as a reflection of how humans interact in the social world, this imposes social dimension on how Web services must be managed in terms of description, discovery , binding and composition (Maamar, Hacid & Huhns, 2011). Incorporating the social elements into the Web service processes including the discovery process symbolizes new Social Web Services (SWSs), which will either implicitly or explicitly include users interactions in the heart of the Web service life cycle, and enable addition of new functionalities through collaboration and composition.

A key challenge with Web service sociability is how best to capture or model service interactions and the evolution of Web service and their properties such as popularity, and at thesame time retain the mechanisms that drives the consumption of the services, which could also be exploited in supporting their discovery. Recent research works (Maamar, Faci et al., 2011; Metrouh & Mokhati, 2013) advocated the use of social network models for this purpose but did not provide clear theoretical guidelines on how to construct such network.

1.2.2 Using Social Network of Web Service for Discovery

Web service social network is built to capture services social attributes and to help facilitate future social interactions of services. SWS emerges as blend of social computing with service computing, where on one hand, social computing facilitates the study of collective actions, content sharing, information dissemination and the use of information and communication technologies in social context. On the other hand, service computing enables the application development based on the principles of service provisioning and request, loose coupling and cross-organisation data integration. According to Maamar, Faci et al. (2011), Web service social networks can be established by exploiting the following service engagements:

- *Collaboration :* Web services interact by collaboration where different services functionalities are combined for the purpose of satisfying complex user's requirements. Thus, Web service social network can reflect collaborators historical patterns, and a service discovery system can exploit these patterns.

- *Competition :* Just like people compete in a typical social network, Web services compete against each other in their ecosystem especially when they offer similar functionalities and can be use as substitute of each other. In such case, their non-functional and social attributes like popularity and QoS could differentiate and decide which service stands out in the competition. Service Social network could leverage drivers of competition between peer services to improve discovery process (Adeleye, Yu, Yongchareon, Sheng & Yang, 2019).

- *Substitution :* Eventhough Web services that offers similar functionalities compete with each other, they can still be used to support each other when one fail. Hence, the similarity in service functionalities can be exploited to facilitate links formation in Web service social network. Doing this, best substitute can be

identify with respect to user's requirements.

These criteria for establishing Web service social network can be consider independently or integrated as network service social behaviours. They are usually the starting point of constructing Web service social networks (Maamar, Wives et al., 2011). Building a global network that can incorporate these interactions is very challenging. Unlike the conventional social network of people, which is based on absolute cooperation and mutual assistance between their members (no competition), Web services in service social network are very competitive as each service competes to be part of composition or replacement processes (Metrouh & Mokhati, 2013).

1.3 Research Questions

Social Web service discovery system demands for technique and service social network modelling approach that can effectively model the *sociability* property of Web services, and capture both the functional and non-functional attributes of Web services such that they can exploited in enhancing service discoverability. Such approach must also be able to preserve the properties of the services ecosystem including its evolutionary characteristics which are relevant to service discovery.

The work presented in this thesis focuses on addressing *three* key challenges, which affect the use of Web service social network for improving Web service discoverability.

1. **Investigating The Dimensions of Attractiveness In Web Service Ecosystem:** To have a clear insight into the underlining mechanisms that drive the emergence of the social behaviours and different interactions (such as composition or collaboration, substitution and competition) in Web service ecosystem, there is a need for thorough investigation into the topology and dynamical mechanisms that drive the evolution of Web service ecosystem. Such study will provide understanding

into the "what" stimulates *service-service* social interactions in Web service systems and how these mechanisms can be integrated into Web service social network construction to improve service discovery. For instance, in the conventional social network, one key phenomenon that breeds social interactions between individuals is *homophily* (McPherson, Smith-Lovin & Cook, 2001), which describes the tendency of individuals to associate and connect with people of similar attributes. This emphasizes the principle that a connection or contact between individuals of similar attributes occurs at a higher rate than among dissimilar ones. In addition to understanding these mechanisms, it is important to know if Web service system share some common characteristics with related real-world network systems like the World Wide Web (WWW) and the Internet, which have explored network capabilities and characteristics to improve the discovery of their entities (Barabási & Albert, 1999).

Addressing this first challenge raises the research question: ***(RQ1)*** *Is social network behaviours universally existing in Web service ecosystem?*

2. **Constructing Evolving Web Service Social Networks:** While several studies have emphasized the significance of Web service social network in improving service discoverability (Maamar, Faci et al., 2011; Maamar, Hacid & Huhns, 2011; Fallatah, Bentahar & Asl, 2014; W. Chen & Paik, 2013; Metrouh & Mokhati, 2013; Huang, Fan & Tan, 2014a), there is still no clear theoretical basis and guidelines on how to design and construct a Web service social network such that it captures the social attributes of Web services and reflects both the evolutionary and topological properties of the service ecosystem. For instance, Fallatah et al. (Fallatah et al., 2014) proposed to add service-service, user-user, and user-service links to build a service social network. Based on the network, metrics such as user popularity, service market share, and user satisfaction can be

measured. Simulation was done but how to build such network from real-world data was not discussed. Semantic information mined from service descriptions is a good reference for adding links among services. Zakaria Maamar et al. (Maamar, Faci et al., 2011, 2011; Maamar, Wives et al., 2011) advocated for the use of Web service social network for services discovery and discussed various benefits of having such discovery solutions. Similarly, Feng et al. (Feng, Lan, Zhang & Chen, 2015) constructed three types of service networks based on the *subsume*, *sequential-total* (the output of service A covers the input of service B), and *sequential-part* (the output of service A partially covers the input of service B) semantic relations. Clearly such networks are static without considering any dynamical properties. Wang et al. (H. Wang, Feng, Chen, Xu & Sui, 2010) used domain knowledge to calculate the degree of semantic match between any two services and then a threshold can be set to determine the number of links in the network. However, the following gaps exist in the current approaches:

- There is no clear theoretical basis or requirement for the network construction as most of the existing approaches constructed the service network without clear insight into the mechanisms that stimulate service social interactions. Therefore, it is difficult for these approaches to either preserve relevant properties of the service ecosystem or integrate/leverage these properties in service discovery solution.
- Networks produced by these approaches are usually static, which is contrary to the dynamic nature of the real-world social network systems.
- They all struggle with constructing a unified, evolving service social network that captures relevant social attributes of Web services.
- Difficult to map the universal properties of real-world social network with the end products of these approaches. For instance, real-world networks like

WWW and internet evolve by addition of new link, and they also exhibit some common properties such as small-worldliness.

- Most of these approaches only include a subset of the Web services available in the service ecosystem and not every services. For example, some existing works (Lyu et al., 2014; Huang et al., 2012b; Feng et al., 2015) only include service that are involved in a specific interaction such as composition interaction.

Addressing these gaps led to the second research question: ***(RQ2):*** *How can we construct a global evolving social service network of Web services such that it preserves the Web service system properties and reflects the universal properties of commonly found in similar real-world network systems?*

3. **Evolving Web Service Social Network for Web Service Discovery.** *Why evolving Web service social network for discovery ?* Web services, their compositions, providers, consumers, and other service elements such as service context including functional and location information, collectively form an evolving service ecosystem (Huang et al., 2014a). Just like most real world networks, this ecosystem *grows* and increases in size over a period of time. Thus, it is progressively evolving to a social network-like service system, where the service providers develop and publish or advertise Web services with their descriptions in order to expose these services and make it easy to discover by the consumers. Consumers like software developers cannot only interact with the services but can also interact with one another. For instance, a developer can follow, consume and comment on other developers' services. Modern service registries like *ProgrammableWeb.com* have service discovery as one of their core functionalities. Most of these registries adopted *syntactic* approach which enables them be able to discover services through matching of keywords (in services consumers

queries) against the Web services descriptions stored in the registries, which may contain similar keywords. They commonly use information retrieval techniques to facilitate this process.

Eventhough there are recent improvements to the discovery capabilities of service registries, these registries suffer some setbacks (N. Zhang et al., 2018; Maamar, Faci et al., 2011). For instance, these registries treated Web services as isolated functional island. They considered Web services as independent elements in their ecosystem, ignoring their social attributes, and do not consider links among services. Moreover, the syntactic-based registries which mainly rely on keyword matching, and implemented by employing the traditional information retrieval techniques, have their performances greatly affected by *term-mismatch* (B. Cao et al., 2017; Zhong et al., 2016). For example, If a service consumer's request contains multiple function-specific keywords that are partially captured in the service descriptions, the syntactic-based registries tend to perform well and return good matches. However, if there is a wide *vocabulary gap* between service consumer request and the provider descriptions such that the potential usefulness of input keywords is reduced, the ability of the system to retrieve relevant services descriptions greatly reduce because only words within the consumer's request are considers in the search and discovery processes. Such mismatch could stem from: (i) same word having different meaning (*polysemy*), (ii) different words with same meaning (*synonyms*) (iii) words that could have thesame contextual meaning but are not synonyms (iv) inability of service consumer to use the right expression for the required service function (use of ambiguous terms in consumer's request) (Lizarralde, Mateos, Zunino, Majchrzak & Grønli, 2020). The precision of these registries is still low and they cannot address complex service requirements (J. Wang et al., 2017). For the *Semantic-based* approaches,

producing semantic services information is very challenging and time consuming as it requires annotating Web service descriptions including data-types, input and output operations, messages etc. (Lizarralde et al., 2020).

Web service social network with its underlying principles and metrics can provide solutions to these issues and other previously mentioned challenges that Web services face today. It is expected that this approach will not only enables the integration of the service social network properties (including common topological properties) into the discovery processes, but it would also allow service brokers and providers to be able to scale the ecosystem dynamically and provides a *link-as-you-go*, user-oriented service discovery environment for consumers where they can browse, search, find, select and rank services based on the functional, social and system properties.

Addressing the aforementioned service discovery challenges raises the research question: ***(RQ3)*** *How do we exploit the Web service social network properties to facilitate the discovery and selection of component services for service compositions?*

1.4 Research Methodology and Objectives

The research questions discussed in Section 1.3 are answered in remaining chapters of this thesis. The challenges describe in *RQ1* are addressed in Chapter 3, Chapter 4 answered *RQ2*, and *RQ3* is answered in Chapter 5. Following the analytical guidelines describe in engineering research (Wieringa, 2005), three key pathways are followed in answering the research questions. The first pathway is the *problem analysis* phase where the details of the problems to be solved as described in Section 1.3 are specified and investigated. In this pathway, thorough investigation of relevant literature is conducted

and requirements are defined for the research objectives. The second pathway is where the *solution analysis* is conducted. In this phase, solutions are proposed to the problems defined in the first pathway, and the properties of these solutions are then thoroughly investigated. In addition, a comprehensive evaluation of the proposed solutions is carried out to determine whether these solutions can indeed solve the respective problems they are designed to address. The final pathway is where the *implementation analysis* is conducted. Here, the implementation of the proposed solutions is carried out, case studies, baselines and experimental procedures are defined to further evaluate the implemented solutions. The analyses of the problems associated with the research questions enable the identification of sets of specific objectives to be achieved. Table 1.1 presents those objectives.

1.5 Research Contributions

This thesis aims to address the challenges of Web service discovery (as outlined in section 1.1.2) from complex network perspective, and contribute towards the improvement of Web services discoverability by studying different dimensions of attractiveness and social interactions in Web service ecosystem, and incorporating the findings into a new Web service discovery framework. The contributions would be realized by providing answers to the research questions specified in section 1.3. More precisely, the major contributions of this thesis are summarized as follow:

1. **A Complex Network Analysis Approach For Characterizing The Structure And Dynamic Evolution Of Web Service Ecosystem**

   This thesis proposes a comprehensive complex network analysis approach for (i) characterizing the topology and the evolution of the Web service ecosystem

Table 1.1: Research objective with reference to the research questions

Research Question	Specific Objectives
RQ1	– Investigate the underlining mechanisms that drive Web service ecosystems. – Statistically analyze the service-service and service-composition interactions in the service ecosystem, and study the fitting of their popularity distributions to classical distribution functions. – Develop a strategy to identify different dimensions of attractiveness in Web service ecosystem. – Evaluate the global network properties in the Web service ecosystem using service-composition *affiliation* network. – Quantify preferential attachment in the ecosystem and classify it based on the attachment exponent.
RQ2	– Investigate the requirements for constructing Web service networks and the existing network models being used for Web service social network construction. – Develop Web service social network construction strategy that captures services social interactions, preserves both the system and network properties of the service ecosystem. – Develop evolving complex network-based approaches for connecting and publishing Web services including isolated services as linked social Web services on an open Web.
RQ3	– Investigate the existing Web service discovery approaches including network-based approaches. – Examine how certain global network metrics and topological properties can be exploited in facilitating Web service discovery. – Develop an evolving complex network-based Web service discovery model that explores functional, topological and system properties to facilitate the discovery and selection of component services for a typical service composition task. – Evaluate the performance of the complex-network-based discovery model using real-world Web service data.

(ii) studying various real-world network system properties and dynamic mechanisms that are present and drive the progression and the structure of the Web service ecosystem. (iii) extracting relationships among Web services such as *similarity* and co-invocation relationships. and (iv) exploiting the properties of service-composition interaction network for enhancing the discovery process. This analysis form the basis for the proposed discovery solution. For the characterization of Web service ecosystem structure and evolution, the analysis provides an insight into the dynamical processes that drives the continual growth of the Web service system and its evolution, and provides insight into how certain topological properties emerge in the system. The Web service system and network properties are investigated using the network analysis approach. How these properties could be exploited for improving Web service application like service discovery is also considered. In addition, specific dimensions of attractiveness which govern the relationships (including complementary and collaborative relationships) among open APIs in the system are investigated.

2. **A Statistical Approach For Evaluating Preferential Attachment In Web Service Network**

This thesis proposed a systematic approach for measuring and investigating the form of *preferential attachment* (PA) presents in the Web service network. The preferential attachment was suggested as generic driver for network evolution that yields certain topology ubiquitous in real-world evolving network systems. However, it is not clear if this assumptions holds for Web service interaction network. Detecting the presence and form of PA in the service network is crucial to the development of the Web service discovery application as it provide insights into the preferential form of attractions (if it exits in the ecosystem), the Web service popularity distribution, and the scaling factor that describes the growth and

topology of the system. The Preferential Attachment hypothesis indicates that the rate $\Pi(k)$ with which a node with k links acquires new links is a monotonically increasing function of k. The thesis examine this hypothesis to ascertain if PA mechanism holds in Web service context and if indeed the $\Pi(k)$ depends on k. To determine directly from the Web service data the functional form of $\Pi(k)$, the scaling exponent α and the relationship between the two, this thesis presents a maximum-likelihood-based estimation for determining the functional form of $\Pi(k)$, for example if its linear (as in the related systems like internet and WWW), or if it follows some unexplored functional form and what the scaling exponent is for the Web service ecosystem.

3. **Complex Network-based Approaches For Constructing Evolving Web Service Social Network For Service Discovery**

   The thesis propose two evolving complex-network-based approaches to address the problem of Web service *isolation*. These approaches are based on the findings of the initial empirical analysis discussed in points (1) and (2) above. Two main dimensions of attractiveness - *similarity* and *popularity* - in the Web service ecosystem are exploited in the construction of the networks. The approaches follow clear theoretical procedures with a well-defined strategy for growing the Web service networks over time. The first network construction approach is solely based popularity as one key attraction dimension that drives connectivity or interactions in the Web services ecosystem. It utilizes the preferential attachment kernel to simulate links between nodes in the Web service network. The second network construction approach specify the growth of the network in hyperbolic space. The approach exploits certain trade-offs between the two Web service attraction dimensions to facilitate link formation among Web services over a period of time.

4. **Metrics For Evaluating Web Service Social Network**

   This thesis provides a set of metrics that are specifically defined for evaluating the quality of Web service network with respect to the service discovery application. The metrics are used to evaluate both the system and network properties relevant to Web service discovery and recommendation. The use of the metrics is also expanded for comparing the structures and properties of the proposed Web service networks, and their suitability for service discovery purpose. This enables identification of certain correlations (relevant to service discovery) between the Web service system properties and the network properties. Some specific metrics are introduced during the evaluation of Web service networks to provide insightful information that would guide the design and evaluation of the complex-network-based Web service discovery framework. That is, identifying the network metrics and properties that contributes most to improving service discoverability.

1.6 Thesis Structure

The remainder of the thesis is structured as follows:

- **Chapter 2** provides an extensive survey of the concepts and notions considered in this research including the state of the art. The beginning of the chapter provides the necessary backgrounds of the underlying concepts that appear in the Web service discovery processes. In addition, detailed discussion regarding the conceptual perspectives of Web service sociability, and the existing approaches used in addressing of Web services discovery challenges are presented. Furthermore, the applicability of complex network theory in solving related problems to the ones addressed in this thesis is presented. This chapter also renders lists of existing works on Web service discovery which are classified based on the

techniques used and the attribute of Web service explored in implementing their respective discovery solutions . Survey of literature relevant to Web service discovery, Complex network and its applications, and requirement for constructing a network for Web services is performed.

- **Chapter 3** presents the proposed network analysis approach for analysing the Web service ecosystem. The chapter presents the key phases of the analysis: First, from the complex network perspective, an investigation of composition services and Web services interactions, as well as the analysis of Web service popularity distribution in typical Web service ecosystem is presented. Second, the chapter also presents a systematic approach for investigating the presence and form of preferential attachment mechanism in Web service ecosystem, and quantitatively measuring the dimension of attractiveness that drives various interactions in the ecosystem. The chapter also includes discussions about the rationale of these attraction dimensions in Web service discovery context, the various attachment mechanisms and existing research efforts in literature that have exploit these mechanisms to simulate product or services interactions. This chapter is based on the works published in Adeleye, Yu ,Yongchareon and Han (2018); and Adeleye, Yu , Ruan and Sheng (2020).

- **Chapter 4** presents the proposed approaches for constructing evolving social networks for Web service using complex network theory. This chapter provides motivating examples, specifications, and algorithms for constructing a Web service social network that could serve as cornerstone for Web service application such as discovery and recommendation applications. The chapter presents three different approaches for constructing Web services social network using different attachment functions and network growing strategies. It also presents the topological properties of each of the networks and map them with Web service system

properties. The networks evaluations (including comparisons of the networks properties) are also presented in this chapter. The chapter also provides a comprehensive discussion of existing Web service network construction approaches in the literature and their limitations, with emphasis on how the proposed approaches address the limitations. This chapter is based on the work published in Adeleye, Yu ,Yongchareon and Han (2018); and Adeleye, Yu , Yongchareon, Sheng, and Yang (2019).

- **Chapter 5** presents the proposed complex-network-based Web service discovery approach. This chapter provides discussions about how the global Web service social network properties with service functional descriptions are exploited to enhance service discovery processes. The chapter presents a network-based discovery algorithm that leverages *motif-based* page ranking and an online Google page-ranking feature to facilitate service node ranking. The chapter also presents a preliminary study made on a real-world Web service ecosystem to observe how the integration of Web service social and functional attribute through complex network impact service discoverability. The chapter also provides the procedure and results of the experiments conducted to evaluate the proposed approach. In addition, various related works on service discovery approaches are presented. This chapter is based on the work published in Adeleye, Yu , Yongchareon, Sheng, and Yang (2019),

- **Chapter 6** concludes the thesis and presents challenges that will set directions for future work.

Chapter 2

Literature Review

In this chapter, comprehensive review of frameworks, concepts, techniques and algorithms that form the basis of this research work is presented. This thesis is at the crossroads of two major research streams : complex network and service computing. In the complex network stream, complex network theory is employ to study and model the dynamical processes involved in Web service ecosystem in a way that the systems properties can be preserved, and social, functional and topological properties can can be effectively exploited to support Web service discovery application. For the service computing stream, a focus on Web service discovery with the motivation to assist Web service consumers in constructing Web service social networks and to use these networks to discover Web services based on user's requirements. This chapter presents a review of background concepts, principles and frameworks relevant to the two research streams considered in this thesis in order to provide context in which the research is undertaken and understand the framework presented in later chapters . The chapter also presents a survey of the state of the art of existing Web service discovery approaches and discusses existing related works to the research questions outlined in Chapter 1. Section 2.1 describes the basic concepts and terminologies related to the service-oriented architecture and computing paradigm. Section 2.2 introduces on Web service, its related

concepts and attributes, and elaborates on the roles of these attributes in Web service discovery processes. Section 2.3, presents the background of composite or mashups and atomic service. Section 2.4 provides detailed backgrounds and state-of-the-art of Web service discovery. Moreover, the section analyzes Web service discovery existing approaches with focus on methods, techniques and information used to enable the discovery processes. Section 2.5 gives detailed discussion about the Web service social network with focus on social networks, social relationships in Web service community and existing methods used to build Web service social networks. Section 2.8 presents the fundamentals of complex network with focus on complex network theory, models and its existing applications in service computing domain and other related real-world applications.The final section 2.9 provides the summary of the literature reviews and surveys in the chapter.

2.1 Service-Oriented Architecture and Computing

This section presents the underlying architecture of service computing, its background concepts, terminologies and other related components including Web service standards and technologies. The section also describes the frameworks and algorithms in service computing that have been studied as the basis of this research work.

2.1.1 Service-Oriented Architecture

Service-Oriented Architecture (SOA) is an architectural style that service models relies on in building integration-ready applications and reorganizing the capabilities of software applications (which are previously isolated from each others) into an interconnected set of services, each accessible through standard interfaces and messaging protocols (M. P. Papazoglou, 2003). SOA as a distributed software architecture aims to allow end-users to compose diverse functionalities to form applications which are

developed purely from existing services and combining them in an ad-hoc manner.SOA implementation involves collection of loosely-coupled functional building-blocks that are platforms and programming language independent, location-transparent services (Web or micro services) which are capable of communicating with each other via standard interfaces (MacKenzie et al., 2006; Y. Chen, 2018). The architectural style is well applied in software design domain where services are provisioned to other components through a communication protocol over the network. Particularly, the style is applicable when multiple applications running on varied technologies and platforms need to communicate with each other, enabling enterprises to be able to compose and match services to perform business transactions with minimal programming effort (M. P. Papazoglou, 2003).

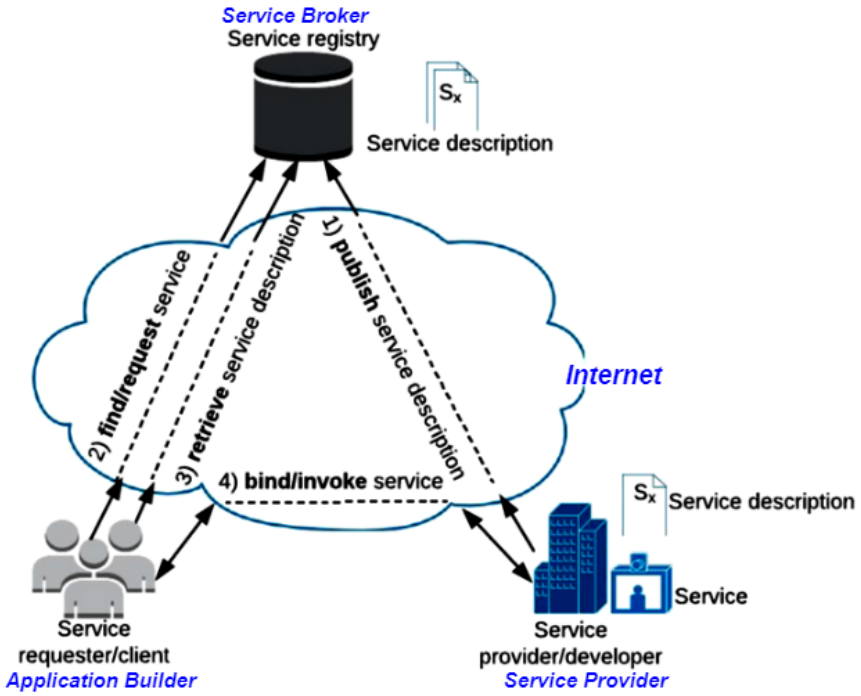


Figure 2.1: A typical Service-Oriented Architecture

Figure 2.1 shows the inherent relationship in a typical SOA among the three participants (*service broker*, *service provider*, and *the application builder* or *service*

requester) realized by interactions that involve specific operations: *publish* service operation, *find/request* services operation, *retrieve* service description operation, and *bind/invoke* service operation. The roles of the participants and the operations explore two service objects: *service description* and service *implementation* (both described in the later of this chapter). The manner in which these roles interact, and the sequence at which they interact are described as follows:

1. **Service provider** and **application builders** (clients/service requester) are both software agents and represent entities such as businesses and software developers. The service providers develop software components inform of classes, objects and functions. Typical service requester seeks services with certain functional descriptions and binds with the service provider through the invocation of the service. The application builder search, find, bind, test, verify, and execute services in their applications dynamically at run-time (Y. Chen, 2018).

2. In general, **service brokers** (service brokers, service registry or service repository are sometimes used interchangeably) are service discovery agencies that propel the activities of providers by ensuring that they are consistent with standards and best practices in the service industry. There principal functionality/role is to make the information related to the web service available to any potential service requester. Service brokers may usually utilizes standards like UDDI and ebXML standards which provide a set of standard service interfaces for registering, advertising and publishing services. Discrete service functionalities are developed independently by diverse service providers based on standard interfaces and are submitted to the service brokers.

3. The application builder searches for services through the internet, the broker's service registry, looking for services that matched his/her requirements and instructions on how to invoke the services. The application builder can then

use the available services to compose new value-added applications. This is considered a high-level programming utilizing service modules to build more complex applications. Doing this, the application builder do no necessarily have to understand or know the low-level programming.

Such a service-oriented architecture gives the application builders the maximum flexibility to choose the best service brokers and the best services.

Figure 2.2 describes the elements of SOA (Krafzig et al., 2005) including the four key abstractions: *application frontend*, *service*, *service repository*, and *service bus*.

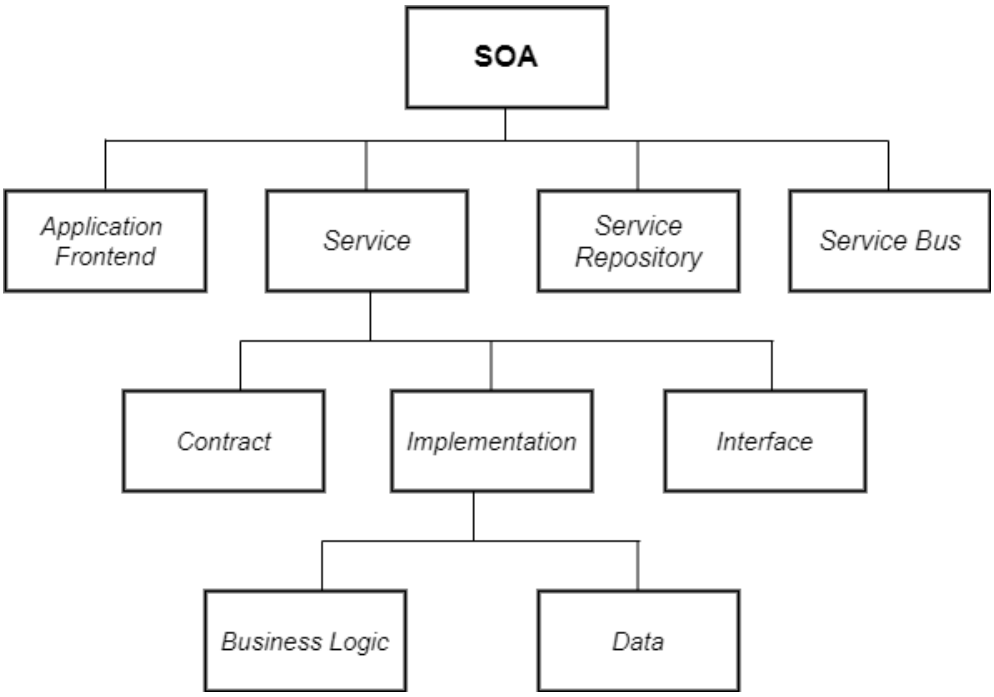


Figure 2.2: Elements of Service-Oriented Architecture (Krafzig et al., 2005)

1. The ***application frontends*** are the active elements of SOA which represent the business process owner and help deliver the values of SOA to the end-users.

2. ***Services*** provide business functionalities such as get geo mapping, get reservation, cancel booking and so on, which could be utilized by the application frontends and other services. In SOA context, services are semantically well-defined abstraction

of a set of computational or physical activities involving a number of resources,put in place to fulfil a client business requirements or needs(Sheng et al., 2014). All business functions that follows SOA architectural style are encapsulated as services. These services exhibit the following characteristic (M. P. Papazoglou et al., 2007; Channabasavaiah, Holley & Tuggle, 2003):

- Services in SOA includes pure business functions, business transactions composed of lower-level functions, and as well as system service functions.
- ***Autonomous*** : All services that follows SOA architectural style are *autonomous*. That is, the service operations are perceived as a black-box or opaque by external components. Service opaqueness in this context indicates that external components do not know how services perform their function, they merely anticipate that they return the expected result. Both the implementation and execution spaces of the application providing the required functionality are encapsulated behind the service interface.
- ***Invocable*** : Generally, services or interfaces can be invoked irrespective of where they are located (locally or remotely), the interconnect scheme or protocol to effect the invocation, or the infrastructure components are required to establish the connection.
- ***Location-Transparent***: Regardless of the location service registry , accessibility of services descriptions by service requester should be transparent and must not be location constrained.
- ***Platform-Independent***: In SOA, services are platform independent, that is, they can be used regardless of service requester's technological platform. Therefore, components such as the descriptions,discovery mechanisms and the related protocols need to follow the SOA universal standards.
- ***Loosely-coupled***: There should be no transnational properties that would

generally apply among the components. Services do not need to know the implementation details of the clients. Loosely coupled property is a loaded term with different dimensions including time/availability, location-transparency and service evolution aspects (Pautasso, Zimmermann & Leymann, 2008). An important dimension of loosely-coupling that concerns *time/availability* involves the ability of service requester to interact with the service provider even when the provider is not available, hence, in this case , service requester or client would not be affected when services have temporary downtime. The location dimension of loosely-coupling property enables service requester/client to discover the location service provider at run-time (dynamic late binding). The evolution dimension indicates the ability to make modification to a service without affecting the client.

- ***Self-contained***: Services are considered self-contained modules with each service having its own state that is independent of the context and state of another service. In thesame vein, services are designed irrespective of the nature, the objective and purpose, and the context of use in the client's system environment.

Technical infrastructural services such as persistence service (Eikermann, Look, Roth, Rumpe & Wortmann, 2017), offer functionalities like store and update data, open cursor. Eventhough this sort of functionalities are very useful when defining and implementing a business function, they have minimal strategic relevance when considered from SOA perspective. More specifically, there should be no technological impact on the high-level structure of the application landscape or dependencies between components. SOA approach must decouple the technical infrastructure services from the business applications, and enables enterprise independence with respect to specific technical implementation or infrastructure

(Krafzig et al., 2005)

3. ***Implementation***: From SOA standpoint, *service* consist of an *implementation* which defines the implementing technology and details of execution procedures to fulfil the functionality of a service. Implementation provides business logic and data. Implementer usually builds SOAs using various Web services standards and service-based technologies including RESTful and SOAP and other related World Wide Web Consortium standards (R. T. Fielding, 2000). Architectures can operate independently of specific implementing technologies, hence implementation can be achieved using a variety of technologies.

4. ***A contract*** specifies the functionalities, usage and constraints for the service *client* (which could be an application frontend or another service).

5. ***Service interface*** physically exposes the service functionality.

6. ***Service bus*** is an inter-operability message bus for implementation, deployment, and management of SOA-based systems, specifically between large-grained heterogeneous enterprise systems. The two key objectives of service bus include: (i) to loosely couple the system taking part in the integration process. (ii) to break-up the integration logic into distinct and easily manageable pieces (M. P. Papazoglou et al., 2007).

SOA provides enterprise with various benefits including the following:

1. From service-based system development standpoint, SOA style advocates a logical way of provisioning services to the end-user's application or other services using discoverable interfaces.

2. SOA can help businesses respond more quickly and more cost-effectively to end-users service needs and changing market conditions (Koch, 2005).

3. SOA promotes reusability at the macro (service) level rather than micro (classes) level, by enabling the composition of value-added, more complex applications from the micro services. It could also enables simplification of interconnection to existing IT assets.

4. SOA indirectly simplify testing and captures many of the best practices of previous software architectures.

In order to realize the various benefits that comes with SOA architectural style, SOA imposes the following requirements:

1. **Implementation Neutrality :** the interface is what matters most and not the implementation. Implementation details of the interacting components should not be depend on. Specifically, the SOA cannot be specific to particular set of programming languages (Singh & Huhns, 2005).

2. **Flexibility configuration :** the system is configured such that flexibility and lateness are accommodated , that is system components are bound to each other late in the process, so that the configuration can change dynamically.

3. **Long lifetime** Because the operations involves computation among autonomous heterogeneous entities in a dynamic environment, provision must be made for error exception handling. That is, the components must exist long enough to be able to detect and eliminate any relevant exceptions, to take corrective action, and to be able to respond to the corrective actions taken by others. A component must exist long enough to be discovered, relied upon, and to generate trust in its behavior.

4. **Granularity :** This is a key design concern that ensures that participants in a typical SOA architectural style are understood at a coarse granularity. This

granularity specifies the scope of business functionality and the structure of the message payload in a service operation that is provided within a SOA. Instead of modelling actions and interactions at detailed level, it is better to capture the key high level qualities that are visible for the purposes of business contracts among the participating entities. Coarse granularity requirement limits dependencies among these entities and reduces communications to a minimal message that has greater importance.

5. **Teams :** Computations in an open systems is more about business partners working as a team rather participant commanding its partner. That is, instead of an individual, a team of cooperating participants is a better modelling unit. A team-oriented perspective is a result of taking seriously a peer-to-peer architecture.

SOA approach does not address several overarching concerns including management, service orchestration, service transaction management and coordination, security and other issues that affects all components in a service architecture (M. P. Papazoglou, 2003).

2.1.2 Service-Oriented Computing

In traditional approaches of software development, the developers takes the software user requirements or request, translates them into specification, and then convert the specifications into executable files that meets or satisfy the user requirements. Different approaches such as object-oriented (OOC), component-based computing and waterfall model are available to translate software specification into an operational system, with each approaches having its own engineering processes and supporting techniques (Y. Chen, 2018). One of the main concerns of the traditional software development approaches is that they are not designed to face the challenges of open environments (Huhns & Singh, 2005).

Service-Oriented Computing (SOC) (also refers to as service computing) emerges as computing paradigm that provides a way to create new architecture that reflects components' trends toward autonomy and heterogeneity. As a computing design paradigm, SOC utilizes services as fundamental elements for developing applications (M. P. Papazoglou, 2003). SOC has reinvented the way software applications are developed, delivered and consumed, and changed the way enterprises work together. SOC relies on Service-Oriented Architecture to build a service model, and seeks to provide computational abstractions, architectures, techniques, and tools to support services broadly. It aims to transform computing resources including hardware and software assets into a service paradigm in which end-users and the resources establish on-demand interactions, binding resources and operations, enabling an abstraction layer that shifts the focus from infrastructure and operation to services for the purpose of promoting cross-organisation integration of functionalities (Bouguettaya et al., 2017). Such integration would not only promote the rapid deployment of distributed software applications but also enables quick and low-cost software development.

Abstractions for Service Computing

SOC can be view in terms of different levels of abstractions ranging from the ones that capture services within an application to those that concerns cross-organisational service applications interactions.The following are different levels of abstractions in which SOC can be considered (Huhns & Singh, 2005) :

1. ***Intra-enterprise Abstraction Level*** : this abstraction level focus on applications inter-operation within an enterprise, which involves enabling connectivity between interacting components, and addressing messaging and semantics problems. Connectivity among the applications could be made possible by protocols such as HTTP. For various business components to understand each other , there

is a need for a clear communicating formatting schema and procedure for encoding and deciphering meaning of messages from each interacting component. XML schema can handle communication formatting but can not decipher the meaning behind a given message. Since meaning generally depends on how various system components processes information, developers need to uncover or reconcile the intentions of interacting components. The reconciliation process assumes that accurate declarative information models exists , however, such models are often poorly maintained in practice or simply are not available. Therefore, developers have to build this models during integration. Another challenge in here is that information models for different systems might express different level of abstractions. Services can encapsulate component behaviour at various levels, and yet describe the behaviour in thesame way, thereby easing process of composing various components. Since most enterprises operate a combination of legacy and other systems and application, different policies have to continuously authenticated and authorized the entities involved in different interactions. Since each of the systems in an enterprise are likely built on different platforms, clearly, compliance to interconnection standards is important. This made interoperability between existing new systems and existing ones challenging.

SOA architectural style addresses all the problems described above. SOA requires that systems policies be made explicit, and thus can organizationally enforce compliance with these policies, thereby simplifying system management. It provide abstractions and tools to model information and relate models, develop processes over the systems, assert and guarantee transnational characteristics, add flexible decision-support , correlate the functioning of the component software systems with the organisations that they represents (Huhns & Singh, 2005).

2. ***Inter-enterprise Abstraction Level*** : Traditionally, businesses have interacted

and interoperated in ad hoc manner, which usually requires human intervention. In some cases, they utilizes rigid standards such Electronic Data Interchange (EDI), which results to difficult to maintain system. In recent time, there have been a growing interest in the use of supply chain management systems and on-demand manufacturing, which further increases the cross-enterprise interactions and processes. This is because businesses that have interact together can enhance their response to customer needs and other business related information, reduce overhead cost, capture and satisfy individual customers preferences and exploit emerging technologies .

SOC provides the ability for interacting parties to capture their behaviours such that each party can apply its local policies autonomously, and still achieve effective and clear cross-organisation processes. SOC also provides support for *dynamic selection* where business entities can pick business partners that offer best terms in a flexible manner, and select the partners to optimize their quality-of-services criteria including performance, reliability, trustworthiness and availability. In addition, SOC enables abstractions through which the state of a business transaction can be captured and effectively manipulated. By enabling this, dynamic selection is further exploited to yield application-level fault tolerance (Huhns & Singh, 2005).

3. ***Infrastructure and Software Component Abstraction Levels*** : It is challenging to build complex applications over distributed platforms such as grid computing model. This has led to increasing interest in modular interfaces based on services. The emergence of utility computing model enables provisioning of computing resources as utility services similar to electric power. Utility computing presumes that diverse computational resources can be aggregated on demand and the computation can be realized based on the demand and service load. Enterprises can

exploit this by concentrating more on their core business and outsource their computing infrastructure to specialist companies that offer utility computing services (Ross & Westerman, 2004). In this case, a service point of view abstracts an application's infrastructure level, allowing for more efficient use of grid resources and promotes utility computing, especially when redundant services can be leveraged to create fault tolerance.

Software development is aided by programming abstractions that consider software components to be potentially autonomous. Services provide programming abstractions that allow software engineers to design distinct software modules using better semantic interfaces. SOC as a computational paradigm provides a semantically rich and adaptable model for simplifying software development.

2.1.3 Major Advantages of Service Computing

The following are the major benefits of using standardized services (Y. Chen, 2018):

1. In a large-scale, open settings, services provide higher-level abstractions for organising applications. Even if these were unrelated to standards. they would be helpful in implementing and configuring software applications in a way that enhanced business productivity and improved the quality of the applications that were built.

2. The high-level abstractions are standardized. Standards allow software developed by various programmers to interoperate. Standards thus enhance productivity in the above-mentioned service use cases.

3. Standard enable the creation of general-purpose tools for managing the full system life-cycle including design development, debugging and monitoring etc. This is a major practical advantage, as it would be nearly impossible to design and field

robust systems in a timely manner without extensive tool assistance. Because tool manufacturers may validate their tools and so move part of the validation effort from the application programmer, such tools assure that the components generated are actually interoperable.

4. The standards feeds other standards. For example, the fundamental standards enables other standards such as dealing with bookings, processes and transactions.

2.1.4 Discussion

Service-Oriented Computing has emerged as a key computing paradigm that provides cross-organisational computational abstractions, technologies and architectures to facilitate and support enterprise services. It offers new way of developing, designing and delivering and consuming software applications. It enables the software-as-a-service concept to expand to capture the delivery of complex business processes, allow on-demand application development, and service reusability. Service Oriented Architecture (SOA) provides an architectural model that allows services to be published, discovered, and consumed by applications or other services, the goal of which is to realize loosely coupled, standard-based and platform-independent distributed computing. SOC relies on SOA to build integration-ready software applications, organize the applications and related infrastructures into a set of interacting services (Bouguettaya et al., 2017; Bouguettaya, Sheng & Daniel, 2014; Pautasso, 2014). Services that complies with SOA are used as fundamental elements to support rapid, low-cost development of distributed applications in heterogeneous environments. Most of the conventional business processes and services such as healthcare, finance, transactions,entertainment, tourism and education can now be offered as Web services. Organisations can now explore the availability of these services to achieve effective Business-to-Business collaboration and interoperation such that business data, functionalities can be shared.

The broad and continuous adoption of SOA and its related technologies by various business and software community continues to encourage paradigm transformation from component-based to service-based (N. Zhang et al., 2018). A large number of software systems have been developed by discovering and composing loosely-coupled Web services provided by different organisation. As the demand for service-based system continues to grow, finding appropriate component services to compose is a key step in developing a service-based system (He et al., 2017). Service discovery is a critical step of service-based system development which focus on finding a set of potential or candidate Web services for each functional requirement of a service-based system (N. Zhang et al., 2018). Over the years, the continual growth growth of Web service ecosystem, the diversity of Web services functionalities coupled with the growing complexity of Web service requirements have made Service discovery a more challenging task. This research work addresses the challenges related to the service discovery component of service computing.

2.2 Web Services: Concepts, Principles, Standards and Emerging Technologies

Web service technologies have evolved in a way that isn't completely dependent on the notion of service. The emergence of Web service technologies and standards in facilitating automated business integration have become a major driver of various advancements in integration software domain, most notably, the SOA implementation (M. P. Papazoglou et al., 2007). It has become the preferred implementation technology for achieving the SOA objectives including maximum service sharing, reusability and interoperability (Kreger, 2003). Over the past decade a lot of research efforts have centered on

improving various components of Web service and its related technologies. Standardization groups such as the World Wide Web Consortium (W3C) have led standardization efforts for deploying to allow smooth interoperations. Nowadays, The term *Web service* is frequently used nowadays in different contexts, thus the existing definitions of Web service ranges from the very broad and all-encompassing ones to the more specific and restrictive ones. In general, a Web service is considered any service accessible using the Web technologies (Sheng et al., 2014). This is an open and generic definition which implies that everything that has a Uniform Resource Locator (URL) is a Web service. According to M. P. Papazoglou, Web services are known to exhibit the following characteristics:

- Using standard internet languages and protocols, a web service exposes its functionality programmatically over the internet.
- Web service can be implemented through a self-describing interface using open internet standards such as XML interfaces which are published in a network-based registry.

Other technical definition of Web services includes:

- A more specific definition given by the World Wide Web Consortium (Haas & Brown, 2004), which defines Web services as a software systems identifiable by Universal Resource Identifier (URI) designed to support interoperable machine-to-machine interactions over a network, and which has an interface described in a machine-processable format,(specifically WSDL). Other systems (for example, systems like end-user software) interact with the Web service using Simple Object Access Protocol known as SOAP (Mitra, Lafon et al., 2003), which is typically conveyed using XML over HTTP and other related Web standards. The W3C definition is quiet encompassing as it explains how Web service should function.

It emphasizes that Web services should be capable of being *defined*, *described*, *invoked* and *discovered*, thus clarifying "accessibility" and substantiating the notion of "internet-oriented and standard-based interfaces". It also specifically states XML as part of the solution that should be the data format used in many Web-based interactions.

- Another formal definition of Web service given by IBM (Sheng et al., 2014) is that Web services are "a new breed of Web applications, and they are self-contained, self-describing, modular applications that can be published, located and invoked across the Web" . This definition emphasizes the need for openness, which in essence, means that a Web service has to be advertised and published such that it can be **discovered** and invoked over the internet.

- The *Dagstuhl* service computing working group (Ludwig & Petrie, 2006), defines Web service as *a possibly remote procedure with an invocation that is described in a standard (preferably XML-based) machine-readable syntax reachable through standard internet protocols with a description, including at a minimum the allowed I/O messages and a possible semantic annotation of the service functionality and data meaning*. Similar to W3C definition, this definition focus on the representation and integration of Web services via standard internet protocols and service technologies.

The following are the descriptions of various Web service standard protocols and technologies.

1. **Simple Object Access Protocol** (SOAP [1]) - is a messaging protocol of the XML specification designed for the decentralised, distributed environment, which exploit the capabilities of internet and XML to exchange of information between

[1] http://www.w3.org/TR/soap/

nodes in a decentralised, distributed environment via Hypertext Transfer Protocol (HTTP) and Remote Procedure Call (RPC). SOAP is inherently a *stateless* , one-way message exchange protocol between SOAP nodes, such as a sender and a receiver. SOAP can be employ to generate more complex interactions like request/response and request/multiple response, by combining one-way exchanges with characteristics provided by the underlying transport protocol and/or application specific information (Box et al., 2000; Suda, 2003). SOAP exhibits certain characteristics including being lightweight, platform and operating system independent, which are attributes inherited from the HTTP protocol and the XML. SOAP does not define any application semantics, such as a programming model or implementation-specific semantics; instead, it provides a simple methodology for expressing application semantics by providing a modular packaging model and data encoding mechanisms for data within modules. As a result, SOAP can be utilised in a wide range of systems, from messaging systems to RPC. SOAP as two related applications with respect to XML messaging: the *Remote Procedure Call* (RPC) and the *Electronic Document Interchange* (EDI). SOAP used for EDI are also referred to as the *document style* SOAP , the XML style for such SOAP could be a purchase order, tax refund, or any related document. When SOAP is used for RPC also known as *RPC-style* SOAP, then the XML will be a representation of parameter or return values (Snell, Tidwell & Kulchenko, 2001).

SOAP-based Web services depend on three important standardization initiatives: the Universal Description, Discovery and Integration (UDDI). In addition, SOAP Web service inherit some attributes from SOAP including protocol independent and statefulness. But they demand more computational resources, particularly when handling SOAP messages (Sheng et al., 2014).

2. **Representational State Transfer** (REST) (R. T. Fielding, 2000) - was original

introduced as an architectural style for building large-scale distributed hypermedia systems that expose data, resources, and functionality via Web services with URIs. The RESTful Web services emphasize the correct and complete use of the HTTP protocol to publish software systems on the Web (Richardson & Ruby, 2008). RESTful Web services interact by exchanging request and response messages, each of which include information that describes how the messages should be processed. Unlike SOAP-based Web services, RESTful Web services are lightweight and stateless, which make them suitable for ad-hoc integration over the Web (Sheng et al., 2014). The architectural style includes the design constraints which have been followed to define the HTTP protocol (R. Fielding et al., 1999) and the basic standards in combination with URI and HTML which has been a major component of the Web. The design constraints includes *global addressability* through resource identification, *uniform interface* shared by all resources, *stateless interactions* between component services, scalability of component interactions, self-describing messages and hypermedia as a mechanism for decentralized resource discovery by referral, use of intermediary components to reduce interaction latency, enforce security , and encapsulate legacy systems and reducing coupling between component (Pautasso, 2014). The constraints are further discussed below:

- ***Global addressability through resource identification :*** Resources published by Web service are to be given a unique identifier, which meaningful globally such that no central authority is involved in managing them, and they can be dereferenced independently of any context. REST has not made any assumption of what a resource could be or not be. Resource in REST case was intentionally kept very general and it can be used to publish some service capability and any source of machine-processable data, which may

include meta-data about the service.

- ***Uniform interface shared by all resources :*** All resources interact by using a uniform interface, which provides a small, generic and functionality sufficient set of methods to support all possible interactions between services. The methods have well-defined semantics which describe their effects on the state of the resources. In the context of the Web and its HTTP protocols, the uniform interface composed of the methods which are fixed set of operations including GET, POST, PUT and DELETE that can be applied to all Web resource identifiers . This operations can be extended if necessary (Dusseault & Snell, 2010; Goland, Whitehead, Faizi, Carter & Jensen, 1999).
- ***Stateless Interactions :*** Services do not create a permanent session between them that last more than one interaction. This ensures that requests to a resource are independent of each other. There is no shared state between clients and servers at the end of each interaction. Requests could lead to change of resource's state, where new state are made immediately visible to the associated clients.
- ***Self-describing Messages :*** Restful Web services interacts by exchanging request and response messages that contains the representations of resources and their metadata. These representations can be of various kinds, defined according to the client, context, interests and abilities. Similarly, Web browsers could request a representation of a Web page in a specific language, based on the users preferences. This improves the degree of intrinsic interoperability of RESTful architecture because a client can flexibly and dynamically negotiate the most appropriate media type or format with the resource as opposed to the enforcing the use of same format for the clients and resources.

- ***Hypermedia :*** this mechanism involves embedding references to associated resources within the resource representations. Therefore, client can locate the hyperlinks (identifier) of related resources during the representation processing, and decide to follow the links as they navigate the network built from the relationship that exist among the resources. Hypermedia aids in the discovery of decentralised resources and is also utilised for the dynamic discovery and description of service interaction protocols between services. In spite of hypermedia utility, it is also the constraints which has been used the least in most RESTful Web service APIs. As a result, Web service APIs that abide by this constraint are frequently referred to as "Hypermedia APIs."

The main design constraints of the REST architectural style can also be adopted incrementally, leading to the definition of a maturity model for RESTful Web services (Pautasso, 2014).

3. **Open Service Gateway Initiative** (Tavares & Valente, 2008): This is a framework that supports the implementation of component-based, service-oriented applications in Java. The framework manages the life-cycle of modules also refer to as *bundles* in OSGi [2], which includes adding, removing and replacing bundles at run-time, while preserving the relations and dependencies among them. It provides means to publish,deploy, manage and search for services. In addition, it supports the dynamic installation and un-installation of bundles.

4. **Web Services Description Language** WSDL (Christensen, Curbera, Meredith, Weerawarana et al., 2001) addresses the requirement for standardization of messaging format and communications protocol in Web community by defining an

[2]OSGi Alliance: http://www.osgi.org.

XML grammar which describes network services as collections of communication end-points or ports capable of exchanging messages. Specifically, WSDL describes the operations in a Web service, messages exchanged by the operations, the parts that make up the message, and the protocol bindings. The abstract definition of message and ports in WSDL is separated from their data format bindings or network deployment.

WSDL service definitions provide documentation for distributed systems and serve as a blueprint for automating the details involved in applications communication. (Christensen et al., 2001). A typical WSDL document utilizes the following attributes in the description of network services : *Types* – represents container for data type definitions using some type system like XML schema specification, *Message* – represents an abstract type definition of the communicated data. *Operation* – which is the abstract definition of an action supported by the network service, *Binding* – which is a concrete protocol and data format specification for a particular port type, *Port* – is an individual endpoint defined as a combination of a binding and network address, *Port-type* – an abstract set of operation supported by one or more endpoints , and *Service* – a collection of related endpoints,

5. **Universal Description Discovery and Integration** (Vaddi & Mohanty, 2019). UDDI is a universal business registry that serves as central repository in Service Oriented Architecture for indexing Web services, where service providers can register their service descriptions so that services can be located by developers and applications. UDDI provides a directory-based approach where service provider would publish their services following a specific format that is based on the directory. Consumers can then query the directory for the published services by using Application Programmable Interfaces. In essence, the UDDI

is an industry standard for service repository developed to houses the details of publishers, their published services and the descriptions and solve the service discovery/search challenges. The UDDI information model composed of four main data structures (Vaddi & Mohanty, 2019) which include *business-entity*, *business-service*, *binding-template* (which contains the technical information related to a single Web service for interacting and binding) , and *t-Model* (contains the technical specification for a Web service − points to the URL where the actual specification is available). UDDI gives access to Web Services Description Language (WSDL) documents and interprets SOAP request message.

6. **Electronic Business using eXtensible Markup Language** − ebXML [3] – is an initiative for a Business-to-Business XML framework that enables the collaboration of Web-based business services with set of specifications that allows electronic trading relationships between businesses. exchange. The heterogeneous nature of eBusiness transactions require a flexible infrastructure/framework that supports simple service calls and complex document. For eBusiness, key integration patterns realize SOA benefits in a pragmatic iterative manner.

2.3 Mashups, Composite and Atomic Web Services

Web services can function as either an elementary *atomic* service or as a composite Web services (Sheng, Benatallah, Dumas & Mak, 2002). An *atomic service*, is a single Web accessible application that does not explicitly rely on another service functionality to fulfill user's application requirements. Each of the atomic services offers programmatic interface as intermediate channel that allows two or more services to interact with each other. Appropriate adapters can be developed for older services written in languages such as CORBA so that they can be invoked as Web services. A *composite*

[3] http://www.ebxml.org/

service (Casati & Shan, 2001; Ngu, Carlson, Sheng & Paik, 2010; Medjahed, 2004) is a conglomeration of outsourced Web services, which brings together other composite and atomic services that collaborate to implement a set of operations with the aim of offering additional values. The services combined together by a composite service are referred to as its component or participant services. A common example of a composite service is a tourism service, which usually combines different but related services such as flight and hotel reservation, searching for attractions, food menus etc. A Web service is known to be defined by an Universal Resource Locator, a collection of attributes, and a set of operations, whether it is atomic or composite. The attribute of a service conveys information about the service to the service's potential consumer. Composite services offer several benefits, which include reduction in software application development time, thereby enabling rapid time-to-market of the application, and offering quick solution to ever-growing complexity of service consumer software requirements. Furthermore, composing Web services enables the reduction of skills and effort required for developing software.

A modern technique that allows service consumers to create value-added application based on existing application is refer to as *mashup*. The terms mashup and composite service are sometimes used interchangeably, particularly for composite services with components that have features such as *RESTful architecture*, *JSON data*, and/or *JavaScript interfaces*. A Web *mashup* could be refer to as an application that integrates multiple components services at any application layers which include data, application logic and presentation layers (Daniel & Matera, 2014a). Mashups come in different forms that expresses one or more layers of the stack at which the mashup components are integrated. The following types of mashups are defined based on the application layers (Daniel & Matera, 2014b, 2014a):

1. *Data mashups*: Data mashups occur at the data layer. They collect data from

various data services or resources, process it, and then return an integrated result set (the data mashup's output). Typically, data mashups are published as Web-accessible resources, such as RSS feeds or RESTful Web services.

2. *Logic mashups*: For logic mashups, components integration occur at the application logic layer , which enable composition of functionality in any of the logic or data component type. Logic mashups define processes that orchestrate components and that are gain as logic components.

3. *User Interface (UI) mashups :* These are found at the presentation layer. They modify user interface components and merge the native UIs of the components into an integrated UI, in which the desperate UIs may be synchronised. Pure UI mashups necessitate the arrangement of UI elements in a shared Web page. Users can interact with UI mashups because they are usually published as Web application.

4. *Hybrid mashups*: Hybrid mashups are made up of several levels in the application stack. They combine a variety of components into a single mashup, with integration occurring at multiple levels. The advantage here, is that the mashups are most feature-rich and complete, allowing full-fledged applications to be developed. Hybrid mashups can be interactive Web services, depending on the layers where integration takes place. Because each layer often addresses concerns that are significantly distinct from the others, the key challenge of hybrid mashups is mediation between the three layers, or cross-layer integration.

Web Services are ubiquitous in today's Business-to-Business interactions, and they are critical to the social and economic developments of both macro and micro businesses. The rapid development of Web service technologies and related standards have help propagate dynamic and automated business integration. Web services can be published,

discovered and invoked over the web using various standards such as XML, WSDL, UDDI, HTTP, SOAP and REST (Jalal, Yadav & Negi, 2019). There has been a paradigm shift in software development from component-based system to a service-based due to various advances in SOA and its related technology.The service-based paradigm not only reduces the cost, time, and effort necessary for software development, but also improves the system's reusability, agility, and quality (N. Zhang et al., 2018; Bano, Zowghi, Ikram & Niazi, 2013). The so called 'Big' Web service technology stack (Pautasso, 2014) provides interoperability for both the Remote Procedure Call and messaging integration styles. Eventhough different styles are able to integrate enterprise applications, recent technology trend in Web services domain infer that RESTful Web services are capable of eliminating the presumed complexity associated with 'Big' Web service technology, especially SOAP (Pautasso et al., 2008, 2008). Since the framework presented in these research is evaluated using RESTful Web services, specific features of these services highlighted in this section to distinguish them from the traditional SOAP-based Web services. It is worth nothing that this research exploit generic attributes of Web services such as service descriptions to facilitate the service discovery processes.

2.4 Web Service Discovery and Its Related Concepts

The rapid increase in the number of Web service on the internet is being propelled by the adoption and continual growth of Web service technologies. The last decade has witnessed an exponential increase in number of service providers and companies that offer their capability as Web services. These services are advertised and published via public Web service registries. The number of registered Web services in the public registries such as ProgrammableWeb as of January 2021 was more than 24,000. Yearly, over 2000 new Web services are added to ProgrammableWeb repository alone (Zarei & Gaedke, 2020). Organisations are exploring these services to develop new value-added

business applications using Web service composition techniques. Discovering and assembling loosely-coupled Web services supplied by many organisations has resulted in the development of a huge number of software systems (He et al., 2017). The primary requirement for developing such value-added application is discovering the most relevant Web services capable of realising the functionalities required to complete the new composite service. Finding and selecting the most suitable Web service from a pool of diverse, potential candidate services that can satisfy the requirements of service requestor are still challenging tasks (Jalal et al., 2019; N. Zhang et al., 2018). This section presents Web service discovery and various concept and research trends related to service discovery. An overview of existing approaches of Web service discovery are also presented in this section.

2.4.1 Definition

F. Chen, Li, Wu and Xie defined Web service discovery as the activities related to identifying services matching user requirements and selection as identifying the best choices among the matching services. In this definition, discovery process mainly involves process of searching that is expected return a list of candidate services. Other authors refers to Web service discovery as the process that enables service requestor or user to find existing Web services that meets their functional requirements based on the requester's functional and non-functional requirements presented inform of a queries (He et al., 2017; Stein, Barchewitz & El Kharbili, 2008; Zarei & Gaedke, 2020; N. Zhang et al., 2018). This processes returns the Web services descriptions previously published by service providers in service registries (Obidallah, Raahemi & Ruhi, 2020). Exploring this the discovery service, a service consumer identifies potential services or service providers whose offerings meets and its functional requirements.

2.4.2 Key Information Attributes for Service Discovery Processing

Web service discovery automation can be achieved by incorporating side information such as semantics, QoS, and social information into a Web service functional description, and using some service matching and selection processes. The following are Web service attributes can be explored in service discovery processing (Kritikos & Plexousakis, 2009; W. Chen et al., 2015a; Hoschek, 2002).

Service Functional and Non-functional Attributes

Web service are usually described in terms of a their functional characteristics, which include operational characteristics that define the overall behaviour of these services. (M. P. Papazoglou & Dubray, 2004). Each Web service is published with its functional description written by the service provider. When a developer builds a composite service, a functional description is also created accordingly along with the service. Most descriptions are purely syntactic, and as a result, service discovery and composition approaches that solely rely on the textual, functional descriptions involve predominantly manual surfing. Non-functional properties include non-functional service quality attributes such as the performance metric including response time, cost, accuracy, integrity, availability , security attribute, scalability and reliability.

Service Semantics

Semantic descriptions of Web services are key ingredient that can be exploited to realise services potentials like dynamic discovery and enabling automated service composition. Semantics annotations could be injected into service discovery processing to enrich the attributes required for high quality discovery solution (W. Chen et al., 2015a). Semantic information such as OWL ontologies are often used to annotate service interfaces. Existing semantic Web Service discovery solutions have demonstrated the advantages

of semantic annotations in discovering services, particularly in terms of accuracy and when dealing with heterogeneous data models (Kanagasabai et al., 2013; Pilioura & Tsalgatidou, 2009). One of the main challenge in exploring Ontological-based Web service semantics is building ontologies. Building an ontological for semantic annotation is a time consuming and complicated task that requires substantial number of knowledge experts to cooperate with each other, resulting in a shortage of consensus and ubiquitous ontologies (W. Chen et al., 2015a). Another challenging issue of integrating semantic annotation to service discovery is the lack of ubiquitous ontologies (N. Zhang et al., 2018).

Service Sociability

Service sociability is service property that characterized the ability Web service to interacting effectively with other related services. This property is usually captured or supported by the network models commonly referred to as Web Service Social Networks (WSSN) (Maamar, Hacid & Huhns, 2011). By connecting Web services into a global service social network, various social attributes such as user/service interactions, collaboration, competition, substitution and association attributes can be easily captured and further explored to enhance service discoverability. However, one of the key research challenge here is how best to capture service sociability, or model service interaction using social network models such that other service properties are well preserved , and that both historical and future interactions can be correctly modelled (W. Chen et al., 2015a). Most Web service discovery solutions usually ignore the influence of Web service interactions in the discovery process.

2.4.3 Quality of Services

Quality of Service (QoS) is a very important attribute of Web service that can be exploited to improve various service solutions including Web service discovery. In service context, It describes the capabilities of a Web service to satisfy the implied needs or stated requirements of service consumers. QoS serves as the benchmark to uniquely distinguish the services and service provider (Raj & Sasipraba, 2010). In addition, QoS can be used in service computing domain to validate web service ranking and describe operational metrics of Web service (Tran & Tsuji, 2008). As there can be multiple Web services available for a particular requirement at a time, QoS provides a realistic way of distinguishing, ranking and selecting most suitable services especially when they share similar semantics and functional attributes (Lin, Lai, Wu & Lo, 2014). QoS is usually used employed to describe some of the non-functional attributes of Webs services mentioned in the previous subsection. Some QoS metrics used for selecting most appropriate services in a pool of candidate services includes; *availability*, which is the probability that system is up and can respond to consumer requests. *reliability* is the capability of Web service to perform its required functions under stated condition over a specific period of time, *cost* is the measure of the cost of requesting or consuming a service, and *performance* is the measure of the speed to complete a service request. It is measured by latency, throughput and response time (Patil & Gopal, 2010).

2.4.4 Requirements and Processing Steps for Service Discovery

In this section, generic steps required for successful service discovery are discussed. The steps are generally classified into *pre-matchmaking* and *matchmaking/selection* steps (Kritikos, 2008; Hoschek, 2002). The pre-matchmaking step is the step that involves collection and pre-processing of Web service data including their descriptions, presentation and publication of Web services. The matchmaking step is the step where

the requests of service requester are matched against the available services. The steps are discussed as follows:

- *Web service description*: Service provider are required to follow a specified format by the service registries for describing the functionalities of their services,while service requester can describe their service needs in form of query using natural language.

- *Web service Publication*: This step involves making the services,and other service-related objects available and reachable to potential service consumers through the storage of these object descriptions in one or more well known service registries.

- *Scalability* is a critical issue that affects the advertisement or exposure of large number of Web services in the service repository as well as the number of user interacting with the repository for service request. The scalability of the service system and that of the system's subsystems and its sub-components must be addressed in the discovery system design process as each of these entities could impact the effectiveness of the discovery service. The scalability of matchmaking engine of the discovery system is the most critical scalability factor. Hence, the discovery process must account for large-scale scenarios, such as the web size, and must not rely on a centralised broker, but rather enable peer-to-peer operations.

2.5 Existing Web Service Discovery Approaches

Web service discovery is a core component of Service-Oriented Architecture that facilitates advertisement or exposure of services by providers for consumption, and finding the services by consumers. The continual increase in number of web services published

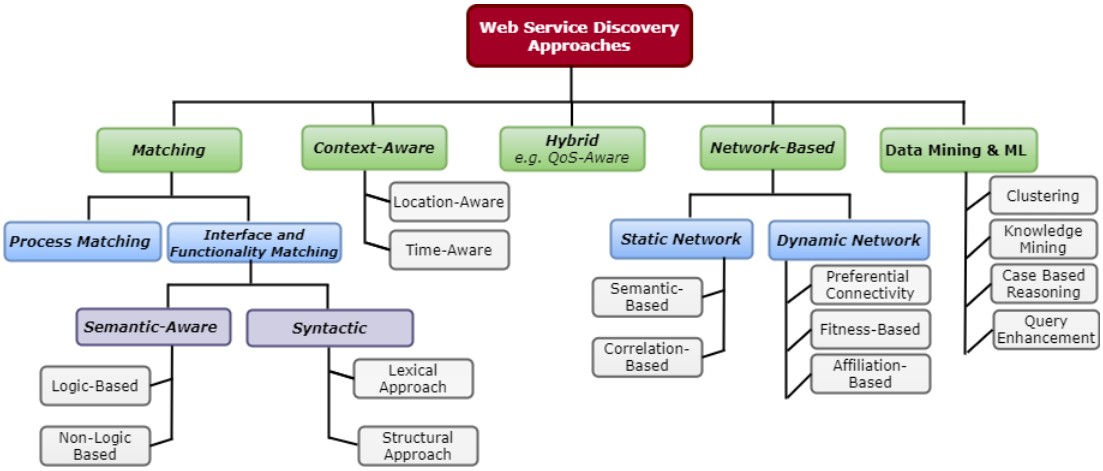


Figure 2.3: The Taxonomy of Web Service Discovery Approaches

on the internet that have never been used, coupled with the increasing complexity of service consumers' demands make the task of service discovery to continue to be very challenging. The success of advertised or published Web services depend on how well the service can be discovered and consumed. Over the years, different approaches have been proposed and explored by researchers to solve discovery problem in SOC, yet, there are still gaps to fill in terms of efficiency, accuracy, diversity , scalability and user-experience. This section provides overview of different approaches for Web service discovery under five different groups according to the main paradigm used in the discovery process. The groups include: the *matching group* , the *Context-Aware* group, the *QoS-based*, the *Data Mining and Machine Learning* group and *Social Network-based*. Each of the approaches discussed in the groups are clearly differentiated from one another with emphasis on their pros and cons. Related works and the state-of-the-art of the approaches are covered.

2.5.1 Matching Group

The matching group includes the interface matching, functionality matching and process matching. The interface of a Web service has various matching elements such as service

description, service operations(operation names and number), input and output schema, data-types. These elements are used for matchmaking in this group (Vaddi & Mohanty, 2019). This group of service discovery techniques explore service descriptions including semantic Web service description and non-semantic Web service descriptions.The semantic web services use ontologies to describe the services. Description language such as $OWL-S$, $DAML-S$ are used to to describe Semantic Web services, while description languages such as WSDL2.0 or WADL (for RESTful service), WSDL (for SOAP-based services) and plain, unstructured HTML are used for non-semantic Web services (Christensen et al., 2001; Hadley, 2006). For Semantic Webs, both $OWL-S$, $DAML-S$ description languages include service sub-classes namely *serviceProfile*, *serviceModel* and *serviceGrounding* (D.-N. Le, Nguyen & Goh, 2009; Kanagasabai et al., 2013). Out of these sub-classes, only the *serviceProfile* is usually used during Web service matching process. The serviceProfile contains information such as the name of service, service functional description, and additional attributes. The functional description is the most important information in the serviceProfile that is used for matching. During the discovery process, the descriptions provided by the service provider is matched against the service requester's request to determine if the two shared some similarity especially functional similarity. Web services that do not use semantic descriptions are usually discovered on a syntactic level using *Information Retrieval* (IR) techniques. Process matching was introduced in order to improve the precision of Web service discovery and support functionality matching (F. Chen, Li et al., 2017). Generally, this involves presenting process model of services in such a way that the similarity of business process descriptions can be measured at either individual component level or composite level. This group of Web service discovery approach is further classified into subgroups : *Syntactic-based* and *Semantic-aware* Web service discovery approaches. Each subcategory is further divided into fine-grain groups with relevant works discussed under each group. The following sub-sections presents the

elements in this group:

Syntactic-Based Web Service Discovery Approaches

These approaches search and find Web services by matching the *Keywords* from the Web services with the *Keywords* in the service requester's query using Information Retrieval (IR) techniques such Term Frequency-Inverse Document-Frequency (TFIDF) (Manning et al., 2008). A typical technique commonly used for determining the importance of a word (or term) in a document is the TF-IDF technique. The TF-IDF is a numerical statistic that considers how many times a term appears in a document as well as how prevalent the term is across the corpus to determine how relevant a term is (Lizarralde et al., 2019). The Web services registry would return some good matches that are closely relevant to service requester's query , if the the query contains multiple, topic/function-specific keywords that approximate the required Web service.

Several works have explored syntactic-based approach for developing Web service discovery. Some research works reduced the problem of Web service discovery to a well-documented problem of finding similar service description documents for a particular service (F. Chen, Li et al., 2017). In this case, the WSDL description documents are processed as documents and existing information retrieval techniques are adapted to for discovery process. Syntactic Web service discovery approaches can be further classified into two subgroups as shown in Figure 2.3: *Lexical discovery approaches* and *Structural discovery approaches*. The two subgroups and the related works are discussed as follows:

1. ***Lexical discovery approaches* :** These approaches exploited the use of natural language descriptions for describing Web services operation names, which usually contains the services functionality descriptions (Stein et al., 2008). A typical lexical algorithm in this case, performs some pre-processing on the descriptions

like removing stop-words, tokenization, stemming and finding words with same meaning (synonyms) using some lexical databases such as Word-Net (Miller, 1998) and so on. After this, the matching, retrieval and ranking processes are then carried out using common IR techniques. Most traditional, keyword-based service discovery approaches like UDDI falls in this category. A related approach to this group was introduced in (C. Wu, 2012), where statistical methods were adopted for WSDL document-term tokenization. These methods exploited the minimum description length principle (MDL), transitional probability(TP) and prediction by partial matching (PPM) for the retrieval process. Another related work was introduced in (Kokash, 2006), where the authors employed syntactic similarity matching technique to search for Web services using the WSDL documents. The authors first converted both the Web service request and service documents into vector representations using VSM, and then computed the similarity between the two vectors to retrieve closely related services to the request. Similarly, Platzer and Dustdar (2005) introduced a distributed scheme which integrates VSM with cosine similarity to discover and retrieve Web services from UDDI registries.

2. ***Structural discovery approaches*** **:** The structural discovery approaches, exploit the syntactic information available in the service profile such as the interface description, the definition of the data message traded between communicating partner or adapt TFIDF to find the most similar Web service to the service requester's query. In this case, the matching process requires the service requester to specify the structural requirements such as data type. These method do not expose the semantic relationships between web services (Fallatah et al., 2014). A syntactic Web service discovery approach that exploit structural information was introduced in Elshater et al. (2015) work. The work used statistical modelling and indexing techniques. The authors integrated TFIDF with VSM and implement

both as a K-dimensional tree structure. They generated TFIDF model for service corpus and then built a K-dimensional tree-index to search the TFIDF model. The user query is also transformed into vector using the TFIDF model. Service retrieval process then follows by navigating the R-D tree. Hao et al. (2010) employed similar IR technique to rank Web services for a required functional description. The authors considered the WSDL documents in XML format as a tree-structure, and developed schema tree-algorithm for ranking the Web services for matchmaking process. Authors in Dong et al. (2004) and (Cong et al., 2015) employed clustering algorithms. Particularly, in (Cong et al., 2015), the authors applied hierarchical clustering algorithm to organize service repository into binary tree-like clusters using distance metric, and then performed discovery process on the binary tree (instead of linear tree) to reduce time-complexity.

In general, the syntactic nature of these approaches (especially the ones that utilize WSDL documents) and the heterogeneity of the Web services may result to the discovery of mostly irrelevant results than the relevant ones (Lizarralde et al., 2019). Eventhough syntactic approaches can leverage IR techniques and incorporate several enhancement methods, they still suffer from low recall and precision (N. Zhang et al., 2018). Because this group of discovery approach rely mainly on the Web service descriptions, they also suffer from any set back related to quality of service description. For example, Zhong et al. (2018) shows that even though composite services are made up of some atomic functionalities from participating services, and that all terms describing each participating service in the composition are very important in describing the services derived from them, composite services descriptions usually do not include or capture these terms. The descriptions given to services by the providers are usually subjective, and may not be comprehensive enough to describe the service. The authors show the negative impact of this shortcoming on the performance of syntactic-based service

discovery approaches. Another challenge of syntactic-based approach is *Vocabulary problem* which includes problem related to *Polysemy* and *Synonyms* (Zhong et al., 2016; Lizarralde, Rodriguez, Mateos & Zunino, 2017; Lizarralde et al., 2019). Service requesters often introduce ambiguous and incomplete natural language sentences in their requests. This also makes it challenging to retrieve relevant services because only the terms captured in the service requester's query would be consider in the discovery process (Lizarralde et al., 2019). Another issue with this group of approaches (especially the ones that rely on markup-based Web Service descriptions) is bad specification practice which makes service discovery through this approach challenging (Palma et al., 2015; Rodriguez et al., 2010).

Semantic-Aware Web Service Discovery Approaches

Using naive text search on service documentations for service discovery have proven to be not so effective because natural language text is usually informal and service textual descriptions provided by providers maybe subjective,as a result, service documentations maybe ambiguous or lack some key terms that better describe the service functions (Kanagasabai et al., 2013; F. Chen, Li et al., 2017). In addition, the increasing diversity of Web services with so many services offering similar functionalities with diverse properties makes it more challenging to use mere keyword for discovery process. Semantic-aware Web service discovery approaches attempt to address the drawback of syntactic approaches by searching for semantically similar Web services of queries (N. Zhang et al., 2018). The synergies between semantic technologies and service discovery make it easier to create rich and formal representations of services and agent interactions as well as to specialise and generalise service requirements. These representations of services and the interactions are needed for capturing the semantics of queries and service descriptions from both the service consumers and providers respectively. The use of an explicit representation allows for principled service selection

and request fulfilment, as well as dynamic engagement and negotiation with service providers (Kanagasabai et al., 2013). Various semantic technologies help facilitate the specialization and generalization of service needs.

A generic framework for semantic service discovery system was introduced in (Kanagasabai et al., 2013). Typical semantic service discovery system has two key active players; the service provider and the service requester. The service providers advertise Web services through the system, while the service requesters search for services that matches some specified requirements. The service repository houses the advertised services after suitable mediation or alignment to resolve the heterogeneity (a result of differences in service platforms, data formats and ontologies). When a service requester initiates a service request, the request is matched against the services in the repository using the matchmaking engine, the matched results are then retrieved. After then, service negotiation is used to communicate with the providers and, if necessary, get dynamic information. Lastly, the selected service (usually done based on requester's preferences) is consumed or invoked compositions. The formalism used for semantic service descriptions is the framework's main building block. The main building block of a typical semantic-aware service discovery framework is the formalism used for its semantic service descriptions (Kanagasabai et al., 2013). Some of the commonly used formalisms are described below:

- ***OWL-S ;Formalism of Web Service Semantic Descriptions*** : OWL-S is an ontology language for describing Web services. It follows the W3C standard for Web ontology language and comprises of three main components: (i) The service profile component of the Web service ontology language, which is used for advertising and finding service functionalities. The OWL-S service profile provides information required for a service consumer to discover a service (ii) The process model, which provides the descriptions of service operation. (iii) The service

grounding specifies how services can interoperate through messages.

- ***Web Service Modelling Ontology***(WSMO) (De Bruijn et al., 2005): WSMO provides a comprehensive conceptual framework and a formal language for semantically describing all elements of Web services in order to automate the discovery, composition, and invocation of services over the Internet.

- ***Semantic Annotation for WSDL and XML Schema*** (SAWSDL [4]): SAWSDL defines how to add semantic annotations to various parts of a WSDL document such as input and output message structures, interfaces and operations. It enables Web service semantic annotation by leveraging and building on the existing extensible framework of Web service description language WSDL. It tries to explore WSDL while being agnostic to the semantic representation language. WSDL-S (Akkiraju et al., 2005) is another formalism for semantic annotation that is similar to SAWSDL as it allows addition of semantic annotations to WSDL.

- ***Semantic Annotation of Web Resources***(SA-REST) (Sheth, Gomadam & Lathem, 2007): Adding semantics to RESTful services is more challenging than adding semantics to WSDL. Unlike WSDL, REST-based services are often embedded in Web pages written largely in XHTML, on the other hand, WSDL was specifically created to capture service descriptions and has a supporting schema for doing so, XHTML is a more general-purpose language that adds semantic annotations only to those page elements that wrap a service or a service description. SA-REST adopted many concepts that were first introduced in WSDL-S [5] and then adapted in SAWSDL — particularly, the model reference attribute that connects service element to the ontological concepts that describe it.

[4] https://www.w3.org/TR/sawsdl/
[5] www. w3.org/Submission/WSDL-S/

In general, the requirements and procedure for automating service discovery processes can be enhanced if machine processable descriptions are incorporated in to the discovery processes. These descriptions can be further explored to verify whether they matched against each request with respect to the services. In addition, new knowledge can be further deduce from existing facts which includes domain ontologies and domain background knowledge, and this knowledge can further be incorporated into discovery process to to enhance service discovery. Hence, the identification of the reasoning provided via this means is critical for enabling discovery process. Generally, the key tasks carried out by a typical semantic-aware service discovery framework include the following:

- **Service Advertisement:** Web services developed by the service providers are advertised on service repository so that the service can be exposed and consumed by users including businesses . To facilitate their invocation, services must be described by a description language such as OWL-S or WSMO. Similarly to what is done in (Küster, König-Ries, Klein & Stern, 2007), provider can also develop their own semantic description languages.

- **Service Meditation** In order to enable interoperability between services that are described using different languages and avoid heterogeneity problem, semantic discovery framework needs to make provision for mediation between the languages or align the descriptions to a standard language.

- **Service Storage** The service repository serves as the storage for Service descriptions. The semantic service repository comprises of domain and service ontologies. The domain ontologies capture the names and terminologies used by the providers and and service users to described the services or service queries, while the service ontologies store information about the Web services. An effective indexing technique is required to facilitate querying the repository. Different

indexing and storing technique may be used for different languages depending on the application requirements and in cases where the description language required specific indexing and storing technique.

- **Service Request** Service consumers create service requests that describe what they want to achieve, what functionality they want to include in their application, and send these to the service repository as queries. In cases where there are semantic differences between the request and advertised services, especially when different description language is used, the discovery framework must be flexible enough to address such heterogeneous environment.

- **Service Matchmaking** This involves the process of matching a requested service against the advertised services. The matchmaking results are usually returned in form of a list or ranked list of advertised services with the coefficient or matching degree that indicate the suitability of the matches. Service matching is one of the most critical aspect of Web service discovery.

- **Service Negotiation** Because different services may offer different quality of service attributes, effective negotiation procedure is required especially in cases where there are trade-offs between quality and cost, or where the invocation of a service determines the quality of another service offerings.

- **Service Selection** This step involves deciding which service to select for the desire service functionality out of the resulting candidate services from the matching stage. It explores result in matching stage.

A number of research work in service discovery have explored the semantic discovery approach. Existing works in this area can be classified into two subcategories based on the nature of reasoning algorithm/matching technique used (N. Zhang et al., 2018; Klusch, Kapahnke, Schulte, Lecue & Bernstein, 2016): the *Logic-based*

semantic-aware Web service discovery approaches and the *Non-logic-based* approaches. The subcategories and related works are further discussed as follows:

1. **Logic-Based Semantic-Aware Approaches** This group of works describe Web services using ontology-based semantic service description languages such as SAWSDL, OWL-S and WSMO and design *logic-based* reasoning algorithms for matchmaking/service retrieval. Works under this subcategories can considered separately based on the description language used as follows:

    - *OWL-S-based Approaches :* In general, this group of works performs logical reasoning on semantic service descriptions. They employ the service profiles and the domain ontologies to facilitate matching process between a service request and a published service. The particular approach used depends on the element of the description, that is, if the services are described by their inputs, outputs, preconditions and effects (IOPEs) and the non-functional properties considered (Kanagasabai et al., 2013). The logical and full functional IOPEs profile matching combines the scores of logical signatures(IO) and specification (PE) matching (Klusch et al., 2016). Notable related works include (Klusch, Fries & Sycara, 2006), (Klusch & Kapahnke, 2012), (L. Li & Horrocks, 2004), (Paolucci, Kawamura, Payne & Sycara, 2002), (Kiefer & Bernstein, 2008), (Thiagarajan, Mayer & Stumptner, 2009), (C. Zhang, Zhu, Zhang & Yang, 2007), (Rodriguez-Mier et al., 2015), (F. Chen, Lu et al., 2017), (F. Chen, Li et al., 2017). For example, in (Klusch & Kapahnke, 2012), the authors introduced a logic-based matching approach for semantic service discovery approach. The authors used semantic service matchmaker $iSeM$ to perform an adaptive semantic selection of OWL-S services. The semantic matching process leverages the computation of strict and approximated logical Input and Output concept subsumption relations, and the

logical specification plugin relation. Authors in (Klusch et al., 2006) proposed hybrid service matchmaker called OWLS-MX for semantic service matching. The approach exploits logic-based semantic matching with token-based similarity metrics (the X in OWLS-MX denotes different instances of matching schemes-$M0$ – $M4$ which depends on the syntactic similarity metric used). Thiagarajan et al. (2009), proposed an OWL-S-based semantic-aware discovery approach that utilizes a two-staged consistency-based matchmaking approach. The first stage involves shortlisting service profiles that potentially match the service request. In the second stage, the concrete information about these services is gathered by directly querying the services to ensure that they meet the requirements (checking for consistency with the requirements). C. Zhang et al. (2007) introduced an approach that used clustering technique utilized OWL-S functional descriptions at the peer nodes to facilitate service cluster creation. First, the published services in the service repository are clustered into communities based on their non-functional properties. Then, query packets move around these communities to make rapid recognition. The query packet is then sent to the most closely matched community. The non-functional features of each service in this community are then examined. Finally, the most appropriate services are selected. The work by Paolucci et al. (2002) is based on matching IO descriptions of services, while the work by (L. Li & Horrocks, 2004) proposed a service matchmaker that uses DAML-S ontology based and a description logic reasoner to match ontology-based descriptions.

- *WSMO-based Approaches :* Web Service Modelling Ontology-based approaches considers service discovery as a task of fulfilling goal that abstracts user's requirements (Kanagasabai et al., 2013). Unlike OWL-S, WSMO

is able to model mediation for addressing heterogeneity which usually arise in an open environment. Notable works that explore this method include (Keller, Lara, Lausen, Polleres & Fensel, 2005), (Klusch, Fries & Sycara, 2009), (Roman et al., 2015). Authors in Klusch et al. (2009) introduced an hybrid matchmaker WSMO-MX for WSML services converted to WSML-MX. In general, WSMO-MX delivers a ranked set of services that are semantically relevant to a given query by iteratively computing logic-based and syntactic similarity-based matching degrees. Ontology-based type matching, logical constraint matching, and syntactic matching are all performed by the matching filters. Keller et al. (2005) introduced a generic discovery model that exploits features of WSMO including its capability to model mediation for handling heterogeneity in open environments. The discovery model enables efficient pre-filtering of appropriate services and accurate contracting of services that fulfil a given requester goal.

- *SAWSDL and WSDL-Based Approaches :* Here, related works that utilize Semantic Annotation and the formalised semantic Service document language to facilitate logical matching process are considered. Notably of these works include: (Kourtesis & Paraskakis, 2008), (Hobold & Siqueira, 2012), (Stavropoulos, Andreadis, Bassiliades, Vrakas & Vlahavas, 2015) (Wei, Wang, Wang & Bernstein, 2011), (Verma et al., n.d.). For instance, Kourtesis and Paraskakis (2008) introduced a SAWSDL-based semantic discovery approach that utilizes SAWSDL for annotating Web service interfaces, and OWL for modelling Web service functionalities, and employs description logic reasoner for matchmaking process. Hobold and Siqueira (2012) proposed an approach that operates in two main stages: In the first stage, the approach tries to find a single published Web service that meets

the user's requirement, if no service is discovered, a composition of single Web services would be returned. The composition services are retrieved by a technique that creates a graph of semantically matched services, based on the information annotated on service descriptions using the SAWSDL (Semantic Annotations for WSDL) standard. Verma et al. (n.d.) introduced a WSDL-S-based Web service discovery approach, applied over federated registries using METEOR-S.

Even though, it has been shown that logic-based approaches can achieve substantial performance because of the accurate descriptions of Web services and queries. However, they are challenging to implement because establishing and managing ontologies, as well as manually annotating services and queries using semantic description languages, take a lot of time (Crasso, Zunino & Campo, 2011; N. Zhang et al., 2018).

2. **Non-Logic-Based Semantic-Aware Approaches** : These are recent non-logic approaches design with the aim to reduce the complexity of semantic matchmaking, some notable works include: (J. Wang et al., 2017), (Aznag et al., 2014), (Cassar et al., 2013), (Naim et al., 2016), (Z. Li, He, Wang & Zhang, 2014). In general, these approaches tend to reduce the complexities associated with logic-based semantic-aware approaches by analysing the frequency of occurrence of some key terms in Web service descriptions and determining the semantics which are implicit in the descriptions(Cassar et al., 2013). These approaches rely on techniques such as topic modelling, graph matching, natural language processing, linguistic analysis, data mining, and information retrieval techniques (Mohebbi, Ibrahim, Khezrian, Munusamy & Tabatabaei, 2010). For example, authors in Cassar et al. (2013) employed *Probabilistic Latent Semantic Analysis* (PLSA) and *Latent Dirichlet Allocation* (LDA) for extracting latent topics from Web

service descriptions and introduced a service matchmaker based on the topic distributions of the Web services. In (J. Wang et al., 2017), authors proposed semantic discovery approach the used Biterm Topic Modeling (BTM) to learn latent topics from services and proposed a concept of "common topic group" (CTG) for organising the services that shared multiple topics. The discovery system was developed using CTG model. One major challenge of these approaches is that the performance in terms of their accuracy is usually limited by the coarseness of the learned latent topics (Cassar et al., 2013). Other non-logic-based approaches including (F. Chen, Lu et al., 2017), (Lu et al., 2016) explore the relationship between words in services descriptions and the hierarchy of these words.However, common issues with this approach include unexpected semantic relationships among descriptive words which in turn may affects the performance of these approaches, and vocabulary gaps.

In summary, existing approaches including syntactic approaches and semantics-aware approaches, either logic-based and non-logic-based, have various drawbacks and are very challenging to achieve quality, highly effective and practical service discovery system (N. Zhang et al., 2018). In addition, it is difficult for service consumers to compose high-quality service queries, which may result to retrieval of services that do not satisfy user request. Nevertheless, most of these approaches do not consider the sociability of Web services or the continuous interactions among these services, which are vital to how they are consumed overtime. All the approaches considered Web services as an isolated functional island and ignore connections among Web services.

2.5.2 Context-Aware Web Service Discovery Approaches

In recent years, context-awareness has become an important ingredient of discovery and recommender systems. Generally, context refers to any information that can

be used to characterized the situation of an entity such as person, place or object. According to Rong and Liu (2010), *context* in Web service discovery is considered as any information that explicitly and implicitly affects the user's Web service request generation. Explicit context is directly provided by the user during the matchmaking process, while the implicit contexts are collected in an automatic or semiautomatic manner. implicit contexts is more applied in web service discovery as the user is not directly involved. A lot of systems rely on implicit context to study and analyse the user's potential need. For instance, a location-based and time-based Web service system, will consider the time and location where the user submits the service request, but the user might be aware of this process. The ability of the matchmaking component to integrate context information is fundamental to the development of an adaptable and personalized service discovery system. Such system can optimize discovery results, user's request, and perform personalization using the context information. Hence, context-awareness in discovery system can enables customization of services in accordance to the immediate condition/situation without human intervention (J. Li, Zaman & Li, 2015). In order to better satisfy service consumers' requests in different context, the discovery system need to understand the context in which the requests are made, especially, during when performing the matchmaking process. Using this approach, when a user submitted a request for Web service, the context is investigated and suitable services relevant to the context are returned.

Context Awareness method can generally be divided into four categories based on how the context information is collected (Rong & Liu, 2010): (i) *personal profile oriented context*, (ii) *user history oriented context*, and (iii) *process oriented context* and other context:

- *Personal profile oriented context:* This type of context mainly captures user's personal profiles, preferences, taste, and other related information that can be

used for personalization. Personal information such as location, time, and user's situation are used as contexts decomposing the discovery goal, setting selection criteria and supplying parameters. The main drawback of this approach is that it makes system architecture more complicated when new features and constraints are introduced (Mukhopadhyay & Chougule, 2012). Authors in Balke and Wagner (2003) show how to create personal profile and indicate it is important in decomposing the discovery goal, setting selection criteria and supplying parameters, thereby achieving personalization in Web service discovery. Nazir et al. (2008) introduced a similar approach to capture user's profile and then apply it to facilitate personalization in Web service discovery. Some research works (Sheng, Benatallah, Maamar, Dumas & Ngu, 2004; Maamar, Mostefaoui & Mahmoud, 2005), introduced a method for building service user's personal profile by modelling different context attributes such as location, time, and user's situation (Medjahed & Atif, 2007; Kuck & Gnasa, 2007). Eventhough these works show promising results, they suffer from some drawbacks related to the core methods used. For instance, the system architecture of would be too complex and complicated when new constraints or attributes are included. In addition, some application domain have their own constraints, which make these approaches less adaptive to such application domain (Rong & Liu, 2010).

- *Usage history oriented context:* Since user's data consumption pattern can be captured and employed for predicting the user's behaviour, context can be constructed from user's usage history. This the fundamental idea behind the usage history oriented context. The assumption of this sort context is that the Web service request of the user over a period of time becomes similar to a degree, hence, through the collection of historical usage pattern data from a long term observation, it practicable to discover some important information from the user's

past experience. The usage history oriented context related works in Web service discovery can be further subdivided into two including *personal usage history* and *group usage history* oriented contexts (Rong & Liu, 2010). For *personal usage history* oriented context based works, they emphasis the usage of previous user's personal experience information in Web service domain to indicate user's service preference and domain background. The aggregated personal usage information can theoretically be utilised to speed up and personalise the matching process. Authors in (Kokash, Birukou & D'Andrea, 2007) proposed a usage pattern model that captures user's historical usage of Web services by storing the log information of for user-system. By using this log information, the model can use this context to facilitate service selection process. The major drawback of this approach is that over reliance to user past usage information could amplify information distortion as user's historical service usage may not necessarily represent their real behaviour or present service need.

Moreover, most historical usage-based Web service discovery solutions explore *group usage history* oriented context, where other user's behaviours in similar context are exploited. The advancement of social networks and Web 2.0 have made it possible to acquire shared experiences of users with other users who have similar attributes. One popular approach used to build discovery/recommendation systems that exploit user's with similar behaviour for the matchmaking process is *collaborating filtering* (CF) method (Rich, 1979). There are two classes of CF: (i) the memory-based CF approach, and (ii) the Model-based CF approach (Bobadilla, Ortega, Hernando & Gutiérrez, 2013). Most web service discovery system use the memory based approach, where all users are required to rate part of all web services in the repository based on their historical usage experience. Then, users who share certain similarities are grouped as neighbours

using some similarity algorithm such as *Pearson correlation* (Sheugh & Alizadeh, 2015; Benesty, Chen, Huang & Cohen, 2009) or cosine similarity (P. Xia, Zhang & Li, 2015). Then, when a user submits a service request, the system will recommend a set of candidates based on the user's neighbours' previous decisions (Rong & Liu, 2010). The major difference between works that based their service discovery solution on the CF based approaches how they define and process user ratings for each web service. For instance, Sreenath and Singh (2004) introduced a collaborative evaluation system for Web service selection by utilizing agent as oppose to using user to collect the rating for each web service. (Shao et al., 2007) adopted memory-based CF for QoS-aware Web service discovery. The authors used QoS values as ratings and sought to identify similar users by observing similar QoS experiences. Authors in Manikrao and Prabhakar (2005) introduced Web service selection method that exploited users ratings. (Kerrigan, 2006) proposed a CF-based Web service selection approach using goal and preference together without using rating for service discovery. One main challenge of usage-based context is that they require sufficient empirical information to perform well otherwise the service selection and matching process could be worse as the irrelevant experience would be pulled into the discovery process.

- *Process oriented context:*

  In comparison with the usage-based context, the process oriented context capture information from the current state of discovery process (Rong & Liu, 2010). The process oriented context focus on studying the reaction of the retrieved set of Web service candidates and then optimize the discovery. Apart from the feedback-based process oriented context, another perspective to this context is the Web service composition process. Due to the increasing complexity of user's software

requirements, rarely can a single Web service fulfill the requirement, hence, services composition provides a means to address this challenge. A handful of Web services will be invoked in certain composition processes to fulfill users' requirement. From this perspective, Web service discovery could be considered from the context of service composition. Cervantes and Hall (2003) proposed an approach to automate Web service dependency management in service-oriented computing .

- *Other context:* Other attributes such as trust and reputation , goal, domain can also be integrated with service discovery process (Rong & Liu, 2010)

2.6 Data Mining and Machine Learning

During the last few years, the method for discovering online Web services has been evolving at a rapid pace. There is a plethora of information available on the many techniques and approaches that have been utilised to address the problem of enhancing web service discovery. Recently, data mining and machine learning techniques (Jordan & Mitchell, 2015) have become popular research topics with application in various domains and new methodologies developed all the time. Emerging Web service discovery approaches are exploring these techniques to build service discovery solutions. Commonly used techniques for this purpose include *Clustering* (Agarwal, Sikka & Awasthi, 2020; Obidallah et al., 2020; Zhao, Chen & Xu, 2019), *Classification* (Swami Das, Govardhan & Vijaya Lakshmi, 2020), Deep Learning (Lizarralde et al., 2020; Yang, Ke, Wang & Zhao, 2019), *Word Embedding* (Lizarralde et al., 2017) and Other natural language processing techniques such as *Query Enhancement* (Lizarralde et al., 2019, 2020; N. Zhang et al., 2018), topic modelling, service description reconstruction (Zhong et al., 2018). These service discovery approaches are described below:

2.6.1 Clustering-Based Web Service Discovery

Clustering algorithms have been used in many service computing research works to boost the discovery and recommendation of Web services. They are used to group Web services into distinct clusters in order to minimize or summarize the search space of Web service using various similarity metrics. Obidallah et al. (2020) described various clustering techniques used in Web service discovery, commonly used techniques included *K-Mean Clustering approach*, *Hierarchical Agglomerative Clustering* (HAC) , *Neural Network-based clustering*. For K-mean clustering-based Web service discovery approaches, Authors in (Tian, Sun, He & Ji, 2016) and (Tian, Wang, He et al., 2016) employed K-mean clustering with Natural Language Processing technique for functionality-based Web service clustering. In order to improve the performance of the discovery system with respect to the clustering approach used, the authors incorporate auxiliary long text from Wikipedia. Ma et al. (2008b) utilized modified k-means for functional clustering and cosine-based similarity between service documents. The approach involves two steps that decompose Web service collection and semantically match services. A large Web service collection is divided into several clusters, then SVD (Singular Value Decomposition) technique is applied to the clusters so as to match the services against the user query. Liang et al. (2014) employed K-mean and T-FIDF approaches to co-clustered WSDL documents and the words after extracting the WSDL document features using NLP . Similarly, Rodriguez et al. (2010) used different types of K-mean approaches for functional clustering after utilizing NLP for pre-processing steps on the Web service documents. A topic-based clustering approach is used Shi et al. (2017). The authors used augmented LDA model to enhance the clustering process based on Web service functional features.They use *Word2vec* (Goldberg & Levy, 2014) to train the latent topic information on online Web service descriptions, where Web services with the same subject are grouped together, motivated by the absence of text

information. Similar terms from web services documents were clustered into similar clusters throughout the training phase. Ma et al. (2008a) utilized k-means clustering to remove and filter out irrelevant Web services. The author also used probabilistic latent semantic analysis (PLSA) to capture word semantics. For similarity measures, NLP, term frequency–inverse document frequency (TFIDF), and cosine and Euclidean distance are employed.

For *Hierarchical Agglomerative Clustering* (HAC), Kamath and Ananthanarayana (2014) used HAC and WordNet in a crawler-based system to acquire service functional descriptions and cluster them based on the functional attributes. In order to improve service clustering using different similarities metrics, Kumara, Paik, Chen and Ryu (2014) and (Kumara, Paik, Koswatte & Chen, 2014) employed HAC for functional clustering, TFIDF, WordNet, and Ontology for term-similarity computation in a multiple-step service clustering method that involved a feature extraction phase using NLP, ontology-learning phase, similarity calculation phase, feature-integration phase, and clustering phase. other approaches such as Neural Network-based clustering are also emerging. For instance in (L. Chen, Yang, Zhu, Zhang & Yang, 2013) applied modified Kernel Batch Self-Organising Map (KBSOM) neural network to cluster Web services using Word-Net and Latent semantic index (LSI) to generate vector representation feature of Web services. For the clustering and matching processes, the authors used Cosine and TFIDF and Mahalanobis distance as the similarity metrics.

2.6.2 Web Service Description Reconstruction And Query Enhancement

Beyond Web service discovery, other initiatives have been made in the Information Retrieval (IR) community to deal with the vocabulary problem that affects the accuracy of keyword-based discovery system. These developments are translating into research

solution in Web service discovery domain. Method such as *Service request expansion* (Mukhopadhyay & Chougule, 2012; Paliwal, Adam & Bornhovd, 2007), *Query expansion* (Lizarralde et al., 2019) and *Service description reconstruction* (Zhong et al., 2018, 2016). These approaches are mainly employed in service discovery to reduce the vocabulary gaps in natural language used to describe the service profiles and user service request, and therefore , they can help in enhancing Web service discovery. In order to enhance Web service discovery system performance, Paliwal et al. (2007) modified user's service request by expanding through combination of ontologies and LSI. The authors use the domain ontology to create Web service request vectors, then extract features from selected WSDL files to create the LSI classifier's training set, and finally project the description and request vectors. They use the cosine similarity measure to compare the service description documents and requests, and then retrieve appropriate WSDL service descriptions. The mapping of ontologies is done using semi-automated method. The keywords used for searching the system are chosen from service request by preprocessing the request are removing meaningless word.

Query expansion (Carpineto & Romano, 2012; Lizarralde et al., 2019) approaches are designed specifically to solve vocabulary problem associated with keyword-based/document-oriented service discovery systems by finding features correlated with query terms. This expansion can be applied to both queries and documents. It involves addition of terms that are not initially included in a query that can be derived from the other terms, for example similar terms (synonyms). Document-oriented methods work on the same principles, but instead, utilizes documents as a source. Although working on a document may introduce noise due to the greater number of words to expand, it does open the door for newer context-aware natural language processing (NLP) approaches (Lizarralde et al., 2019). Document expansion can leverage the much higher number of terms documents contain compared with queries. Query expansion has been applied using different corpora including WordNet and it has been shown to be capable of improving

service discoverability (Vechtomova & Karamuftuoglu, 2007). Instead of relying on static, subjective service descriptions provided by service providers based on their opinion of the services and language knowledge , authors in (Zhong et al., 2016, 2018) explore service description reconstruction to improve Web service discovery. The authors leverage service usage history information and service occurrence in mashups to improve the service descriptions quality.

Learning-based Web discovery approaches

Recently, alternative feature learning-based approaches such as Word-Embedding (Lizarralde et al., 2017), dimensionality reduction and Auto-encoders (Lizarralde et al., 2020) are been used to address challenges associated with syntactic-based and semantic-oriented Web service discovery approaches. These approaches are used to learn features from Web service descriptions. In order to make Web services registered in different syntactic-oriented registries available to service consumers, the registries such as programmableWeb usually process each service description in order to make the it available to consumers to discover. The additional processing can be divided into two phases (Lizarralde et al., 2020): First phase is the *pre-processing* stage, which involves two major tasks including *tokenization* of service descriptions, and creation of *Bag of Words* (BoW). The process for creating the BoW involves polishing, splitting, stemming and removing irrelevant words or stop-words from the BoW (In case where Word Embeddings are to be employed for service description representation in the registry, only stop-word removal process is done, no stemming – this is because in order to add features to the BoW , the features must be converted to vectors, and once this is done, it cannot be reverted)

The second phase involves *indexing* process where the pre-processed BoW are transformed into vectors and an index is created in the service registry. Different techniques such as LSA (Latent Semantic Analysis) (Dumais, 2004), Vector Space

Models and Variation Auto-encoder (Lizarralde et al., 2020) can be used to create the service description vector space. Most research works utilize VSM to index service descriptions, where descriptions are transformed to vector using TF-IDF weighting scheme (Platzer & Dustdar, 2005). The drawback of TF-IDF representation is that it results in big sparse vectors that become ineffective as the vocabulary increases size (Lizarralde et al., 2020), which applies to Web service registries in this case as they contain thousands of descriptions from different providers. To address this challenge,dimensionality reduction techniques such as LSA or Principal Component Analysis are usually employed to find concepts in data, group terms in the vocabulary and reduce sparsity. Recently, researchers are introducing new approaches like Word Embedding (Lizarralde et al., 2017) and Variation Auto-Encoder (VAE) (Lizarralde et al., 2020) for generating good, non-sparse representation of service descriptions that can hold valuable information for improving service discovery.

2.7 Social Networks For Web Service Discovery

As new services are emerging , and the number of Web services continue to grow rapidly with their and diversity, a wide range of similar functionalities are expected to be offered by a vast number of Web services from different providers. More Web services are expected to engage in outsourcing part their functionalities to other services resulting into an increase in number of mashups (Metrouh & Mokhati, 2013). Due to the above reasons, Web service discovery approaches like the syntactic and semantic-aware approaches are no longer effective to discover services. One of the key reasons for this is that most the traditional service discovery approaches consider Web services as an isolated functional island that does not interact with one another (W. Chen et al., 2015b). In addition, the registry-based Web service discovery approaches like UDDI have several drawbacks because of their centralized structure and their reliance on

static descriptions for discovery, which makes searching, selection, and localizing Web services challenging. Moreover, these approaches suffer low precision because of the way they handle Web services , that is, as an isolated functional element (Metrouh & Mokhati, 2013). To ensure effective discovery and retrieval of appropriate Web services, it is important to devise new mechanisms that can be scaled over time and improve service discoverability. Recent research works (Maamar, Hacid & Huhns, 2011; Maamar, Faci et al., 2011) in Web service discovery indicated that service engineers can benefit from various Web services' interactions including *collaboration*, *substitution* and *competition* to construct a social network for Web service discovery. And that through these networks, a service can be discovered based on its connection and affinity with other Web services in the network.

This section provides a general background to the cross-path between social computing and ervice oriented computing domain. It presents overview of using Social network for Web services discovery, different network approaches and the state-of-the-art approaches of constructing such Web service social network. In addition, existing works based on Social Web Service discovery approach are discussed. Note that in this section, the terms **Social Web Service Network**, **Social Web Services** and **Web Service Social Network** are used interchangeably in this work.

2.7.1 Overview of Social Networks

Social networks [6] are online communities (Metrouh & Mokhati, 2013; Hafsi, Gamha, Njima & Romdhane, 2020), which consist of "actors" (nodes) that can be businesses, services, poeple like authors,friends, professionals, teams or any social entity or groups interacting over the Web, and connected through set of social ties or relationships (edges), such as friendship, co-working, knowledge or information exchange in varied

[6]https://en.wikipedia.org/wiki/TheSocialNetwork

context like entertainment , politics, religion, dating or business. The capacity of social networks to capture real-world events like cooperation, rivalry or competition, and alliances or partnerships makes them a powerful instrument in several domains .

Social networking brings a new dimension to Web by providing a new novel channel of communication between the communities on the Web. Popular social network services such as *Facebook*, *MySpace*, *Twitter* and *LinkedIn* currently play significant roles in the day-to-day interactions of millions of internet users across the globe (Maamar, Wives et al., 2011). Businesses are now integrating these services into their customer relationship systems to connect with their customers. Organisations are now capitalizing on the social applications to attract new costumers, identify and connect with their suppliers and other stakeholders and study their competitors. The growing interest in social networks has opened up many new spaces of possible research in computing. Enterprises can use social networks to collect and disclose various informal relationships amongst their stakeholders (Maamar & Badr, 2009).

The study of these networks could provide better understanding into how and why entities or network actors interact with each other, as well as how these interactions can be altered, leveraged or exploited to improve other system processes. McDonald (2003) and Konstas et al. (2009) analysed the application of social networks to recommendation process. (J. Zhang & Ackerman, 2005), investigate search techniques that aid in the discovery of knowledge on social network. This is very important for locating the proper people who can provide the correct solution to a given situation. Maamar (2003) demonstrates how the social context is important for the success of e-commerce applications. These apps have various sorts of relationships depending on the parties involved in these relationship. Through sell, bargain, and purchase connections, shoppers and vendors can be connected. A request-for-advice relationship can link shopper and friend participants. The following subsection describes possible overlap between between social networks and Web service discovery.

2.7.2 Social Network of Web Services

The mix of social computing domain (exemplified by social applications including social networks) and service oriented architecture with is main implementation being Web services, gives birth to the notion of *Social Network for Web Services* (Hafsi et al., 2020; Maamar, Wives et al., 2011). Social network of Web services or Social Web Services is different from the conventional social networks like Facebook, Twitter and LinkedIn (Metrouh & Mokhati, 2013). Social networks are based on the absolute cooperation and mutual assistance between their members, that is, no competition exist between the members. On the contrary, Web services in social networks are known for competing as each Web service in the network wants to be: (i) part of a mashup or composition services; (ii) a substitute or replacement for for faulty or low quality service (iii) invoked to augment the currently invoked services (Maamar, Wives, Badr & Elnaffar, 2009).

Constructing a Web service social network involves an incremental and continuous process, which starts when a service is chosen for the first time to participate in as either a partner in mashup or a replacement/substitute in case of failure. Co-invocation among Web services and dependency interactions between a service and other services in a given mashup scenario could initiate the links in the social network Future participation of this Web service in other composition situations would allow it to expand its social network, but it would also necessitate a prospective evaluation of existing edges, as new edges may need to be introduced (Maamar et al., 2009). Additionally, the status of Web service social network could be reviewed when a service becomes obsolete and ceased to function. In such case, the node and the edges associated with this service are to re-evaluated, which may affect the overall structure of the network. In this context, as Web service ecosystem evolves and new services are continually published, it is expected that new interactions, relationships can be formed and existing ones may

become obsolete, inactive or dynamically modified. The following subsection described the requirements for building Web service network.

Building Web Service Social Networks

The construction of social network for Web service requires the following steps (Maamar, Faci et al., 2011):

- **Identifying the Web Service Social Network Components :** Generally, a typical social network's components include the nodes and the links or edges. For Web service network, these corresponds to the Web services and their relationships or interactions such as similarity, collaboration, invocation, competition and substitution. For collaboration-based social service network, where invocation relationships between a composition Web services and their component are presented as edges, service requester can explore or consider other services that are used in a composition with a particular service at hand (Lyu et al., 2014; Hafsi et al., 2020). For a Social Web Service networks where some edges are formed based on substitution relationships , a Web service can accept that similar peers will replace it when it is obsolete or fails. Such substitution is possible owning to the similarity between the services functionalities. In order to achieve a more effective substitution of Web services, the services non-functionalities should also be considered that is, non-functional attributes of the services should be similar as well. And for the competition-based links , two Web services with thesame functional properties and non-functional properties may compete for invocation preference, and user will only consume one.

- **Matching Web Services Analysis** In order to determine the nature of social interactions between Web services, their respective functional and non-functional attributes can be matched again each other. The matching would indicate whether

two adjacent services are similar or complementary.

- **Social Network Management** The social network management step allow for role specification for nodes. A special type of Web service node is defined as the *root*. The Web service root node is defined with respect to the three states that constitute a social network's life cycle; the first state is the *building state*, which defines any service that will be part of future social network. Root selection can be random or defined, and other service nodes can connect to this root. The next state is *exploitation state*. When Social Web Service is used to facilitate service composition, two cases emerges: i.) *collaboration and/or competition* , where any component service in the composition stands a chance of becoming the root, and then potential candidates compete and collaborate to make up the require composition functions. ii.) Then when there is a failing component, a substitution take place where the failing component is the root service. The *expansion state* occur when the Web service that is added to the network is the root. After determining the necessary relationships between the services, the network can be grow accordingly by adding new nodes.

- **Initial Edge Weight Evaluation** For a weighted service social network, the initial weight of an edge between 2 adjacent nodes correspond to the degree of similarity between the nodes. As the network grows, the weight values can be updated.

- **Navigating Social Service Network** To effectively navigate through the service network, service engineers or consumers require some support. Each Web service root could serve as an entry point to initial navigation. Depending on the type of interactions being sort for, certain factors like popularity can be used to drive peer identification or link formation.

- **Ongoing Edge Weight Evaluation** This involves continuous evaluation and update of edge weight. It reflects the Social Web Service role in service discovery (Maamar, Faci et al., 2011).

Existing Web Service Social Network For Service Discovery

As for the research on constructing service social networks, Fallatah et al. (Fallatah et al., 2014) proposed to add service-service, user-user, and user-service links to build a service social network. Based on the network, metrics such as user popularity, service market share, and user satisfaction can be measured. Simulation was done but how to build such network from real-world data was not discussed. Semantic information mined from service descriptions is a good reference for adding links among services. Wang et al. (H. Wang et al., 2010) used domain knowledge to calculate the degree of semantic match between any two services and then a threshold can be set to determine the number of links in the network. Similarly, Feng et al. (Feng et al., 2015) constructed three types of service networks based on the *subsume*, *sequential-total* (the output of service A covers the input of service B), and *sequential-part* (the output of service A partially covers the input of service B) semantic relations. Clearly such networks are static without considering any dynamical properties. From the evolving network perspective, Chen et al. (W. Chen et al., 2015b) built a service social network partially based on the Bianconi-Barabási (BB) model. One limitation of their work is that the fitness parameter of an existing service node is calculated dynamically on the arrival of a new service, while the BB model requires a quenched/fixed fitness value for a node, which makes the closed-form solution of the BB model not applicable to this network.

2.8 Complex Network Theory and Applications

Complex networks described a wide range of real-world systems including internet, WWW, computer networks, network of chemicals linked with chemical reactions (Albert & Barabási, 2002). The practice of network science is based on the ideology that all systems are networks. Theoretically, networks are considered as abstract mathematical objects composed by vertices that are connected by various links. Vertices could represent different objects, people, activities, services, systems, routers, protein, web pages, etc., depending on domain of application. For example, the World Wide Web is a vast example of virtual network that comprises of web pages as vertices, which are connected by hyperlinks. Likewise, in service oriented computing, several combinations of related services and workflows can be visualized as Networks. The study of complex networks follows the conventional approach of graph theory, which could be implemented randomly or in a regular fashion. The network is formed by continual addition of new vertices based on some pre-defined constraints and increasing the number of vertices added to network throughout its lifetime. Vertices could be stand-alone systems, individual system component or workflow, while edges represent the relationship between these vertices. Although nodes and links may possess different meanings and interpretations, nevertheless, complex networks exhibit distinctive statistical and structural resemblances. Therefore, a generic model can be set up to capture significant features of a complex network to enable thorough analysis of such networks.

Complex networks are usually modelled using various network theories, which are used in studying the component interactions and topology of complex networks. Large-scale networks with no obvious design concept were initially described as random networks](Barabási, 2012), a model originally initiated by Paul Erdos and Alfred Renyi ER (Erdős & Rényi, 1961) (Erdos, Rényi et al., 1960). ER model considered start nodes N in a network and connect every pair of nodes with the probability p,

creating a graph with approximately $pN(N-1)/2$ edges which are distributed randomly. Recent advances in computational ability and the amount of data available in various field provide researchers access to diverse network related databases prompting more research in the study of complex networks. More Recently,there has been a shift from the traditional networks models to Scale-free model proposed by Barabasi and Albert (Bedogne & Rodgers, 2006). This was due to various studies that shows that ER model failed to incorporate the topology of large-scale real network, which emphasis the influence of power-law degree of distribution. The model failed to reproduce many of the observable real world network properties (Cohen, Havlin & Ben-Avraham, 2003). On the other hand, Scale-free model have emerged from networks of different real world contexts, such as the World Wide Web, Protein interaction network, Scientific citation network. Scale free networks are characterized by having power law degree of distribution ,a phenomenon that is popularly exhibited by real world networks (Barabási, 2016). Preferential attachment mechanism is a dominant concept that is traditionally believed to underscore the emergence of power-law degree of distribution (Nguyen & Tran, 2012). The idea of preferential attachment is termed the a "rich-get-richer' mechanism, where a node with higher degree is more likely to be connected by a new node joining the network. However, various research related to network growth (Nguyen & Tran, 2012), (Chattopadhyay & Murthy, 2017a),(Bedogne & Rodgers, 2006) , (M. Bell et al., 2017), (Caldarelli, Capocc, Rios & Munoz, 2002) have argued that the connectivity power-law behaviour is neither related to dynamic attributes nor preferential attachment of complex network.

2.8.1 Scale-Free Network and Power-law Behaviour of Real World Networks

Many real world network has been considered to exhibit the attributes of scale free model as described by Babarasi and Albert. Such networks display power-law degree of distributions (Barabási & Bonabeau, 2003), (Bedogne & Rodgers, 2006) and are fundamentally based on two generic mechanisms commonly exhibited by real world networks. In contrast with random network model described by ER (Erdős & Rényi, 1961), (Erdos et al., 1960), which is deeply within pure mathematics and starts with a fixed number of vertices that are randomly connected or rewired without emphasis on modification of number of vertices or growth (Albert & Barabási, 2002). Scale free model introduce growth attribute into network modelling as portrayed by most real world network. Starting with a small number of nodes and gradually increasing over lifetime of the network. In addition, unlike other network models that assume that the probability that two nodes are connected or rewired is independent of the vertices' degree, that is new links are introduced randomly, scale free model indicates that real networks exhibits preferential attachment mechanism such that connectivity probabilities are proportional to the degree of target node. That is the probability p that a new node links with an existing node i with degree ki is related by:

$$p(k_i) = \frac{k_i}{\sum_j k_j} \tag{2.1}$$

After time t, the algorithm results in a network with $N = t + mo$ nodes and mt links, Where N is the set of nodes to which the new nodes could link. These two fundamental ingredient (growth and preferential attachment) formed the basis to which Scale Free Networks model exist. This research work adopted this mode because of the suitability of its characteristics to real world network like the APIs network, it is

considered suitable because of the following properties:

- It is characterized by having power-law degree distribution .
- It supports the growth characteristic experiences by most real world networks
- Various empirical studies validate its suitability to real world networks.
- It follows preferential attachment mechanism, which correlates the degree of some nodes to their age or time of publication (M. G. H. Bell et al., 2017; M. Bell et al., 2017).

2.8.2 Preferential Attachment

Preferential attachment concept follows a rich-get-richer mechanism, such that nodes with higher degrees are prioritized and more likely to acquire more links. This has been the focus of scale free network approach to growth over some period. However, recent efforts in Complex network study (Chattopadhyay & Murthy, 2017a) (Caldarelli, Capocci, De Los Rios & Munoz, 2002), shows that in situations where information about degree of distribution of single nodes are not available for modelling the network growth, it will be difficult to follow the preferential attachment assumption in such scenario. In many real world network, there are other factors and preferential attachment that can influence network growth and ability of a node to acquire more connections.

2.8.3 Node Fitness

It has been repeatedly shown in many context of real world network that new nodes in a network can be relatively popular in terms of their connectivity with other nodes (Barabási, 2016) (M. G. H. Bell et al., 2017). For example, despite the fact that Google was a late-comer, it quickly became the most preferred search engine, overtaking other search

engines such as Alta Vista in quality of service, performance, number of connections, and quickly turn out to be the biggest hub of World Wide Web. In order to make provision for this kind of behaviour in networks, researcher (Barabási, 2016; Chattopadhyay & Murthy, 2017b; Bedogne & Rodgers, 2006; Caldarelli, Capocc et al., 2002) have suggested a growth model such that a fitness parameter be define for such behaviour to take into account the intrinsic property of each node. This property enables each node to compete for edges at the expense of other nodes. The concept of fitness of nodes can be considered as the aggregation of several attributes of a given node, which contributes to its propensity to attract new connections. These attributes could be related to several intrinsic quality of a node such as the quality of benefits provided by these nodes, the rank, closeness or the between-ness (Chattopadhyay & Murthy, 2017b). The attributes could also be the node degree as a dynamic parameter of the network that varies as the network grows (M. G. H. Bell et al., 2017).

2.8.4 Small-World Networks

Watts and Strogatz (1998) show in their work that the topology of some real-world networks including social networks is neither completely regular nor completely random, they refer to this group of networks as *small-world* networks.These networks are characterized as *highly-clustered, like regular lattices and have small characteristics path length like random graphs* (Watts & Strogatz, 1998; Latora & Marchiori, 2001). Numerous real-world network systems have been described as exhibiting small-world properties including systems like the internet, social networks and groups, biochemical pathways. Eventhough, the widely acceptable definition of small-world network indicate that it has clustering similar to a regular lattice, and its path-length similar to that of random network. In reality, however, small-world networks are often defined by comparing clustering and path length to a comparable random network. Unfortunately,

this means that networks with very little clustering can be classified as small-world networks, which they are. In order to separate such not from sharing thesame definition with an highly clustered network that behaves like small-world network, Telesford et al. (2011) define some metrics that can used to quantify small-world properties and places the network under consideration on a spectrum ranging from lattice to small-world to random. Some of these metrics are defined as follows:

- **High clustering**: Small-world networks are differentiated from other networks by two characteristics, the first of which is strong node clustering (C). Mathematically, C is the fraction of edges e_i that exist between the neighbours of a specific node i compared to the total number of potential edges between neighbour (Bullmore & Sporns, 2009):

$$CC_i = \frac{2|e_i|}{k_i(k_i - 1)} \tag{2.2}$$

  Equation 2.2 is used to compute the value of C at an individual node of degree k. A network's overall clustering may be calculated by averaging the clustering of all individual nodes. High clustering encourages specialisation because local groups of tightly linked nodes may easily exchange information and resources.Clustering is a simple concept to grasp from a conceptual standpoint. Clustering, in a real-world example, shows the likelihood that one's friends are also friends of one's friends.

- **Short Path-Length**: Small-world networks have short path lengths (L). Path length is a measure of the distance between nodes in a network, calculated as the average of all feasible node pairs' shortest geodesic lengths:

$$L = \frac{1}{N(N-1)} \sum_{i \neq j \in G} l_{ij}. \tag{2.3}$$

where l_{ij} is the shortest geodesic distance between node pairs i and j. Small L values guarantee that information or resources travel quickly across the network. This feature enables dispersed information processing on technical networks and promotes the six-degrees-of-separation phenomenon that is frequently described in social networks.

2.9 Chapter Summary

In this chapter, comprehensive review of frameworks, concepts, models, techniques and architectures that form the basis of this research work are presented. The chapter presented the crossroads of the two major domains of knowledge considered in this research including complex network and service computing. For service computing, various concepts, methods, technologies and components of the service computing and its related architectures are discussed. For complex network stream, complex network theory and various relevant developments in network science, particularly with respect to complex network applications are discussed. The potentials of complex network in solving discovery problems and related network-based discovery approaches in service computing are discussed.

Chapter 3

Analysing the Topology of Web Service Ecosystem

"All systems are networks," is the de facto network science maxim (Pham, Sheridan & Shimodaira, 2015). The study and investigation of these large-scale networks (also refers to as complex networks with emergent topological properties that are not found in simple networks) structures with their evolution are considered a hallmark of network science. Over the years, complex networks have been extensively studied and several significant discoveries have been made including the well-acclaimed small-world networks (Watts & Strogatz, 1998) and scale-free networks (Barabási & Albert, 1999). Large-scale network research have focused more or less on two related tasks (Pham et al., 2015): (i) Studying the emergence of topological features in complex networks and investigating possible mechanisms underlying the network formation. (ii) Modelling of dynamical processes involved in complex network systems in a way that enables effective exploitation of the known topological features.

This chapter is mainly concern with the first task. In particular, in order to get clear insight into the structure and the dynamical mechanisms that drives a typical Web service ecosystem, a comprehensive analysis for investigating the topological

feature and dynamical mechanism of the ProgrammbleWeb registry (as a template for Web service ecosystem) using network analysis approach is performed. The main motivation of using complex network as a simplified representation of real-world systems like Web service ecosystem is that they shed light on the behaviours the systems via the study of the underlying patters of connections (Pham, Sheridan & Shimodaira, 2017). Eventhough, this over-simplification for systems depends mainly on domain-specific information, this approach have been shown to offer a first view of the the system's topological features and can serve as a corner stone for subsequent in-dept studies or model design. Various mechanisms that governs a network's topology and evolution have been investigated and found ubiquitous among many real world networks. In particular, *preferential attachment* and *growth* have garnered special attention in evolving complex networks research (Barabási, 2016; Pham, Sheridan & Shimodaira, 2016), not only because they are fundamental to explaining the topological features observed in many real world networks but also because they have been empirically validated to be the drivers of many evolving networks. For instance, the topology of the Internet, the World Wide Web and the citation network have been investigated using evolving network models and shown to be fundamentally governed by the PA and growth mechanisms (Albert, Jeong & Barabási, 1999; Barabási, 2016; Barabási, 2012). In terms of evolving network models, the PA and growth driven Barabási-Albert (BA) model (Barabási, 2016) is the foundation of other models such as the fitness-based Bianconi-Barabási model (Bianconi & Barabási, 2001b).

Moreover, this chapter presents a methodology to quantitative characterize the static and dynamical evolution of the Web service ecosystem. *Web Service ecosystem* is considered as a collection of services including mashups, their relations and associated elements such as service consumers, provider or vendors in the service market (Huang et al., 2012b). The richness of the correlation between elements of this ecosystem results in a complex community structure (J. Zhang, Tan, Alexander, Foster & Madduri,

2011; Maamar, Wives et al., 2011). Moreover, the perishing of some existing Web-APIs and the emergence of new ones coupled with their dynamic collaborations drive the evolution of this service ecosystem over time(Adeleye et al., 2018). Mining the correlations between Web service domain is very important as it can help provide answers to questions like ; what sort of Web services are most likely to be used together in the future or in the past? What relationships are shared between connected services and how can they be exploited? Providing answers to these sorts of questions would help facilitate service consumers discovering suitable services for their applications. It could also help service providers locate candidate partners to collaborate with. To study the ecosystem topological properties, an analysis of the popularity (degree distribution) of the Web-APIs based on the mashup-API bipartite graph is conducted.

The remaining parts of this chapters are organised into several main sections: Section 3.1 presents the data structure, preprocessing and visualization of Web service interactions in the ecosystem. Section 3.2 presents a comprehensive analysis of Web service popularity distribution including fitting the distribution to some *classical models* including Power-law, Poisson, exponential and log-normal distributions and determining the goodness of fittings. Section 3.3 shows measuring of *preferential attachment* one of the key mechanism that drives the real-world network systems. Section 3.4 presents the computation of *similarity* in the ecosystem, which are other key measures of attractiveness for link formation between pair Web services. Section 3.5 concludes the chapter.

3.1 Notations and Definitions

This section describes various symbols and notation used in this work, and present several definitions related to the data schema of Web-APIs and mashups in a typical evolving Web-API ecosystem.

Definition 1 - Web-API and Mashups: *Web-API*, *Web-service* and *service* are used interchangeably in this paper. Web-API network comprises a collection of linked Web-APIs denoted as A={a_1, a_2,..., $a_{|A|}$}, where $|A|$ is the total number of Web-APIs in the network. Each Web-API a_i is associated with a description document W^{a_i}, which represents the Web-API's functional descriptions and original profile offered by the Web-API's provider. We represent M as the collection of all mashups in the service ecosystem, where M={m_1, m_2,..., $m_{|M|}$}. Each mashup m_i is associated with a description document W^{m_i}. A full list of notations are given in Table 3.1.

Definition 2 - Invocation Relationship: Suppose we have historical invocation records x Web-APIs in y mashups. We define the invocation relationship between the APIs and mashups as binary matrix $\mathbf{R} \in \mathbb{R}^{x \times y}$, where each element $r_{i,j}$ indicates whether or not an API a_i is invoked by a mashup m_j. That is , if API a_i is invoked by a mashup m_j, $r_{i,j}$ is set to 1, otherwise 0.

$$r_{i,j} = \begin{cases} 1, & \text{if } a_i \text{ is invoked by } m_j \text{ ;} \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

Note that $r_{i,j} = 0$ does not necessarily means API a_i does not have any relationship with mashup m_j, it could be that a_i is isolated in the ecosystem or/and m_j is not aware of a_i during its composition. Where each mashup m_i comprises of n number of component Web-APIs represented as $\{a_i, \ldots, a_n\}$, where $n \geq 1$. The invocation relationships between mashups m_i and the Web-APIs a_i means that a_i is a member of m_i.

Table 3.1: A summary of notations used in this paper

Notation	Description
G	a service network
V	Set of nodes in the given service network
E	Set of edges in the given service network
M	Set of mashup nodes in the given service network
A	Set of API nodes in the given service network
N	Network size
W	Service description document
$\|M\|$	Number of mashup nodes
$\|A\|$	Number of API nodes
$\|E\|$	Number of edges
W^a	Web-API functional description document
W^m	Mashup functional description document
k	Node degree
w	Service description word token
d	Pair-wise distance between two service nodes
α	Attachment dynamic exponent
γ	Power-law degree distribution exponent
θ	Angular position in the metric space
Π	Attraction probability
L	Web service popularity-based ordered-list
S_f	Web service functional similarity

3.2 Data Acquisition and Processing

This section presents the overviews of the datasets used for the analysis in this chapter, and the pre-processing step. Time-stamped raw data was collected from ProgrammableWeb [1], which is currently the largest Web service repository, and contains information regarding Web-APIs and mashups from $June$ 2005 to $November$ 2020. Since the $ProgrammableWeb$ backend database is not publicly accessible, only its web pages can be employed for collecting the data. Data scraping technique was employed to crawl data from ProgrammableWeb web pages, and examine the attributes of real-world mashup-API invocations. The web pages are separated into two categories: Web-APIs and mashups, where every Web-API has attributes including *name*, *tags*,

[1] www.programmableweb.com

short descriptions,*endpoint*, *provider information* , *publication date*, and *category*; similarly, each mashup also contains the above metadata plus the list of Web-APIs invoked within it. Table 3.2 gives an example of the ProgrammableWeb dataset with the attributes. Table 3.3 shows the statistics of data after the initial pre-processing. Before pre-processing the crawling process returns a list of 17,828 Web APIs and 6,340 Mashups for our analysis, and after pre-processing and removing redundant mashup points or mashup with less than 2 components, the data is then reduced to 17,828 Web APIs and 5,889, with 11,287 mashup-API interactions extracted along with the dataset. Only 1,525 Web-APIs are invoked in the 5,889 mashups or involved in the mashup-API interactions in the dataset, indicating a very low interaction matrix density of 1.26×10^{-3}.

The interaction matrix density shows how extremely sparse the mashup-API interactions in the ecosystems is. The very few number of Web-APIs involved in the interactions indicates that only few number of Web-APIs are actually exposed or consumed the service consumers. Over 92% of the mashups interact with just less than 5 APIs. Figure 3.1 shows the API invocation distribution across the mashups. Only about 9% of the entire Web-API in the ecosystem are involved in any interaction or invoked in any mashup. Majority of these interactions, frequently involve very few APIs which are the most popular. Figure 3.2 indicate this sparsity, showing about 8 popular Web APIs that have been invoked over 200 times.

To extract the popularity distribution of Web-APIs in ProgrammableWeb, we model the ecosystem in the form of an *affiliation network* that depicts the invocation relation between mashups and Web-APIs. As shown in Fig. 3.3, technically, the network is a bipartite graph, where the edges indicate which Web-APIs are invoked by which mashups: G=$(M \cup A, E)$ where M is the set of Mashups and A is the set of Web-APIs, and for any edge $(m \times a) \in E, m \in M$ and $a \in A$.

Although there are over 19,000 Web-APIs in ProgrammableWeb, only 1,525 of them

Table 3.2: Sample mashup and Web-API data form on ProgrammableWeb Dataset

Sample mashup and Web-API from ProgrammableWeb Dataset		
Attributes/Type	**Mashup**	**Web-API**
Name	Digireality	TwitPic API
Description	Real estate search engine in Czech Republic.	The TwitPic API lets you upload and post images to your Twitter account. You can upload an image for later posting or upload an image to TwitPic and automatically send it as a status update to Twitter.
Categories	Real Estate, Classifieds	Photos
Related API	Google AdSense, Google Maps, Google Maps Data	–
Publication Date	01.07.2019	01.08.2009
URL/endpoints	*https://www.digireality.cz*	*http://twitpic.com/api/*

Table 3.3: Summarize Features of the ProgrammableWeb Dataset

Number of Web APIs acquired	17,828
Number of Mashups acquired	6,340
Average number of Web APIs invoked by Mashups	2
Number of Mashups with less than 2 services	241
Number of Web APIs invoked in at least one Mashup	1,525

Table 3.4: Top 5 most consumed Web-APIs

Web APIs	Number of links
GoogleMap	2,072
Twitter	663
Youtube	557
Flickr	484
Facebook	377

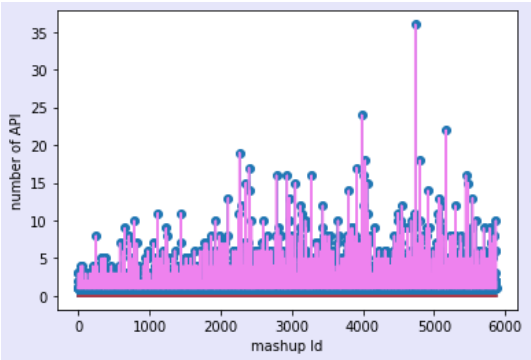


Figure 3.1: The number of API invoked per mashup

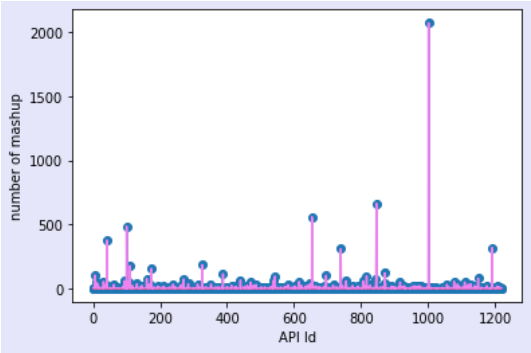


Figure 3.2: The number of connected mashup per Web-API

appear in one or more mashups. We found that the Google Map Web-API takes a center stage in the affiliation network, attracting 2,072 edges(mashup-consumption), which account for about 33% of the total mashups in the ecosystem. As shown in Table 3.3, Popular social media Web-APIs such as Twitter, Youtube, Flickr, and Facebook also appear 663, 557, 484, and 377 times respectively in the network. We also found that less than 7% of the Web-APIs involved in the network are consumed more than 100 times, and over 47% of the Web-APIs are used less than 4 times.

The complete affiliation network is visualized using the Force-Atlas 2 layout in Gephi [2] as shown in Fig. 3.4. The hubs as listed in Table 3.3 are clearly visible in the figure as the green disks with Google-Maps API being the largest one sitting at the left bottom.

3.2.1 Analysing Web-API Popularity Distribution

An integral part of analyzing the topology of a network system is the plotting and fitting of its degree distribution $p(k)$. Networks with long-tailed degree distribution that follows a power-law are known to exhibit the *scale-free* topology. Most real-world network systems such as the internet, WWW and the citation network are *scale-free*

[2] https://gephi.org/

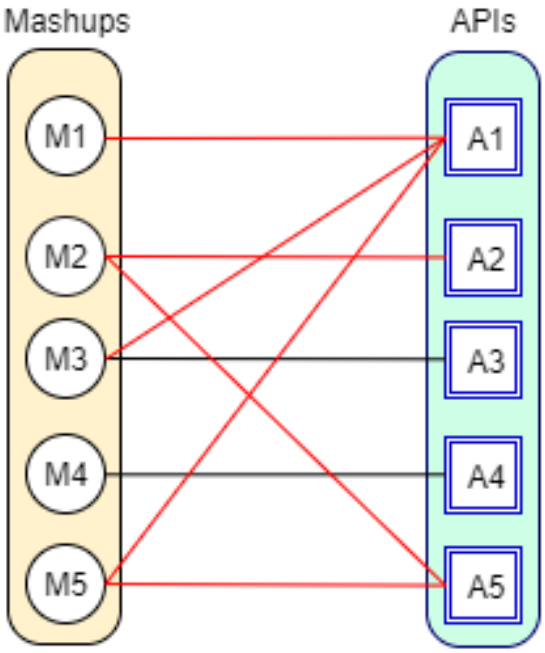


Figure 3.3: Illustration of the Mashup-API bipartite graph

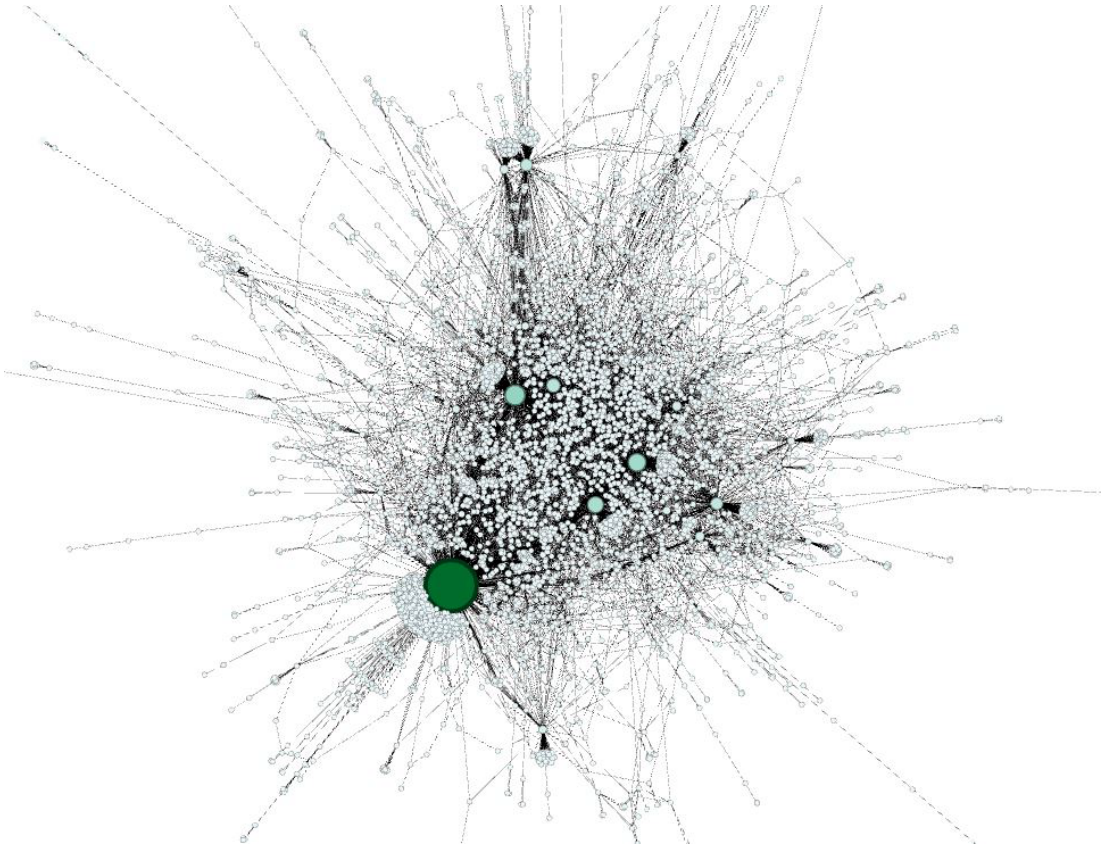


Figure 3.4: Visualization of the Mashup-API affiliation network

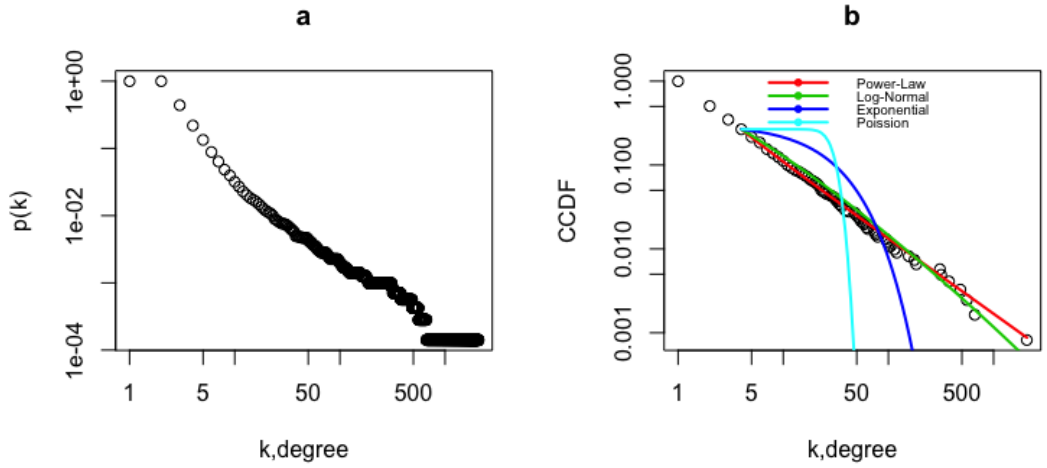


Figure 3.5: Degree distribution plot of the Web-APIs nodes in the affiliation network. (a) shows the linear-binned plot of the Web-APIs degree distribution, (b) shows the CCDF plot of the distribution with Power-law (PL), Log-normal, Exponential, and Poisson models fitted to it.

networks (Barabási, 2016). On the other hand, networks with exponentially-decaying-tail degree distribution are collectively referred to as *exponential Networks*.

Fitting Web-API Popularity Distribution to Classical Distribution Models

To gain insight into the popularity of Web-APIs, we plot the degree distribution of the 1,525 Web-APIs based on their degrees in the affiliation network. As shown in Fig. 3.5, both the PDF (Probability Density Function) in log-log scale, linear binning, and the CCDF (Complementary Cumulative Distribution Function) plot with Power-law (PL), Log-normal, Exponential, and Poisson models fitted to it. In Fig. 3.5a, the small degree region demonstrates a log-linear relation between $p(k)$ and k ($\log p(k) \sim -\gamma \log k$, or $p(k) \sim k^{-\gamma}$) , which is a typical feature of the scale-free network; while a plateau is formed at the large k region as typically we have only one copy of each large-degree node and this plateau affects our ability to estimate the degree exponent γ (Barabási, 2016). One way to extract information from the tail of the distribution is to use the

CCDF (Fig 3.5b), which enhances the statistical significance of the large-degree region, and if $p(k)$ follows the power-law, then the CCDF is also power-law: $P(k) \sim k^{-\gamma+1}$.

Fitting : In order to determine the best fit for the Web-API degree dataset, we first fit the data to four classical models including *Power-law* , *Exponential*, *Log-normal*, and *Poisson*. Fig 3.5b shows the result of the fitting when k_{min} = 3. We can see that both the power-law and the log-normal offer a good fit to the data, while the exponential and the Poisson fit poorly to the data.

To quantitatively measure the plausibility of each distribution, next we conducted a *goodness-of-fit* test based on the Kolmogorov-Smirnov (KS) distance which measures the difference between the model and the empirical data, and a $p\text{-}value \in [0, 1]$ is calculated to measure the model plausibility. The closer p is to 1, the more likely that the difference between the model and the empirical data is attributed to statistical fluctuations alone. If p is very small, the model is not a good fit to the empirical data (Barabási, 2016). Table 4.3 shows the resultant $p\text{-}values$ for each distribution. Clearly, power-law is the most plausible fit ($p\text{-}value$ = 0.7841) and next to it is log-normal (0.6871); for both exponential and Poisson, the $p\text{-}values$ are 0.0000 and 0.0002 respectively.

Exponent Estimating: In the above, we have justified that the power-law model provides the best fit to our data, next we use MLE (Maximum Likelihood Estimation) to estimate the scaling parameter/degree exponent γ (Muniruzzaman, 1957):

$$\hat{\gamma} = 1 + n\left[\sum_{i=1}^{n} \ln \frac{k_i}{k_{min}}\right]^{-1} \tag{3.2}$$

where k_i, $i = 1...n$ are the observed values of k such that $k_i \geq k_{min}$, k_{min} represents the minimum degree of node in the network.

The assumption for estimating the parameter is that $\gamma > 1$, since the case of $\gamma \leq 1$

does not exist in real world (Clauset, Shalizi & Newman, 2009).

When $k_{min} = 1$, the appropriate estimator for γ was given as:

$$\frac{\zeta'(\hat{\gamma})}{\zeta(\hat{\gamma})} = -\frac{1}{n}\sum_{i=1}^{n}\ln k_i \tag{3.3}$$

where $\zeta(\hat{\gamma})$ is the Riemann Zeta function.

Otherwise, when $k_{min} > 1$, the appropriate estimator for γ is:

$$\frac{\zeta'(\hat{\gamma}, k_{min})}{\zeta(\hat{\gamma}, k_{min})} = -\frac{1}{n}\sum_{i=1}^{n}\ln k_i \tag{3.4}$$

Using the method described in (Clauset et al., 2009), which is also based on the KS distance, we can find the optimal k_{min} with respect to each data point and select the value that gives the minimal KS distance between the CCDF of our data and the fitted model. The resultant γ value for the CCDF is around 2.2 (or the γ value for the PDF is 3.2), which is close to that of the Internet ($\gamma = 3.42$) (Barabási, 2016).

3.2.2 Measuring Preferential Attachment

Real-world networks reach their current size by adding new nodes to the network progressively, and a common phenomenon occurs, where new nodes tend to connect to existing nodes with high degree. This phenomenon is called *Preferential Attachment (PA)* (Barabási & Albert, 1999). If the probability that a newly arrive node connects to an existing node i is proportional to the degree of that node k_i, or

$$\Pi(k_i) = \frac{k_i}{\sum_j k_j} \tag{3.5}$$

then we call it *linear-PA*. The combination of growth and linear-PA play a critical role in shaping a network's topology and are responsible for the emergence of the *scale-free*

property (Barabási, 2016).

We can use the exponent α to classify different types of PA

$$\Pi(k) \sim k^{\alpha} \tag{3.6}$$

if α is 1, then PA is *linear*; if α is less than 1, then PA is *sub-linear*; otherwise PA is *super-linear* (Barabási, 2016).

We aim to detect the presence of PA in the Web-API node set of the affiliation network of ProgammableWeb and also measure its α value. To do so, we can examine the degree increase of a node i between a fixed span Δt: $\Delta k_i = k_i(t + \Delta t) - k_i(t)$. For example, if $\Delta t = 5$, $k_i(t + \Delta t)$ is the degree of node i after five new nodes joined the affiliation network. The relative change $\Delta k_i / \Delta t$ should follow

$$\frac{\Delta k_i}{\Delta t} \sim \Pi(k_i) \tag{3.7}$$

Actually, to reduce the noise we can measure the cumulative preferential attachment:

$$\pi(k) = \sum_{k_i=0}^{k} \Pi(k_i) \tag{3.8}$$

We employ both the *PAFit* method (Pham et al., 2015) and Newmans's method (Newman, 2001) to estimate PA. As we can see in Table 3.5, Node Spans 10, 20, 50, 100, and monthly all output consistent results of $\alpha \approx 1$, which demonstrates the existence of *linear-PA*, or *scale-free property*, of Web-APIs in the ProgrammableWeb affiliation network.

Table 3.5: Preferential Attachment Measurement

Node Span	α **(Newman)**	α **(PAFit)**
10	0.97± 0.05	1.09± 0.06
20	0.96± 0.05	1.08± 0.06
50	0.95± 0.07	1.06± 0.07
100	0.94± 0.06	1.05± 0.08
$Monthly$	0.96± 0.09	1.03± 0.09

3.2.3 Estimating Web-API Similarity for Network Construction

Being able to answers questions such as: *"How similar are two nodes in a network" ?* or *"Which other node are closely related to a particular node" ?* would be very helpful in discovery or recommendation tasks. However, hardly can a single similarity metric captures all the required relationship in a network. There are several ways in which two nodes could be similar. For instances, in World Wide Web, where nodes are web pages , two web pages may be considered semantically similar if they share related content or same words. Likewise, given the pattern of links between the web pages, a useful structural similarity measure that reflect how similar two nodes are can be defined (Leicht, Holme & Newman, 2006). Researches in network sciences have shown that the topological properties of a network carries real information about the creation of links between two nodes in the network. Therefore, it seems reasonable to consider structural or topological similarity measure in addition with other similarity measures in this case. Here we show methods used for quantifying both semantic and structural similarity for Web-API node in the affiliation network.

First, we define Web-APIs similarity measure as a function $s(a_i, a_j)$ that returns a numerical quantifier on the similarity between APIs a_i and a_j. In this case, the larger the $s(a_i, a_j)$ is, the more similar the two APIs are. Taking the inverse of $s(a_i, a_j)$ gives a distance measure $d(a_i, a_j)$ (defined in equation 3.9), where the similarity measure between a_i and a_j is considered as a transformed *euclidean distance* between two the APIs: $s(a_i, a_j) = 1 - d(a_i, a_j)$ or $s(a_i, a_j) = \frac{1}{1+d(a_i,a_j)}$

$$d(a_i, a_j) = \sqrt{\sum_{i,j=1}^{n} |a_i - a_j|^2} \tag{3.9}$$

Existing Web-API applications (B. Cao et al., 2017; Bianchini, Antonellis & Melchiori, 2017; Wan, Chen, Yu, Liang & Wu, 2016) utilize various deterministic measures to compute Web-API similarity. However, recent work (Hamilton, Ying & Leskovec, 2017) show that *stochastic measures* node similarity can achieve superior performance. Therefore, we define a more robust Web API similarity computation that integrate both deterministic and stochastic similarity measures by exploiting the semantics of service descriptions and properties of the *Affiliation* network . Specifically, we define a global service similarity measure as an aggregate of two key types of similarities that exist in the Web service ecosystem: *Functional Similarity* and *Structural Similarity* . The functional similarity is purely a semantic-based similarity where the idea of distance between set of descriptions (saved as documents) is based on the likeness of their meaning as opposed to their syntactic similarity. Structural similarity measure exploits the structure of the affiliation network, and captures service *proximity*.

Measuring Web service functional similarities

Since the Web service textual descriptions mainly reflect their predefined functionalities, we measure the *functional similarity* as latent semantic similarity between two adjacent services by exploiting the semantic information in the descriptions. We assume Web

services that shared explicit functional attributes can be considered similar or serve as substitutes in different capacities. We collect textual description of Web services including mashups. For each service, we create a document which contains the service textual description, category, tags and name including the service components. Consequently, we have service documents corpus with each document in the corpus capturing the functionality vocabularies of the service.

We employ Latent Dirichlet Allocation topic model (Blei, Ng & Jordan, 2003) to analyze each service document and obtain the associated topic distribution. We assume that service documents that share similar topic distributions to be functionally similar. We compute the functional similarity of services based on the latent similarity that exist between the service documents. Specifically, we apply LDA to analyze each service document and extract the associated topic distribution. Prior to utilizing LDA approach, we perform series of data pre-processing steps including *tokenization*, *removing stopwords* and *stemming* to extract feature vectors representing the documents' contents. Detail discussion on LDA topic model component is presented in section **??**. Finally, we define the functional similarity score between two adjacent Web services a_i and a_j as the similarity of the topic distributions $a_i : \theta_i$ and $a_j : \theta_j$. We achieve this by simply comparing the topic distribution of services a_i and a_j using the *Jensen-Shannon divergence* method defined in equation 3.10.

$$JSD(\theta_i||\theta_j) = 0.5 \times D(\theta_i||M) + D(\theta_j||M) \tag{3.10}$$

$$f_s(a_i, a_j) = 1 - JSD(\theta_i \parallel \theta_j) \tag{3.11}$$

where $M = \frac{1}{2}(\theta_i + \theta_j)$ and $D = (. \parallel .)$ is the smoothed version of the *Kullback–Leibler divergence* . Since the JSD is a positive definite measure of the difference between the two distributions θ_i and θ_j , where $0 \leq JSD(\theta_i \parallel \theta_j) \leq 1$ and $JSD(\theta_i \parallel \theta_j) = 0$ if only if $\theta_i = \theta_j$, we compute the functional similarity between services a_i and a_j using

equation 3.11. The similarity values were stored as matrix Fs.

Quantifying Structural Similarity.

We define Web service *structural similarity* from *proximity* perspective. We consider both local and global network structures of the Web service affiliation network. The local network structure characterised as *first-order* proximity captures the pairwise similarity between two nodes connected by an observed edge. The drawback with this notion of *first-order* proximity is that for two service nodes to be similar or close, they have to be connected. That is, nodes that are farther away in the network or disconnected from each other will always have zero similarity value. However, there exist some nodes that are not connected but are structurally equivalent or similar. Therefore, a *second-order* proximity is defined between pairs of nodes which captures the similarity of the pairs' neighborhood structures. This considers structural equivalence where two nodes share many of thesame neighbours in the network (Leicht et al., 2006; Carstens, Jensen, Spaniel & Hermansen, 2017). For instance, people can be consider as similar if they share common neighbours likewise two Web services can be considered similar in the service network if they share many common neighbours.

Common approaches for computing global proximity include *Jaccard index* and *Cosine Similarity* define in Equation 3.13 and 3.14 respectively. Where Γ_i is the neighborhood of vertex i in the network. The rudimentary measure of similarity between nodes i and j is given in equation 3.12

$$\sigma_{unnorm} = \mid \Gamma_i \cap \Gamma_j \mid \tag{3.12}$$

$$\sigma_{jaccard} = \frac{\mid \Gamma_i \cap \Gamma_j \mid}{\mid \Gamma_i \cup \Gamma_j \mid} \tag{3.13}$$

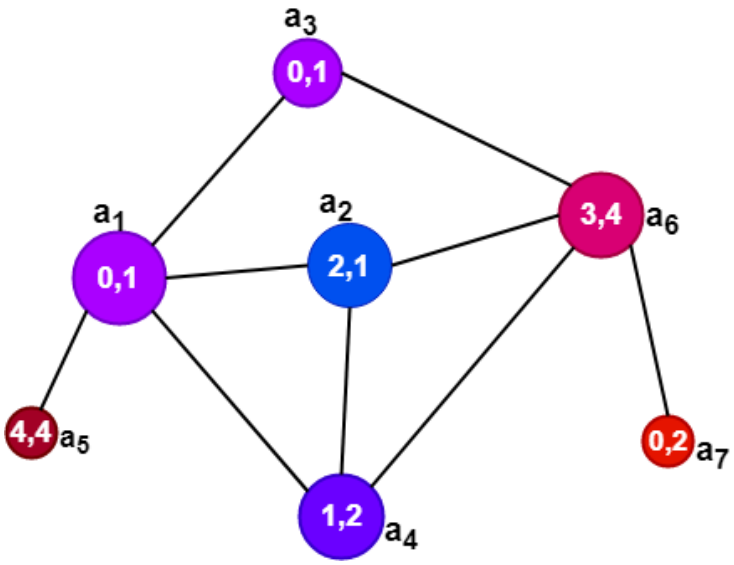


Figure 3.6: illustration for global service similarity computation

$$\sigma_{cosine} = \frac{|\Gamma_i \cap \Gamma_j|}{\sqrt{|\Gamma_i||\Gamma_j|}} \tag{3.14}$$

Figure 3.6 further illustrate various similarity measures consider in our network construction. The two numeric labels are functional attributes. Node a_3 and a_1 are closely functional related as they share same functional descriptions or attributes, hence, they maximally semantically similar, but the nodes are structurally dissimilar as their number of neighbours differ. a_3, a_1, a_7 are somewhat semantically similar, same applies to a_2, a_4, a_7. *Proximity-wise* a_5 is strongly similar to a_1, even-though both are semantically dissimilar. Structurally, a_1 and a_6 are strongly similar as they share thesame number of neighbours (same node degree) even-though they are functional attributes are different. In the global similarity setting, where all the three similarity measures are considered, a_2 will be the most similar to a_1 as they both share to some extent similar functional,proximity and structural attributes.

Web Service Structural Similarity with SDNE

For our network construction, we employ *Structural Deep Network Embedding* model (SDNE) (D. Wang, Cui & Zhu, 2016) to capture and integrate both local and global proximities simultaneously. We use SDNE to map the Web service affiliation network data into a low-dimensional space, where each service node in the network is represented as a low-dimensional vector and the original network structures (both global and local proximities) are preserved. The model exploits both structural levels using semi-supervised embedding learning technique. For the local structure preservation, the model exploit the pairwise similarity information to refine the node representations in the latent space using a supervised learning setting. For second-order proximities which characterises the global structure, an unsupervised learning approach was used to reconstruct the neighbourhood structure of every node. Then, both learned components where jointly optimized using semi-supervised deep model with *loss-function* defined in equation 3.18. The loss-function captures the two components. The first component shown in equation 3.17 is the loss function for the unsupervised learning component which captures the second-order proximity, where x_i is a sparse adjacency matrix which is the input to the model, $\hat{x_i}$ is the output obtained by reversing the calculation process of encoder. The encoder here consists of a nonlinear-functions that maps the input data x_i to representations space as shown in equation 3.15 and 3.16, where W^k is the $k-th$ representation layer weight matrix, b^k and y_{i^k} are the $k-th$ layer biases and hidden representation respectively. σ (sigmoid function) is the non-linear activation function.

$$y_i^1 = \sigma(W^{(1)}x_i + b^{(1)}), k = 1 \tag{3.15}$$

$$y_i^k = \sigma(W^{(k)}y_i^{(k-1)} + b^{(k)}), k = 2, \ldots, K \tag{3.16}$$

$$\mathcal{L}_{2nd} = \sum_{i=1}^{n} || \ (\hat{x}_i - x_i) \circ b_i \ ||_2^2 \tag{3.17}$$

$$\mathcal{L} = \sum_{i=1}^{n} || \ (\hat{x}_i - x_i) \circ b_i \ ||_2^2 + \alpha \sum_{i,j=1}^{n} s_{i,j} \ ||y_i - y_j||_2^2 + v\mathcal{L}_{reg} \tag{3.18}$$

A bias b^k is introduced to reduce the reconstruction error due to data sparsity, where $b_i = \{b_{i,j}\}_{j=1}^{n}$. If $s_{i,j} = 0$ (note $x_i = s_i$ and for if link exists between node i and j, $s_{i,j} = 1$), then $b_{i,j} = 1$, else $b_{i,j} = \beta > 1$. For the local structure preservation characterised by first-order proximity, second component of equation 3.18 captures that, where $\mathcal{L}_{reg}$ is an $L2 - norm$ regularizer term to prevent overfitting, which is defined in equation 3.19 For the model optimization, $\mathcal{L}$ is minimize as function of θ. Detail mathematical form of the optimization step is shown in the original paper(D. Wang et al., 2016).

$$\mathcal{L}_{reg} = \frac{1}{2} \sum_{k=1}^{K} (||W^{(k)}||_2^F) + (||\hat{W}^{(k)}||_2^F) \tag{3.19}$$

After getting the embeddings for each Web service node (including mashup nodes) in the affiliation network, the vector similarity measure between a pair of service nodes i and j is calculated as a function of the *cosine* of the angle between their embeddings using equation 3.14 [3], where θ_{u_i,u_j} is the angle between embeddings u_i and u_j. We store the resulting structural similarity into matrix S.

$$S_{i,j}^{n} = \frac{\cos(\theta_{u_i,u_j}) + 1}{2} \tag{3.20}$$

Finally, we integrate the functional (semantic-based) similarity Fs computed in equation 3.11 and the structural similarity in equation 3.20 to get the global similarity G_s as shown in equation 3.21.

$$G_s = f_s(a_i, a_j) + w_s.S_{(u_i,u_j)} \tag{3.21}$$

[3]https://documentation.sas.com

w_s is the model parameter which is estimated based on the functional similarity dataset.

3.3 Chapter Summary

This Chapter presents an empirical study of Web service ecosystem by investigating the evolutionary properties of service ecosystems and the complementary features of services and their compositions particularly on ProgrammableWeb. It examined the structure of the ecosystem and looks into the forces of attractiveness in the ecosystem using the ProgrammableWeb dataset. It presents the analysis of the relationships between mashups and Web-APIs using a bipartite graph, and found that while the growth rate of new Web-APIs and mashups is linear, the distribution of mashups over APIs follows a power-law. It uses network analysis approach to study both the usage patterns and the evolution traces of Web-APIs in the ProgrammableWeb. The analysis were conducted on the Composition-Service network, which is the same bipartite graph of Mashup-APIs and the Service-Service network, which is a network of services that are used together in the same mashups. It was confirmed that the service popularity distribution is highly concentrated, which is consistent with the findings in (Weiss & G.R, 2010), and that the reuse rate of services is low and the advanced use of many services together is still rare, which provides evidence to the motivation of building a social network for services/Web-APIs in this Thesis. Different methods for computing Web service similarity are also presented.

Chapter 4

Constructing Evolving Complex Networks for Web-API Discovery

Web-APIs are paving the way for a new generation of loosely-coupled and cross-organisational business applications over the Web. This is evident in the large-scale and ever-increasing number of Web-APIs currently available the Web (Yu, Liu, Bouguettaya & Medjahed, 2008; Adeleye et al., 2018). Many organisations like *Google*, *Facebook*, *Twitter* and *Amazon* are encapsulating various components of their services and functionalities as Web-APIs for other organisation to consumer. The continuous adoption of Web-APIs as means of facilitating cross-enterprise operations, and a cost effective way of creating value-added applications, have led to a rapid growth in the number and diversity of APIs available on the internet. This incessant growth of the number of Web-APIs coupled with their diversity makes their discovery and automated composition more challenging (Maamar et al., 2009; Sheng et al., 2014). Traditional methods of discovery, such as the usage of registries like UDDI and ebXML, have inherent limitations in that they only describe the functionality of one API and not how it interacts with others (Huang, Fan & Tan, 2014b; Hafsi et al., 2020; Maamar et al., 2009; Maamar, Wives et al., 2011; W. Chen et al., 2015a). Moreover, the complexity of

user's API requirements has made it even more difficult to find a single API that could satisfy the user's need.

Over the years, service compositions or *mashups* (Adeleye et al., 2018) have been used to handle complex users' API requests that cannot be satisfied by existing atomic Web-APIs. Based on the analysis of the Web-API ecosystem (Adeleye et al., 2018; Weiss & G.R, 2010; Huang et al., 2012a), it has been shown that only few APIs are frequently used or invoked in such composition. For instance, 50% of the total available composition in programmableWeb ecosystem only involve just 11 most popular Web-APIs and 10% of Web-API in the ecosystem are involved in any sort of composition (Adeleye et al., 2018; Botangen, 2020). This is another indication that the not so popular APIs are not been discovered. Even-though handful of Web-APIs are related based on their *co-occurrences* in mashups, there are a lot of APIs that are not involved in any mashup and therefore cannot be discovered via the social interactions or links in the ecosystem. At the moment, Web-API consumers including mashup developers normally turn to Web-APIs repositories such as *ProgrammableWeb* and *Mashapes* to discover API of their interest. However, most theses repositories are limited because they only rely on functional descriptions of APIs, and do not perform effectively with complex API requirements. Moreover, Web service registries like ProgrammableWeb and *Mashape*[1] consider Web-API ecosystem as *isolated* functional islands, where APIs are registered by diverse providers independently and progressively without considering relevant dynamic information or continuous social interactions that exist among the services which could influence their discovery. For instance, in ProgrammableWeb, Web-APIs have categories, and several Web-APIs can be involved in a mashup, however, there is no *direct* connection between two Web-APIs. The reason behind this is that Web-APIs in ProgrammableWeb registry are registered by diverse service providers independently over time, and the connections or social relationships

[1]https://rapidapi.com

between Web-APIs are never directly created or defined.

From a service consumer's perspective, if a user wants to create a mashup, the first step is to search the Web-APIs registry either by using functional descriptions of require API components or using generic descriptive service requests, and then manually sieve through the search result and select Web-APIs that meets the consumer's requirements. However, it is very challenging to sieve through large number of Web-APIs covering diverse functionalities and select suitable Web-APIs that match the exact consumer requirements especially when dealing with mashup-oriented user's requests. Existing research works in Web service discovery domain focused on how to retrieve a set of candidate Web-APIs which can satisfy specific user's request from the API registry. Most of these works rely on the API textual, functional descriptions, and adopted either the semantic-based or syntactic-based discovery approaches (N. Zhang et al., 2018). Because there are many semantically similar functionalities in the API registries, these methods are not sufficient to discover APIs with similar functional descriptions as the users' query. Relying solely on the keyword-based search approaches have been proven ineffective be insufficient to discover Web-APIs diverse functionalities (N. Zhang et al., 2018; W. Chen & Paik, 2013; Y. Wang et al., 2017). Another major downside of these approaches is that they totally ignore the contribution of Web-API social dimension, and only consider each Web-API as *isolated* functional islands, that does not interact with others in the ecosystem (W. Chen et al., 2015a; Metrouh & Mokhati, 2013; Fallatah et al., 2014; Maamar et al., 2009; Maamar, Wives et al., 2011; Hafsi et al., 2020).

To address these challenges, some recent works (W. Chen et al., 2015a; Adeleye, Yu, Yongchareon Yongchareon, Han & Sheng, 2020; Metrouh & Mokhati, 2013; Fallatah et al., 2014; Maamar et al., 2009; Maamar, Wives et al., 2011; Hafsi et al., 2020; Huang et al., 2014b) indicate that the inclusion of Web service social dimension in the discovery solution can help enhance their discoverability, and that capturing and keeping track of how APIs interact with one another may be beneficial in a variety of ways. These

works adopted network-based approach to capture the relationships among Web services and enhance service discovery. Web-APIs and social networks are gaining popularity, allowing service consumer to seamlessly and automatically search for and construct services based on the demands of users (Metrouh & Mokhati, 2013). This combination of two domains, social computing and service-oriented computing, allows for novel discovery techniques to emerge. It is the beginning of a new concept known as Social Web Services. Indeed, including social features into online services might help them become more active entities that can collaborate, compete, or replace one another (Hafsi et al., 2020).

However, most of the existing network-based approaches rely solely on Web-API invocation network data which is limited to a very few numbers of APIs (Adeleye et al., 2018). For example, authors in (Hafsi et al., 2020) used mashup-API bipartite graph for the discovery solution. Such solution is only limited to APIs that are invoked in mashups (detail explanation of the composition-service bipartite network is presented in section 4.2 of this chapter). Moreover, the network is static, and does not capture the true dynamic nature of Web-APIs (Huang et al., 2012a; Adeleye et al., 2018). Notable among this works is the work of W. Chen et al. (2015a), who attempted to use the *Bianconi-Barabási model* (Bianconi & Barabási, 2001b) complex network-model build a network of Web services. One limitation of their work is that the fitness parameter of an existing service node is calculated dynamically on the arrival of a new service, while the BB model requires a quenched/fixed fitness value for a node, which makes the closed-form solution of the BB model not applicable to this network (Adeleye et al., 2019). Other group of works (H. Wang et al., 2010; Fallatah et al., 2014; Feng et al., 2015) exploits semantic information and historical patterns to add links between services. For example, Wang et al. (H. Wang et al., 2010) utilize domain knowledge to compute the degree of semantic match between any pair of services and then set a threshold to determine the number of links in the network. Clearly such networks are

static without considering any dynamical properties. None of the existing works provide a clear cut theoretical approach/explanation of *"how to construct an evolving, network representation for web services"* that could be used to facilitate discovery applications.

In this research, it assumed that a Web service network should exhibit specific properties common to most real-world network-like systems such as World Wide Web and the Internet due to certain similarity these networks share with Web service ecosystems. For instance, just like most real-world networks, Web service ecosystems like ProgrammableWeb are not static but are dynamically evolving , and *grow* through addition/publication of new Web services. In addition, certain service like *Google Map* have high tendency to be invoked more often than others (i.e. socially, more popular)– an attribute that could either be likened to nodes with high degree centrality in real-world networks, and driving by popular natural phenomenon like *Preferential Attachment* (PA) observed in many real-world network systems (Barabási, 2016). Capturing network topological properties like the *geodesic distance*-based properties (including *small-worldness* and *Network navigability*) and the neighbourhood-based properties like *Transitivity* (clustering coefficient) that play unique roles in characterising real-world networks (Newman, 2003) could be beneficial to improving Web service discovery. Therefore, a typical evolving Web-API network is expected to meet (at minimum) he following requirements:

- Capture the underlining mechanisms that drive Web-APIs ecosystems. For instance, results of existing research works (Weiss & G.R, 2010; Lyu et al., 2014; Huang et al., 2012a; Adeleye et al., 2018) on the analysis of the Web service ecosystems indicate that popularity (captured by Preferential attachment mechanisms) is a key force of attractiveness that underlies service relationships or interactions in typical service ecosystem.

- Preserve Web-APIs social properties such as their popularity, similarity and

growth. Considering the influence of popularity, functional similarity and service evolution in their consumption and discovery, it is important that these properties are captured and preserved during the Web-API network simulation process.

- Incorporate other intrinsic properties of certain Web-APIs that propel them ahead of others in the ecosystem into its growing process.

- Have a network construction strategy that follows valid theoretical procedure and capable of replicating universal properties of real-world networks.

- Connecting all Web-APIs based on their social and functional attributes. A global social network for Web services should enable the incorporation of both functional and social proprieties of the services into their link formation process.

This chapter build on the results of the Web service ecosystem analysis presented in Chapter 3 and presents three key approaches for constructing evolving network for Web-API that enable connection of distributed API islands into a global Web-API social network. Each network construction follows specific theoretical procedure with clearly defined algorithm that enables the preservation of Web service system properties, and modelling of certain network properties that are ubiquitous to real-world social networks that could be exploited in Web service discovery. Instead of solely relying on the co-invocation or functional relationships (as they appear on some registries) to link web services , each network construction algorithm presented in this chapter leverages the underlying drivers of Web service ecosystems in its link formation process. The chapter presents: *(i)* Popularity-Based Web-API network which is a network construction based on the principle that popularity is attractive in Web service ecosystem. The network construction procedure is inspired by the Barabási-Albert complex network (BA) model (Barabási, 2016). *(ii)* Fitness-based Web-API network which uses random-walk to capture some intrinsic properties of Web-APIs called API *Fitness* and integrate this

information with the API's popularity information to facilitate the network growing and link formation processes. *(iii)* Popularity-Similarity based Web-API network, which exploits the trade-offs between Web-APIs functional similarity and the popularity of the API to define link formation between APIs. Instead of simply connecting service nodes based on their popularity, a balance s determined between two dimensions of attractiveness through their local optimization. In this case, while popularity attracts new connections, similarity is just as attractive. Nodes that are similar also have a higher chance of getting connected even if they are not popular. This network construction algorithm is fundamentally based on the Popularity-Similarity Optimization model introduced by authors in (Papadopoulos, Kitsak, Serrano, Boguná & Krioukov, 2012) .

The rest of this chapter is structured as follows: Section 4.1 presents the background and motivation for the approach used in this chapter. Section 4.2 elaborates on the limitation of mashup-API affiliation network. Section 4.3 presents the adapted complex network models used in this chapter to construct the evolving Web-API networks. Section 4.4 presents the construction of the Web-API networks. The section presents the conceptual and algorithmic descriptions of the networks, the node ordering strategy used in the construction process and the link formation procedures. Section 4.5 presents the network analysis of the networks. Detail comparison of the Web-API networks topological properties is presented. In addition, each network property is mapped with a typical discovery system properties. Section 4.6 concludes the chapter.

4.1 Background and Motivation

This section contains the background of the approach used in this chapter and the motivating elements of works. The section uses a motivating example to illustrate the drivers of some of the methodological decisions taken in this chapter.

4.1.1 Complex Network Theory Applications in Modelling Evolving Complex Systems

This research work is inspired by the recent developments in complex network theory. Over the past 20 years, complex networks have been extensively studied in the network science domain, and several significant discoveries have been made including the well-acclaimed small-world networks (Watts & Strogatz, 1998) and scale-free networks (Barabási & Albert, 1999). Network scientists have used generative network models to address two related tasks (Pham et al., 2015): (i) Studying the emergence of topological properties in complex networks and investigating possible mechanisms underlying the network formation. (ii) Modelling of dynamical processes involved in complex network systems in a way that enables effective exploitation of the known topological features.

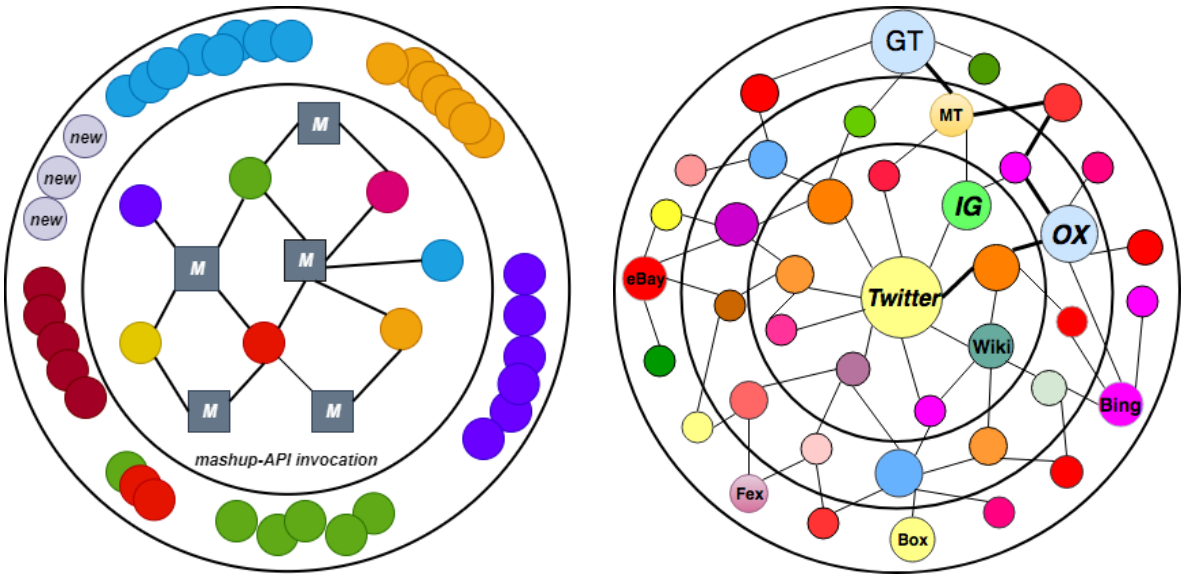
For the first tasks, various mechanisms that governs a network's topology and evolution have been investigated and are found ubiquitous among many real world networks. In particular, *preferential attachment* (PA) and *growth* have garnered special attention in evolving complex networks research (Barabási, 2016; Pham et al., 2016), not only because they are fundamental to explaining the topological features observed in many real world networks systems, but also because they have been empirically validated to be the drivers of many evolving networks. For instance, the topology of the Internet and the World Wide Web have been investigated using evolving network models and shown to be fundamentally governed by the PA and growth mechanisms (Albert et al., 1999; Barabási, 2016; Barabási, 2012).

For the second task, which involves modelling the dynamics of complex systems, several generative network models that exploit certain attraction mechanisms and can capture common topological features of complex networks have been proposed (Barabási, 2016; J. Sun, Qu, Chakrabarti & Faloutsos, 2005; Y. Cao, Wang, Jiang

& Han, 2006; Bianconi & Barabási, 2001a; Caldarelli, Capocci et al., 2002). The Barabási-Albert (BA) model (Barabási, 2016) is the most widely known PA-based evolving network model, and serve as the basis for several other models. The BA family of models generally exploit *popularity* as the main dimension of attractiveness which underlies PA phenomenon, and explain the emergence of scale-free structure characterised by heavy-tailed degree distributions commonly found in growing networks. However, other empirical observations universal to complex networks, particularly the small-worldliness of real world networks characterised by strong clustering and significant community structure are not captured or explained by BA model. For instance, in synthetic networks generated with BA, clustering is asymptotically zero (Barabási, 2016; Zuev, Boguná, Bianconi & Krioukov, 2015). In order to resolve the zero-clustering issue in BA, several extensions to the original BA model have been proposed, most notable of them all is the Bianconi-Barabási (BB) model (Bianconi & Barabási, 2001b). However, none of the extended models totally captured all the fundamental universal properties of complex networks. Recent works in network science (Papadopoulos et al., 2012; Alessandro & Vittorio, 2018; Zuev et al., 2015) that exploit latent network geometry coupled with PA of nodes addressed these shortcoming. Unlike BA model, these works show that not only *popularity* contribute to PA phenomenon but a combination of *popularity* and *similarity*. The popularity-similarity optimization complex network model introduced by (Papadopoulos et al., 2012) provides a natural geometric explanation for the limitations of BA model.

4.1.2 Motivation - From Isolated Web-APIs Functional Islands to Evolving Web-API Social Networks

Figure (4.1a) illustrates typical Web service ecosystem where very few APIs are logically linked based on invocation history in mashups and majority are isolated including



a b

Figure 4.1: Illustrative examples of Web-APIs connections and interactions in typical Web-API ecosystem.

the new services. Fig. (4.1b) illustrates a network representation that enables creation of direct connection among all services. Unlike (4.1a) where only functional relationship/categorisation (inform of tagging) exist, (4.1b) provides numerous potentials including functional and social. As illustrated in Fig. (4.1a), a typical service discovery registry like ProgrammableWeb considers Web-service ecosystem as *isolated* functional island with no *direct* link among Web services. Only very few services are logically connected through $mashup - API$ invocation. Historical API invocation information obtained from ProgrammableWeb portal and previous works (Huang et al., 2012a; Adeleye et al., 2018) reveal that the few APIs involved in such connections have enjoyed far more discoverability than the isolated ones, giving evidence to the importance of social connections to service discovery.Fig. (4.1b) illustrates how a complex network representation could help solve this problem, and indicate how network properties such as the highly connected hubs could be exploited to enhance service discovery. Given a Web-API ecosystem modeled as as network $\mathcal{G}$, where each API is a node with a predefined functional description, and each API is connected to one or more other

APIs based on : (i) its social attributes such as popularity (number of connections with others) , role and interactions with nodes in the network. (ii) its similarity with other nodes in the network including functional similarity and/or structural similarity. Based on this network, an intelligent *search* can be achieved by exploiting not only the functional descriptions (integrated as the node attribute) but also various network properties including topological and social properties underlying the construction of the network. For instance, specific real-world network properties such as *high degree distribution* with highly connected hubs, certain centrality measures like *closeness*, and small-world property can be exploited to identify important nodes in the network or help discover relevant clusters. Moreover, from user-interaction perspective, nodal and neighbourhood attributes can be utilized to facilitate effective network *navigation* (Boguna, Krioukov & Claffy, 2009).

From network application perspective, authors in (Maamar, Faci et al., 2011; Huang et al., 2014a; W. Chen & Paik, 2013) have demonstrated how Web service networks can be applied to improve their discoverability. Fig. (4.1b) shows an example of how the API network could be exploited. Consider the a new service consumer , who wants to leverage different Web-APIs from different domains (say Dictionary, Translation and Social) to create a mashup that allows users to find the meaning of a word in English, translate to French and post it on social media. Given the Web-API network $\mathcal{G}$, a user can query the network using the functional requirement description, and get the result in form of a *subgraph* (consist of the list of connected candidate Web-APIs) shown in Fig. (4.1b). The user can further surf or navigate through the *subgraph* and discover require or more interesting related Web-APIs. Further resourceful exploration can be perform by using *per-node* or *node-wise* attributes such *node degree centrality* to find most important node (could be most relevant API to the consumer's mashup request) e.g. *Twitter* in the *subgraph*. *Pair-wise* attributes like *Shortest Path Length* or *neighbourhood connectivity* (Y. Cao et al., 2006) can be exploited. For instance, the

consumer can *zoom-in* to the network, starting from *Twitter API* and then navigate by following optimal number of *hops* require to discover potential candidate APIs such as *Instagram (IG)* dictionary (OX), *Oxford dictionary (OX)* and *Google Translate(GT)* API that are either relevant to the consumer's mashup query or share certain feature with the original 'important' candidate *Twitter* . Such user activity pattern is very similar to surfing the WWW (just in our case the user is surfing the service network and would improve the chance of each node discoverability.

4.2 Limitation of Mashup-API Affiliation and *One-mode* Projection Networks.

As earlier discussed, the Web-API invocation network data is limited because it only contains very few numbers of services that are involved in mashup-APIs invocation relationship (Adeleye et al., 2018). In order to build a social network for Web services, a common, yet very limited approach used in many works (Huang et al., 2012a, 2014a; Lyu et al., 2014; Weiss & G.R, 2010) is to compress the *Mashup-API* affiliation network by applying one-mode projection (Zhou, Ren, Medo & Zhang, 2007) onto the API set to derive an *API-API* network (as illustrated in Fig. 4.2) . The limitation of this approach is apparent: *Only Web-APIs used in mashups (suppose every mashup contains at least two Web-APIs) will appear on the projected network.* For example, the ProgrammableWeb affiliation network contains only 1,525 Web-APIs, which is less than 10% of the total 16,959 Web-APIs on the registry. Furthermore, the popularity/number-of-links of a Web-API node on the projected network is discounted as there are mashups that use only one Web-API, and such links are not counted in the projected network (for example the link between M_5 and A_6 in Fig. 4.2). Eventhough some existing network-based discovery applications used bipartite network/ one-mode projected networks, issues

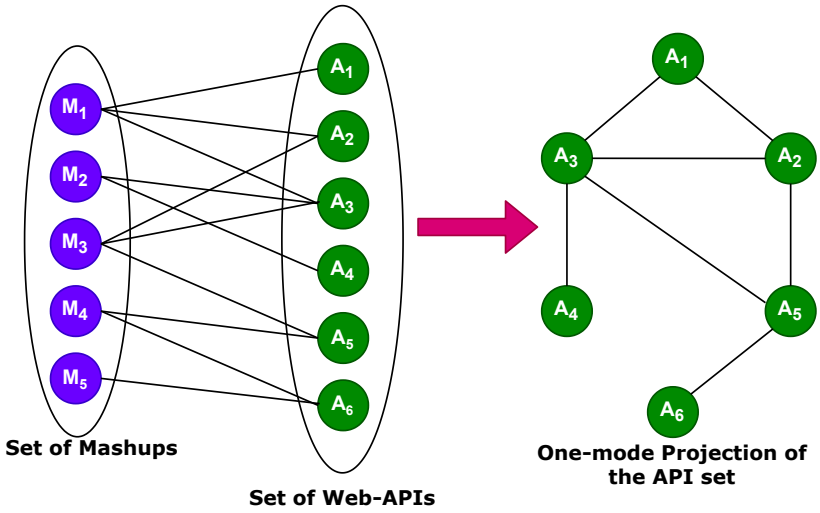


Figure 4.2: Illustration of the Mashup-API bipartite graph (left), Projected Network of API with respect to mashups (right).

such as node *isolation* and loss of crucial information are major setbacks to the success these applications. Therefore, it is important to explore network modelling approach for capturing the properties of the Web service ecosystem so that the network can be sufficient enough to facilitate Web-API supporting applications like . This research work based the network construction algorithms on the theoretical procedures of three evolving network models described in Section 4.3.

4.3 Evolving Complex Network Models

Many social, technological, biological and economical systems are best described using evolving complex network models (Barabási, 2016; Y. Cao et al., 2006). Generally, most real-world network systems evolve either by addition or removal of links or nodes over time, and they share common characteristic such as *small-worldness*, *high-clustering coefficient*, *community structure* and *heavy-tail* in degree distribution . Unlike other types of networks especially static networks which are time independent, evolving networks are time-varying and grow dynamically. That is, links and nodes

are continuously added over period of time. Among several mechanisms that govern the evolution of real networks and determine their topology, *growth* and *preferential attachment* have garnered more attention (Pham et al., 2016), and have been empirically validated in many real world networks such as the WWW (Barabási, 2016), the internet (Albert et al., 1999) and citation networks (Barabási, 2012). Various generative network models (Easley & Kleinberg, 2010; Barabási, 2016; Dorogovtsev & Mendes, 2002) have been proposed to investigate the underlying mechanisms that define the topology of real-world complex networks. Out of these models, the Barabási-Albert evolving network model (Barabási, 2016) have been the most widely-known *preferential attachment-based* model mainly due to its simplicity. The preferential attachment mechanism in BA model dictates that its principle of connection to any pre-existing node will not be entirely random but linearly biased by the number of connection (node degree) of the pre-existing nodes.

Some variations of *preferential attachment-based* complex network model (Choromański, Matuszak & Mikisz, 2013; Barabási & Albert, 1999; Hébert-Dufresne, Allard, Marceau, Noël & Dubé, 2011; Klemm & Eguiluz, 2002; Vázquez, 2003) have emerged in recent time including the joint preferential and fitness based model (Bianconi & Barabási, 2001b; Caldarelli, Capocci et al., 2002). While preferential attachment may be a valid assumption in many circumstances, it is not in some others. In some cases, newly added nodes do not have access to information on the degree of each and every node, either directly or indirectly. Instead, it is reasonable to connect two vertices if the relationship provides a mutual advantage (in the case of bidirectional or undirected edges) based on some of their inherent qualities such as correlation, similarity or difference, friendship, social success, scientific significance, interaction strength, and so on (Caldarelli, Capocci et al., 2002; Barabási, 2016). As a result, it's plausible to assume that scale-free behaviour (when it exists) in some of these network-based systems has a source unrelated to preferential attachment behaviour. To explore this assumption, the

concept of *node-importance* or *fitness* was introduced by some network scientists in complex network as an additional ingredient on top of the preferential attachment-based, BA network with more emphasis on the node's fitness (Barabási, 2016; Caldarelli, Capocc et al., 2002; Caldarelli, Capocci et al., 2002; Pham et al., 2015; Adeleye et al., 2019). In this case, each node acquires new links with the probability proportional to a time-independent fitness value which quantifies the intrinsic excellence of the node. While the preferential mechanism emphasis the *rich-get-richer* phenomenon, the fitness mechanism is often referred to as *fit-get-richer* (Barabási, 2016; Caldarelli, Capocc et al., 2002).

Recent advances in network geometry showed that global topological properties observed in real-world complex networks can emerge from *hyperbolic space* (Krioukov, Papadopoulos, Kitsak, Vahdat & Boguná, 2010; Muscoloni & Cannistraci, 2018; Alessandro & Vittorio, 2018; Papadopoulos et al., 2012; Zuev et al., 2015), and that preferential attachment mechanism is not entirely a degree-based event as demonstrated in BA model but rooted in an optimization framework that exploits the trade-off between node popularity and similarity (Papadopoulos et al., 2012; Zuev et al., 2015). The classical preferential-attachment based generative models like the BA model mainly capture the scale-free property of but failed to explain the emergence of community structure and strong clustering coefficient in real-world networks. In contrast to this, the *popularity-similarity-based* model is able to capture all the three universal, fundamental properties of the networks. The following sections presents the theoretical details of the three network models (*popularity-similarity* optimization, *fitness-based* and the *preferential attachment-based* evolving network models) that the Web-API network constructions algorithms are based on.

4.3.1 Popularity-Based Network Model

As previously discussed, the Barabási-Albert complex network model is the most popular *popularity-based* network generative model defined by a simple form of preferential connectivity in which the likelihood that a vertex v_i of degree $k_i(t) = k$ acquires a link at time-step t is defined to be proportional to a *time dependent function* $A_k = k$. A_k is referred to as the *Preferential Attachment function* (Pham et al., 2016). The model involves the two generic aspects: (i) Growth Aspect (ii) Preferential Attachment Aspect. These aspects are described in the following section:

Growth Aspect

For the *growth* aspect, it involves continuous addition of new nodes (Web-APIs) into an open network. Therefore, the number of nodes N in the network increases throughout its life span. To achieve this, first, the network is initialized, starting with fully connected m_0 number of nodes. At each time step, a new node (Web-API) A_j with m links is added to the network.

Preferential Attachment (PA) :

As described min Chapter 3, the PA describes the probability $\Pi(i)$ that a newly arrived link of a new node connects to existing node i as proportional to the node $i's$ degree (k_i). PA is dynamically measured following similar description in section 3.2.2. A new node is free to connect to any node in the network. However, in equation 3.5, the dependence of Π_i on k_i implies that higher-degree node have more visibility. For instance, if a new node has a choice between degree-two and degree-six node, it is three-times as likely that it links with the degree-six node.

Algorithm 4.1 described the inputs and step-by-step procedure of BA model. $Line2 - 5$ of the procedure captures the two aspects. $Line - 2$ shows the dynamic

Algorithm 4.1: BA Network Generative Model

Input:

1: N: total number of nodes
2: m_0: number of initial nodes
3: m: number of edges added at each time step

Output:

1: G_N: the preferential-attachment-based network

Procedure

1: Initialize G_0 by completely connected network with m_0 number of nodes
2: **for** t = 1 to $N - m_0$ **do :**
3: $\Pi(k(t)) \leftarrow A_k(G_{t-1})$
4: $G_t \leftarrow SELECT_{neighbors}(G_{t-1},\ \Pi(k(t)), m)$
5: **end for**
6: **return** G_N

growth process for N numbers of nodes. At each time step t, a new node with m number of edges is added to the network G_t and connected to m number of already existing nodes in the network, where $m \leq m_0$. In $Line-3$, PA is dynamically computed for each node in the network G_{t-1} with the function A_k based on equation 3.5 which returns probability distribution $\Pi(k(t))$. Finally, a selection function $SELECT_{neighbors}$ in $Line-4$ is used to select the m neighbours in G_{t-1} based on the distribution which the new node will connect to. After all N nodes join the network, the BA network G_N is obtained.

4.3.2 Fitness-Based Network Model

In reality, there are other factors other than age and node degree which influence how nodes in a network acquire links. Web service, web-pages, actors (Barabási, 2016)(M. G. H. Bell et al., 2017) all posses some inherent qualities that influences the rate at which they can attract nodes or acquire links. For instance , In the World

Wide Web, search engines such as Alta Vista, Inktomi started earlier and dominated the market before the arrival of Google. However, Google soon overtook these search engines in terms of number of connections and became the leading search engine with the largest hub of the Web. In similar fashion, Facebook took over from Google in 2011 as the biggest hub on the web. This shows that some *latecomers* can actually acquire links relatively quickly and others who arrived earlier may not make it. The *Bianconi-Barabási* (BB) model (Bianconi & Barabási, 2001b, 2001a), a variant of BA model, captures this behaviour by having on top of *growth* and *PA* another concept called the *fitness*. The model explains that age is not the best predictor of a node's success, rather latecomers with high fitness values also have the chance to attract links to form hubs.

The BA model assumes that a node's growth rate is determined solely by its degree k. In order to incorporate the role of fitness, BB model consider preferential attachment to be driven by the product of a node's fitness η, and its degree k. Both BA and BB models are known for their plausible explanation for the emergence of *scale-free* topology characterized by heavy-tail degree distribution in real world networks. While the BA model explains the *"first mover advantage"* phenomenon, the BB model explains how latecomers can become hubs. In a fitness-driven network, a node with higher fitness will acquire links at a higher rate than less fit nodes. The fitness values are assigned to each node. The value embodies all the properties other than the degree (or popularity). The higher the fitness, the higher the probability of attracting new edges. Fitness here is a quantitative measure of a node's ability to stay in front of the competition (Barabási, 2016). The model consists of the following two steps :

Growth:

Starting with a small number of node (m_0), at every time step, a new node j with m links and fitness η_j is added to the network and is connected to $m \leq m_0$ nodes

already-existing in the network, where η_j is a random number chosen from a fitness distribution $\rho(\eta)$. Once the fitness value is assigned, a node's fitness remains unchanged. Random-walk with restart approach is adopted in this work to estimate the fitness value of API nodes. Detail of this is reported in Section 4.4.3.

Preferential Attachment (PA):

The probability $\Pi(i)$ that a link of the new node connects to node i depends on the product of node $i's$ degree (k_i) and its fitness (Barabási, 2016) :

$$\Pi(i) = \frac{\eta_i k_i}{\sum_j \eta_j k_j} \tag{4.1}$$

A new node is free to connect to any node in the network. However, in equation 3.5, the dependence of Π_i on k_i implies that higher-degree node have more visibility. For instance, if a new node has a choice between degree-two and degree-six node, it is three-times as likely that it links with the degree-six node. Moreover, the dependence of Π_i on η_i captures the fact that between two nodes with same degree, the node with higher fitness value is selected with a higher probability. Therefore, a relatively new node with initially few connections, can acquire links rapidly if such node has larger fitness value compare to other nodes.

Using continuum theory to predict the temporal evolution of each node (Barabási, 2016), according to equation 3.5, the degree of node i changes at the rate :

$$\frac{\delta k_i}{\delta t} = m \frac{\eta_i k_i}{\sum_j \eta_j k_j} \tag{4.2}$$

The factor m appears because each new node is expected to add m links to the network each time. To simplify equation 4.2, it is assumed that the time evolution of k_i follows

power-law with an exponent $\beta(\eta_i)$ that is dependent on the fitness (η_i):

$$k(t, t_i, \eta_i) = m(\frac{t}{t_i})^{\beta(\eta_i)} \tag{4.3}$$

where t_i represent the time at which node i was introduced to the network. Plugging equation 4.3 into equation 4.2, the dynamic exponent $\beta(\eta)$ of BB-model satisfies:

$$\beta(\eta) = \frac{\eta}{C} \tag{4.4}$$

where C is :

$$C = \int d\eta \rho(\eta) \frac{\eta}{1 - \beta(\eta)} \tag{4.5}$$

Equation 4.4 shows that the dynamic exponent in BB-model is proportional to the node's fitness η . Hence, given sufficient time t, a node's with higher fitness will increase its degree faster and leave behind nodes with smaller fitnesses over time. The *degree distribution* $p(k)$ given as

$$p(k) \approx C \int d\eta \frac{\rho(\eta)}{\eta} \left(\frac{m}{k}\right)^{\frac{C}{\eta}+1} \tag{4.6}$$

Equation 4.6 is a weighted sum of multiple power-laws, which implies that pk is dependent on the exact form of the fitness distribution,$(\rho\eta)$ (Barabási, 2016).

4.3.3 Popularity-Similarity Optimization Network Model

The *Popularity-Similarity Optimization* – PSO evolving network model introduced in (Papadopoulos et al., 2012) is used to specify the growth of networks in an hyperbolic space. The model sustains that topological properties like *high clustering* and *scale-free* degree distribution commonly found in most real-networks representing complex systems are the result of an optimisation process in which nodes seek to form ties or

connect, not only with the *popular* (most connected) system components (nodes), but also with the components *similar* to them. The model has a geometric interpretation in which network evolves by optimizing certain trade-offs between *node popularity* abstracted by the *radial coordinate* in hyperbolic space and *node similarity* represented by the *angular coordinate distance*.

The components of the model is described as follows:

Input Parameters: In general, the model includes four input parameters: (i) N : total number nodes in the network ; (ii) m: parameter controlling the average node degree $\bar{k}$ = $2m$ (iii) β : popularity fading parameter, $\beta \in (0, 1]$. (iv) $T \geq 0$: network temperature, which controls the network clustering, where the network clustering is maximized at T = 0, and it decreases almost linearly for T = $[0, 1)$ and it remains asymptotically 0 for any $T \geq 1$;

Growth Aspect : In PSO model, node *birth time* is used as the proxy for popularity such that nodes with the earliest *birth time* have more chances to attract new links and become popular. If nodes join the network one at a time, then the birth time is simply the node number t = $1, 2, 3, \ldots N$, where N is the total number of nodes. For similarity, nodes were randomly placed in the geometrical space, and the angular distances between the nodes in the space model their similarities.

The network is dynamically generated in the following steps: (i) Start with an initially empty network ; (ii) at each time step $t > 1$, add new node t at a random angular position θ_t on the geometrical space (circle) ; and (iii) connect new node t to a subset of existing nodes s, where $s < t$ and consist of the m previous nodes with the m smallest values of product $s\theta_{st}$. m is the parameter controlling the average node degree k = $2m$, and θ_{st} is the angular distance between nodes s and t. At early times $t \leq m$, node t connects to all the existing nodes. Unlike PA-based model, where new node connects with the same probability $\Pi(k)$ to any nodes of $k - degree$ already existing in the

network, in the PSO model, new node only connects to a specific *subsets* of $k - degree$ nodes in the network that are closest to the new node along the similarity dimension θ.

If T is set to 0, the new node t connects to the m hyperbolically closest nodes, if $T > 0$, the new node t randomly selects existing node $s < t$, and given that t is not already connected to the s, t connects to s with probability $p(st)$;

$$p(st) = \frac{1}{1 + exp(\frac{x_{st} - R_t}{2T})} \tag{4.7}$$

where R_t is the current radius of the hyperbolic disk, defined in equation 4.8 and x_{st} is the hyperbolic distance between node s and node t defined in equation 4.9, and θ_{st} is the angle between node s and t.

$$R_t = r_t - 2\ln\left[\frac{2T(1 - e^{-(1-\beta)\ln(t)})}{\sin(T\pi)m(1-\beta)}\right] \tag{4.8}$$

$$x_{st} = \operatorname{arcosh}(\cosh r_t \cosh r_s - \sinh r_t \sinh r_s \cos\theta_{st}) \tag{4.9}$$

This procedure will be repeated until t becomes connected to m nodes. The growing process will stop after all the N nodes have joined the network.

Geometric Interpretation of PSO Model: As mentioned above, PSO model has a geometric interpretation in which popularity preference emerges from local optimization (Papadopoulos et al., 2012). Specifically, the network growing task is transformed such that a new radial coordinate system $r_t = \ln(t)$ which maps the birth time t of a node to its radial coordinate r_t is defined. Consequently, all nodes are positioned on a plane with their associated polar coordinate (r_t, θ_t) (the distance becomes log scale). Then, new nodes are simply connected to the closest m nodes on the plane , where distances are hyperbolic. The *hyperbolic distance* x_{st} between any two nodes at polar coordinates (r_s, θ_s) and (r_t, θ_t) is computed as $x_{st} = r_s + r_t + \ln(\frac{\theta_{st}}{2}) = \ln(st\frac{\theta_{st}}{2})$.

Therefore, minimizing $x_s t$ becomes equivalent to minimizing $s\frac{\theta_{st}}{2}$ when dealing with the connectivity from a new node t. In essence, the trade-off between popularity and similarity is abstracted as the minimization of distance between two nodes on the hyperbolic plane.

Preferential Attachment and Popularity Fading in PSO Model:

In PSO model, *preferential attachment* phenomenon emerges from an optimization framework in which new connections optimize certain trade-offs between popularity and similarity, rather than simply preferring popular nodes as in PA-based model. A new node will connect to the node that is most similar to it but also has the largest degree (most popular). In this case, both popularity and similarity contribute to the PA emergence, and are both key dimensions of attractiveness. The authors (Papadopoulos et al., 2012) have shown that the probability $\Pi(k)$ is thesame linear function of k in the model as in PA.

PSO can also model *popularity fading*, another phenomenon found in many real network. Specifically, in many network , early nodes are usually more popular, however, over time, the popularity of these nodes will continue to decrease or fade. For example, in the World Wide Web, search engines such as Alta Vista, Inktomi started earlier (early birth-time) and dominated the market before the arrival of Google search engine. However, Google later became more popular (higher connection or degree) and dominated the market, while the popularity of the earlier search engines continue to fade with time. PSO modelled popularity fading by letting nodes drift away from the centre of the hyperbolic space such that radial coordinate of node s at time $t > s$ is increasing as $r_s(t) = \beta r_s + (1 - \beta) r_t$, where $r_s = \ln(s)$ (an old node and its drifts) and $r_t = \ln t$ (the current node at time t), and hyper parameter $\beta \in [0, 1]$.

As β approaches 1, the nodes becomes stationary and no drifting is allowed. On the other hand, as β approaches 0 , all the nodes move to the circle of radius r_t, reducing

the network to a random geometric network growing on the circle. Another view of this fading is that the attraction probability $\Pi(k)$ power law exponent γ changes to $\gamma = 1 + \frac{1}{\beta} \geq 2$. Preferential attachment emerges at any $\gamma = 1 + \frac{1}{\beta}$ since $\Pi(k)$ is a linear function of degree k, $\Pi(k) \propto k + m(\gamma - 2)$ - similar to PA (Papadopoulos et al., 2012).

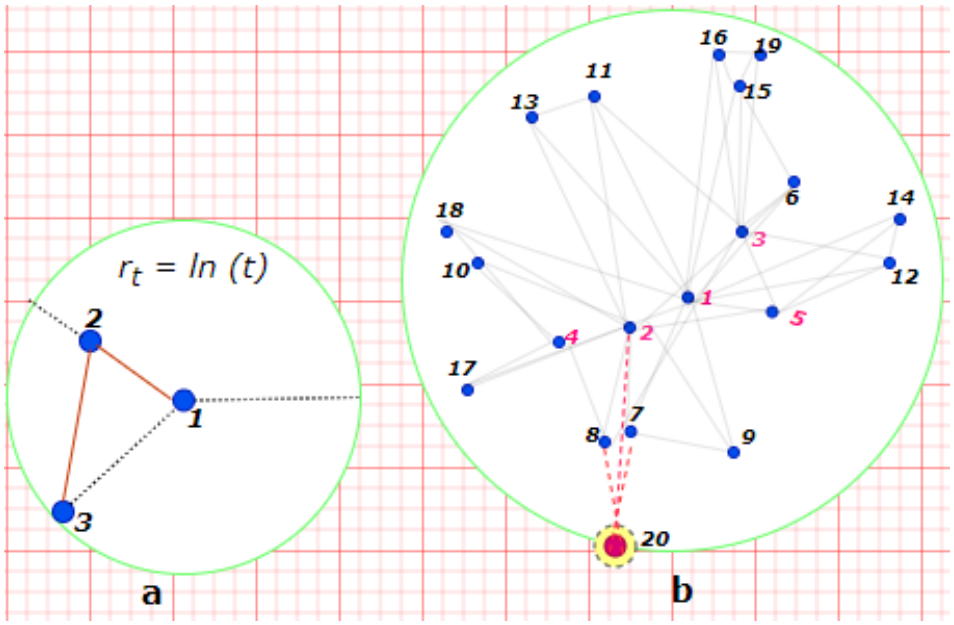


Figure 4.3: Placing node s in the network using polar coordinates at (r_s, θ_s), where $r_s = \ln s$, and θ remains the normalized Web-API functional similarity.

Figure 4.3(a) shows the transformation to new radial coordinate system where distance becomes log scale. The short *hyperbolic distance* between the new node t and the existing node s is approximated as :$x_{st} = r_s + r_t + 2\ln(\frac{\theta_{st}}{2})$, where θ_{st} is the angle separating node t from s, which is calculated as $\theta_{st} = \pi - |\pi - |\theta_s - \theta_t||$. x_{st} correspond to the probability of ***link formation***. Minimizing x_{st} becomes equivalent to minimizing $s\theta_{st}$ during link formation process between a new node to existing ones. The competition between popularity and similarity is simply the minimization of distance between the two node points on the hyperbolic space. Figure 4.3(b) show sample PSO network with $m = 3$ and $N = 20$, new node $t = 20$ with radial coordinate $r = \ln(t)$ connected to three hyperbolically closest node

4.4 Constructing Evolving Web-APIs Social Networks

This section presents the construction of the proposed evolving, Web-API complex-networks based on the theoretical procedures of BA, BB and PSO complex network models. First the node ordering strategy used for the network construction is discussed, then, the algorithms for constructing the Web-API network including network growing procedures and node ordering strategies are discussed.

4.4.1 Node Ordering Strategy

In order to preserve the popularity information of the existing Web-APIs, a node ordering strategy that enables our network to captured API popularity information similar to their popularity in their ecosystem is defined. For a growing network, there are two common ways to decide which node gets into the network first, such that the social properties like *popularity* are well preserved; One way is to use the node birth time (like the publication date of Web-APIs), this is because, in principle and without any external influence, older nodes have more chances to become popular and attracted (Papadopoulos et al., 2012). Another and more direct proxy is to use the node degree (Albert & Barabási, 2002), provided the information is available. For the later, the node with higher degree are introduced into the network before the lower degree nodes. For the Web-API network construction, a mix of both node ordering approaches is used to order the Web-APIs. Numbers are then assigned to each API based on its positions in the ordered node-list. Apart from preserving the popularity information of the Web-APIs in their ecosystem, another reason for using both ordering strategies is that, for Web-APIs with no degree information (that is, APIs that have never been consumed or invoked in any mashup), the publication dates of these APIs as specified in the original dataset are used to decide when they enter the network. Since, popularity is one of the key drivers of our networks, API nodes with degree information are first

added to the Web-API node-list, then followed by the other with no degree information, which are ordered by publication date.

4.4.2 Strategy For Constructing Preferential Attachment-Based Web-API Network

This section presents the construction of the *Popularity-based* Web-API network based on the theoretical procedure of the Barabási-Albert evolving network model (Barabási, 2016). As discussed in Section 4.3.1, the combination of *growth* and *PA* are the two generic mechanisms that drive many real-world networks, and the presence of both growth and PA in the ProgrammableWeb affiliation network have been validated in Chapter 3. Based on these findings, the focus here is to construct an evolving network of Web-APIs that preserve both the topological properties of Web-API affiliation network including the popularity information of the Web-API nodes, while including all the Web-APIs in the ecosystems.

Growth Aspect

For the *growth* aspect, it involves continuous addition of new nodes (Web-APIs) into an open network, therefore increasing the number of nodes in the network throughout its life span. In order to achieve this, first, the network is initialized, starting with fully connected m_0 number of nodes. At every time step , a new node is added with m number of edges.

Preferential Attachment Aspect

In the PA-based Web-API network, the probability that new node A_j is connected to an existing node A_i already in the network is neither uniform nor random but depends on the degree k_i of node A_i. To incorporate *PA*, the probability that a link of the new node

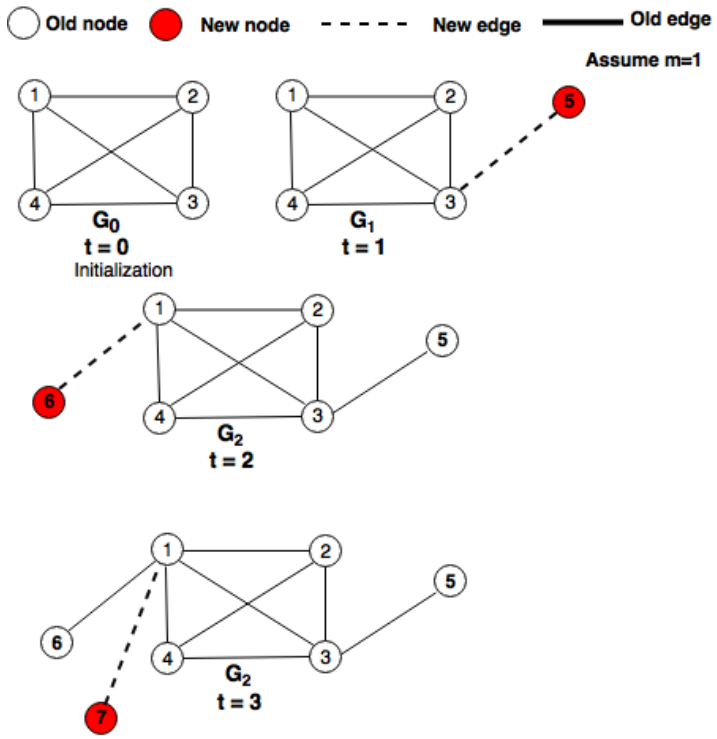


Figure 4.4: Growth-PA illustration in BA network growth procedure

A_j connects to an existing node A_i based on A_i's degree k_i is dynamically computed.

Figure 4.4 illustrates the growth and Preferential attachment processes: Given $m_0 = 4, m = 1$, at time 0, 4 fully connected nodes form the initial network; at time 1, *node 5* joins the network; based on PA, as *node 1-4* each has the same degree, each of them has the same probability to attract *node 5* to connect to it, which is 1/4; at time 2, based on PA, the probabilities for *node 1-5* to attract the new node are $[\frac{3}{K}, \frac{3}{K}, \frac{4}{K}, \frac{3}{K}, \frac{1}{K}]$ where $K = 13$ is the total degree of the network and the numerator is the degree of each existing node. In this case, older nodes in the open network always have better chance to attract edges than newer nodes because they have better degree (popularity).

The node-ordering strategy discussed in 4.4.1 (specifically tfor ProgrammableWeb) is used to preserve the popularity information in the affiliation network. it indicate *when* a Web-API will join the growing network. The full list of Web-APIs in the original dataset is sorted, and then each API is introduced into the network one-by-one (or

step-by-step) based on their position in the ordered node-list.

A network growth strategy is defined for this as follow:

Growth Strategy for Web-API network

- First, the Web-API nodes is sorted based on their degree (or popularity) in the affiliation network in a descending order so that higher-degree nodes are in the front to produce L_1;
- Then, for Web-APIs that do not appear in the affiliation network ,they are sorted based on their date of publication/birth in an ascending order so that older nodes are in the front to produce L_2; (this is similar to the *date-of-birth* based ordering strategy used in (Papadopoulos et al., 2012) and discussed in Section 4.4.1.)
- Finally, L_2 is appended in the end of L_1, and each node is introduced into the network one-by-one based on their order in the list.

Algorithm 4.2 describes the complete construction procedure of the BA-based Web-API evolving network. $Lines\ 1-7$ capture the procedure for creating API *node-list* used for populating the network (as described above). $Sort_{desc}$ takes the nodes with their degrees ($nodeDegree$ dictionary) as keys and values respectively, and sort the nodes by their degrees in descending order to get L_1. The *date-of-birth* (same as published date) for all APIs in ProgrammableWeb is acquired. In $Lines\ 3-4$, function $getDoB$ takes the DoB information and the names of APIs with no degree information, then returns a dictionary $isolatedNode$ (isolated nodes from the affiliation network) with the node's $DoBs$ as values. $Sort_{asc}$ takes the $isolatedNode$ and sort in ascending order based on the DoB to get $L2$. $Line\ 8-12$ similar to $Line\ 1-5$ of Algorithm 4.1.

An overview of the constructed ProgrammableWeb API network containing all its Web-APIs is shown in Fig. 4.5, with popular nodes labelled. Next, the Fitness-based

Algorithm 4.2: BA-Based Web-API Network Model

Input:

N: total number of API nodes
m_0: number of initial nodes
m: number of edges added at each time step
G_{af}: Affiliation network
$node_{DoB}$: *date-of-birth* for all APIs

Output:

G_{ban}: the BA-based Web-API complex network

Procedure

$nodeDegree \leftarrow G_{af}.getDegree$
$L_1 \leftarrow Sort_{desc}$(nodeDegree.$keys()$, $degree$)
for i in N **and** i **not** in L_1 **do :**
    $isolateNode[i] \leftarrow getDoB(node_{DoB}, i)$
    $L_2 \leftarrow Sort_{asc}(isolateNode.keys(), DoB)$
end for
$L_1.\ append(L_2)$
Initialize G_0 by completely connected network with m_0 number of nodes on top of L_1 list
for $t = m_0$ to $L_1.getLenght - m_0$ **do :**
    $\Pi(k(t)) \leftarrow A_k(G_{t-1})$
    $G_t \leftarrow SELECT_{neighbors}(G_{t-1},\ \Pi(k(t)), m)$
end for
return G_{ban}

Web-API evolving network is considered, this model unifies the PA and fitness in a single model.

4.4.3 Constructing Fitness-Based Evolving Web-API Network

One of the main driver of our proposed Web-APIs network is the *fitness* of APIs, which represent certain intrinsic property that propel some APIs ahead of others. This section presents the processes involved in constructing the fitness-based Web-API network. First, the process for estimating Web-APIs fitness using random walk algorithm is

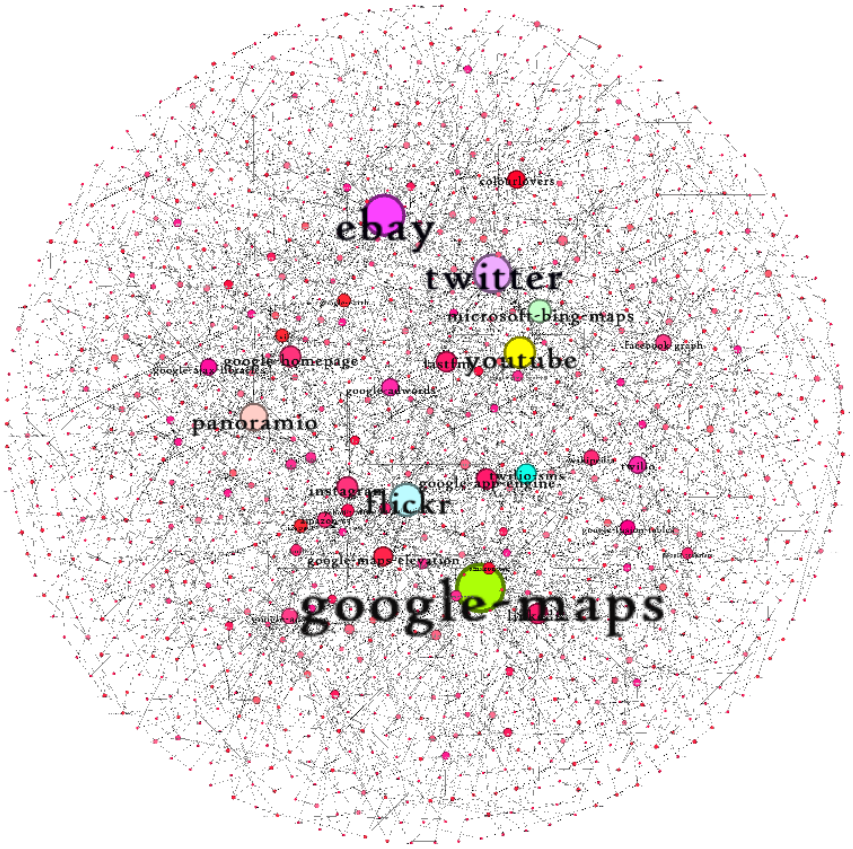


Figure 4.5: Overview of the BA-Based Web-API Evolving network

discussed. Then, the strategy and procedure used in constructing the network using *ProgrammableWeb* dataset. As mentioned earlier, the network growing procedure is based on the *BB model* (Bianconi & Barabási, 2001b) described in section 4.3.2 . Details of the network construction procedure is described in the following sections:

Web-APIs Fitness Value Estimation using Random Walk with Restart

Web-API fitness values is estimated using Random Walk with Restart (RWR) (Tong, Faloutsos & Pan, 2006). RWR offers a strong relevance score between two nodes in a weighted network and has been effectively utilised in a variety of situations, including automated captioning of pictures, extensions to "connection subgraphs," customised PageRank, and many more. The fitness value of a given API is estimated as the expected

value of its correlation with other APIs in the service ecosystem. The RWR approach works such that if a random particle is that starts from node *i*. The particle iteratively transmits to its neighbourhood with the probability that is proportional to their edge weights. At each step, the particle has some probability *c* to return to the node *i*. The affinity score of node *j* with respect to *i* is defined as the steady-state probability $r_{i,j}$ that the particle will finally stay at node j (Tong et al., 2006). The estimation of Web-APIs fitness values involves three key steps: (i) Construct an interaction graph based on Mashup-API relationship (ii) Correlation Inference (iii) Finding the mean Random walk score for each API

Intuitively, the RWR algorithm computes a relevance score $r\ (A_i, A_j)$ between two APIs nodes A_i and A_j in an interaction network, base on the likelihood that random walk through the network starting at A_i will ends at A_j.

For *N* number of nodes in the interactive network, the *N-by-1* steady-state probability vector $\overrightarrow{p_i}$, which contains the RWR scores of all API nodes with respect to node i, satisfies the equation (4.10) below:

Where W is the column-normalized adjacency matrix of the interactive network, c is the restart probability of the RWR from node i and $\overrightarrow{e_i}$ is the N-by-1 restarting vector, the i_{th} element 1 and 0 for others. We further explain the detail procedure for RWR fitness estimation in sect

Given an interactive graph G defined as G=$(M_k \cup A_n, E)$, where M_k is the set of Mashups of size k and A_n is the set of Web-APIs of size n , and for any edge $(m, a) \in E$, $a \in A$ and $m \in M$. M_k=$\{M_i|1 \leq i \leq k\}$ and A_n=$\{A_i|1 \leq i \leq n\}$. We compute the relevance score r_i between pairs elements in G using RWR, which is formally defined in equation 4.10 . Such that, for every $M_k(A_n)$!=0 in G, a steady-state probability r of size $(k + n) * n$ is computed.

$$\overrightarrow{r_i} = (1 - c)W\ \overrightarrow{r_i} + c\ \overrightarrow{e_i} \tag{4.10}$$

Where $r_i \in^{Nx1}$, and $W \in^{NxN}$ is a weighted transition matrix obtained based on graph G , and $e_i \in^{Nx1}$ is the restarting vector with i_{th} entry set to 1 and all other entries are 0. c is the restart probability of the random walker from node i .

Equation 4.10 can be further simplify (Yao, Sheng, Ngu, Li & Benatallah, 2015) as:

$$\overrightarrow{r_i} = c(I - (1-c)W_m)^{-1}e_i = Qe_i \tag{4.11}$$

where I denotes an identity matrix and W_m is obtained by row-normalizing the weighted transition matrix W using equation 4.12:

$$W_m = WD_m^{-1} \tag{4.12}$$

where D_m is a diagonal matrix with $D_m(i,i) = \sum_j W(i,j)$. The random walker iteratively transits to other API nodes that have edges with the starting node A_i in the network, with the probability proportional to the edge weight between them. At each time step, A_i has a restart probability c to return to itself.

From equation 4.11 , $Q = c(I - (1-c)W_m)^{-1} = c\sum_{t=0}^{\infty}(1-c)^t W_m^t$, which defines all the steady-state probabilities of API node A_i transiting other API nodes when the random walk process is converged. W_m^t represents the t_{th} order transition matrix with elements w_{ij}^t interpreted as the total probability for the random walker that starts at node i (API node A_i) and ends at node j (API node A_j) after t iteration, traversing all possible paths between i and j. When the random walker reach a convergence threshold C_{th}, the steady-state probabilities for each pair of API nodes is obtained . Figure 4.6 shows a sample result of RWR estimated relevance score for 5 nodes. The resulting probability values represent the relevance scores of pair elements in graph G i.e. the *long-term* visiting rate from a given node to any other node in G . We set $C_{th} = 10^{-6}$ for this experiment .

	A1	A2	A3	A4	A5
A1	0	0.0159	0.2511	0.0159	0.2511
A2	0.0153	0	0.00022	0.00002	0.7423
A3	0.6457	0.00056	0	0.09835	0.00601
A4	0.01527	0.00002	0.7423	0	0.00022
A5	0.3238	0.09835	0.00601	0.00057	0

Figure 4.6: Example of fitness-Values of A_i wrt A_j using RWR

.

Finally, to get the fitness value for each API, the *expected value* of the relevance score for each API node is computed, which represent the *long-run* average of each API's r_i, where $r_i \in^{Nx1}$. Then, the resulting value is assigned as the fitness of each corresponding APIs.

The plausibility of the resulting fitness distribution is then examined using the Kolmogorov-Smirnov (KS) distance test in (Clauset et al., 2009). The test enables the generation of p-value between 0 and 1 according to differentiation of our data and the synthetic model. If the p-value is closer to 1, then it can be ascertained that the data is more plausible to a given distribution. Bootstrapping approach is then used to obtain the p-value=0.0325 and p-value=0.525 for the expected-similarity data points fitted to power-law and log-normal respectively. This shows that the distribution is not exceptionally plausible as power-law distribution but plausible for log-normal distribution. Figure 4.7 shows plot of the distribution of Web-APIs fitness on a log-linear plot.

Link formation and network grow process in the fitness-based network construction involves three generic aspects: (i) Growth Aspect (ii) Preferential Attachment Aspect

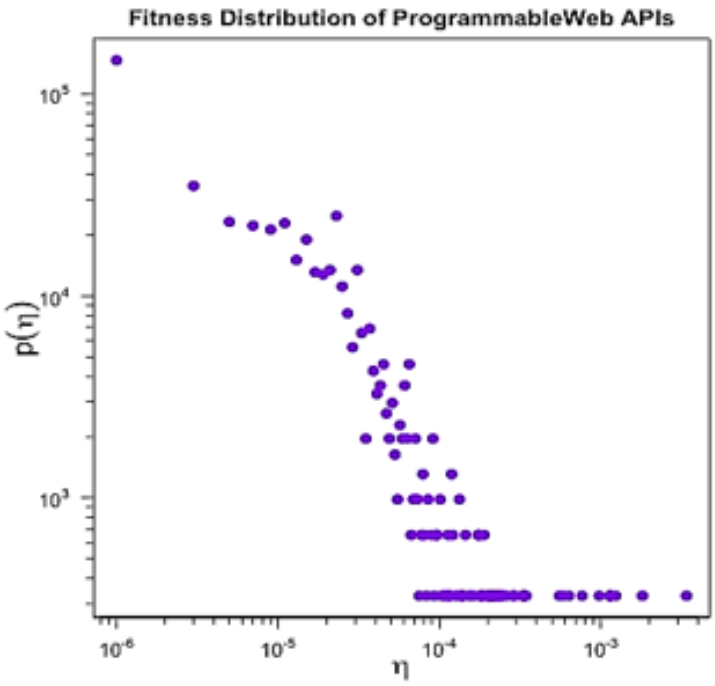


Figure 4.7: Web-API fitness distribution in Log-Linear Binning

.

and (iii) Competitive aspect, which involves incorporation of fitness values.

Growth Aspect :

The growth aspect is very similar to the BA-based network construction approach. it involves continuous addition of new nodes (Web-APIs) into an open network. Therefore, the number of nodes N in the network increases throughout its life span. To achieve this, similar to the *popularity-based* network, the network is initialized, starting with fully connected m_0 number of nodes. At each time step, a new node (Web-API) A_j with m links is added to the network.

Preferential Attachment :

In the Web-API network, the probability that new node A_j is connected to an existing node A_i already in the network is neither uniform nor random but depend on the degree

k_i of node A_i. To incorporate *PA*, the probability that a link of the new node connects to an existing node *i* is dynamically estimated based on node's i degree k_i .

Competitive Aspect :

Each candidate node is assigned a fitness value η_i, which enables it to compete for edges at the expense of other nodes. As shown in Equation 3.5, the probability π_i that a new node will connect to a node i already existing in the Web-API network is directly proportional to node $i's$ degree k_i and fitness η_i. In general, the BB model enables the Web-API network to account for the fact that APIs' nodes with different intrinsic properties acquire links at different rate. It predicts that node (API) growth rate is determined by its fitness η and allows estimation of the degree distribution on the fitness distribution $\rho(\eta)$.

The same node ordering strategy used in the previous section is used to to preserve the popularity information in the affiliation network. Note that the strategy is defined to know *when* an API joins the growing network.

The complete network construction procedure is described below:

Network Growing Procedure

Input Parameters (N: number of nodes in the final network, m_0: number of initial nodes, $m \leq m_0$: number of links added at each time step)

1. Creating node list: Sort Web-APIs nodes based on popularity and date-of-birth
2. Assign fitness value to each corresponding new node according to the distribution $\rho(\eta)$
3. Initializing network: Start with a fully connected m_0 number of most popular nodes;

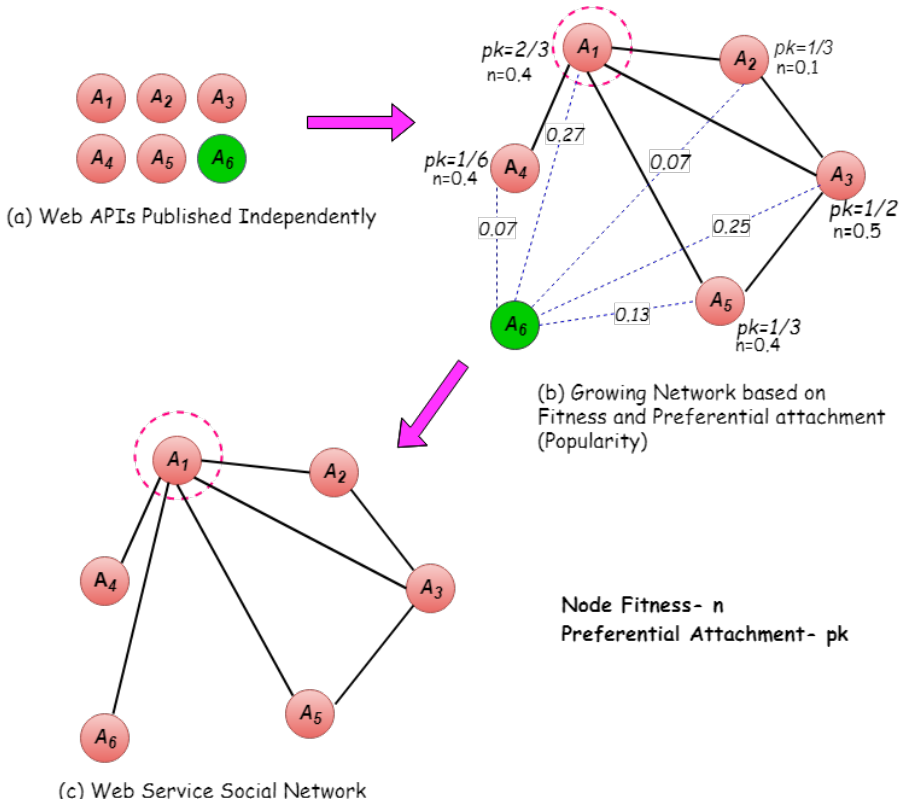


Figure 4.8: Illustration of Fitness-based Web-API *Network Growth*.

4. Growth: At each time step, a new node with m number of links is added and connected to m number of already existing nodes in the network, where $m \leq m_0$;

5. Attachment Probability: With probability $\Pi(k_i)$ estimated dynamically based on equation (3.5), the new node connects to an already existing node i with degree (k_i) and fitness η_i.

6. After all N nodes join the network, we obtain the fitness-driven, evolving Web-API network.

Figure 4.8 illustrate the simple network growing procedure. Figure 4.8(a) shows how Web-APIs published independently in service registries (b) shows new API A_6 joining the open network. A_6 can connect to any node in the network, but A_1 has highest probability $\Pi(k_i) = 0.27$ (combination of its fitness and degree) to attract

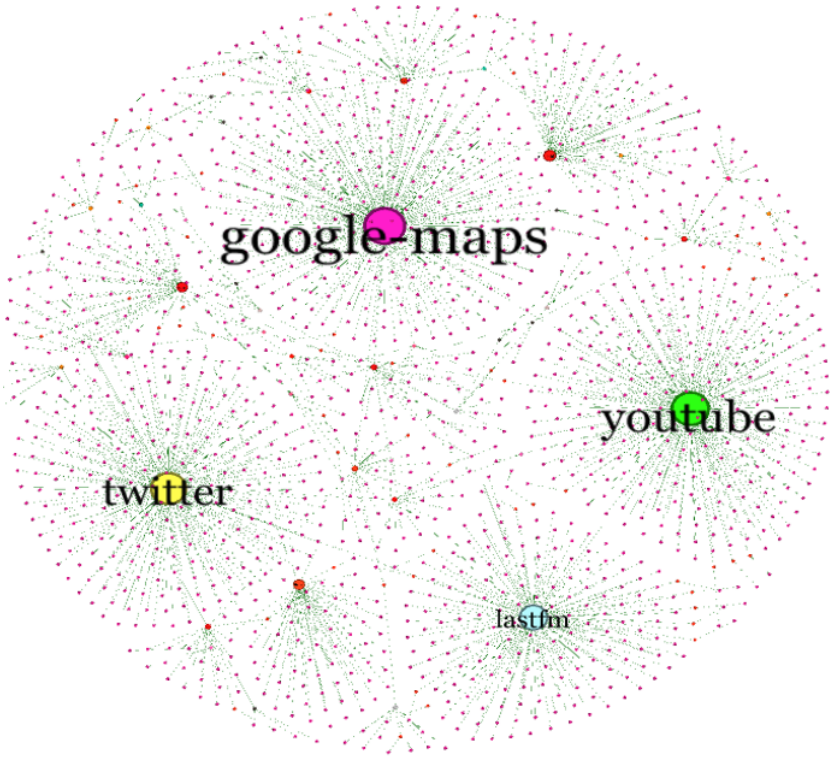


Figure 4.9: Fitness-based Web-API *Network Overview*.

A_6. An overview of the constructed fitness-based ProgrammableWeb API network containing all its Web-APIs is shown in figure 4.9. In contrast with the BA-based network topology shown in 4.5, which is based solely on degree of nodes with many hubs, figure 4.9 shows the fitness-based network with few hubs. Three API nodes, *google-maps*, *twitter* and *youtube* with high fit and popularity attract majority of the links, and also connected to some smaller hubs called *spokes*, leading to formation of *hub -spoke* topology.

4.4.4 Constructing Popularity-Similarity Based Web-API Network

For Web-APIs, popularity is just one dimension of attractiveness that drives Web-API interactions or their co-invocations. Functional *Similarity* is another dimension of attractiveness. Web-APIs with similar functionality can compete in an invocation

process as either substitutes or replacements (Maamar, Faci et al., 2011). Hence, while popularity attracts new connections, similarity is just as attractive. For example, just like a reader could follow a link from a blog post (say a popular or trending post) to another interesting and related post, likewise, a service consumer could discover a new *mapping* Web-API that is not as popular as *Google Map API* but similar to it, if the new API node is positioned close or directly connected to the *Google Map API*. As earlier discussed, the PSO model (Papadopoulos et al., 2012) has a geometric interpretation in hyperbolic space where the trade-offs between popularity and similarity with which new nodes are expected to optimise when joining a network system are abstracted by the *hyperbolic distance* between the nodes and the existing nodes. The hyperbolic distance constraints play a significant role in the link-formation process between Web-API nodes. To combine the two dimensions of attractiveness in Web-API network construction, a product metric *Popularity* × *Similarity* is used. Thus competitions between the two dimensions determines the overall connection probability. Even-though there is a competition between popularity and similarity, it expected that larger popularity and small similarity (more similar), small values of *Popularity* × *Similarity* are more preferable connections, which takes similarity more seriously

This popularity-similarity optimization forms the basis for the construction of the Popularity-Similarity Web API network. This enables the integration *similarity* (with popularity) as another dimension of attractiveness in our network construction. The same Web-API node ordering to the one described in section 4.4.1 is used, where the nodes are ordered based on their *date-of-birth* (L_2), and their *popularity* (L_1) in the service ecosystem (ProgrammableWeb) as captured by the degree information of the *affiliation* network. The nodes were labelled based on their position in the sorted node-list (we append L_2 to the end of L_1 and sort accordingly). Then, the similarity space is defined as an hyperbolic disc of radius R and denote the time of the i_{th} node on the list as t_i (same as the time the node joins the network) and its angular coordinate

Algorithm 4.3: PSO-Based Web-API Network Model

Input:

N: total number of nodes, $N > 0$
m: parameter controlling the Avg. node degree $\bar{k} = 2m$, $m > 0$
β: popularity fading parameter , $\beta \in (0, 1]$
W: Web-API functionality similarity matrix
$node_{DoB}$: *date-of-birth* for all APIs
G_{af}: Affiliation network

Output:

G_{psn}: the PS-based Web-API complex network

Procedure

$nodeDegree \leftarrow G_{af}.getDegree$
$L_1 \leftarrow Sort_{desc}$(nodeDegree.$keys()$, $degree$)
for i in N **and** i **not** in L_1 **do :**
    $isolatedNode[i] \leftarrow getDoB(node_{DoB}, i)$
    $L_2 \leftarrow Sort_{asc}(isolatedNode.keys(), DoB)$
end for
$L_1.\ append(L_2)$
Compute $\tilde{\mathbf{W}} = (D^{\frac{-1}{2}} W D^{\frac{-1}{2}}) * (-1)$
$diag(\tilde{\mathbf{W}}) \leftarrow 0$
$\eta = \left(\frac{\tilde{\mathbf{W}} - min(\tilde{\mathbf{W}})}{max(\tilde{\mathbf{W}}) - min(\tilde{\mathbf{W}})}\right)$
Compute scaled angular distance matrix $\mathbf{S} = (1 - \eta)2\pi$
$\theta = \mathbf{S} \rightarrow \mathbb{R}^n$ where $n << N$
$Coords = \{\}$;
Initialize the network G by placing first node i at (r_1, θ_1) where r_1 is set to 0 (at the centre, radius is 0).
for $i = 2 \ldots L_1.getLenght$ **do :**
    for $j = 1 \ldots i - 1$ **do :**
        $r_j \leftarrow \beta * 2 * \log(j) + (1 - \beta) * 2 * \log(i)$
    $r_i \leftarrow 2 * \log(j)$
    Get $Coords\{r_i, \theta_i\}$
    Compute $p(i, j)$
    $G_i \leftarrow (G_{i-1}, p(i, j), m)$
$G_{psn} \leftarrow G_i$
return G_{psn}

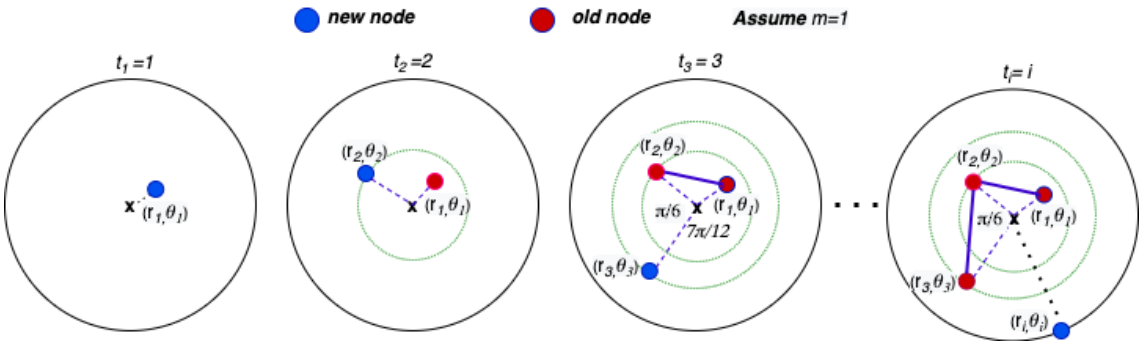


Figure 4.10: Illustration of the PSO model growth procedure. The angular coordinate θ_i abstracts the API *similarity*, while radial coordinate $r_i = 2\ln(i)$ represents *popularity* – node birth time/degree. We start with an empty network and initialize the network by placing the first node $i = 1$ on the API node-list at angular position θ_1 on the circle . At early times $t \leq m$ (assume $m = 1$), node i connects to all the existing nodes. At time $t = 3$, new node at polar coordinate (r_3, θ_3) connects to a subset of the existing nodes. It connects to node r_2 because $2\theta_{2,3} = 2\frac{\pi}{6} < 1\theta_{1,3} = 7\frac{\pi}{12}$.

in the similarity space as θ_i. Following the procedure in section 4.3.3, we describe in detail the processes involved in the Web-API network construction as follows:

Input Parameters:

The model parameters for the Web-API PS-based network construction were set based on the results of the initial analysis of the Web service ecosystem described in Chapter 3 : (i) N: total number API nodes in the network (as in the original dataset); (ii) m: parameter controlling the average node degree $\bar{k} = 2m$, m is set as the average degree k of the original network data (*API affiliation network*) (iii) β : popularity fading parameter, $\beta \in (0, 1]$. Recall the relationship between β and the exponent γ of the power-law degree distribution $\gamma = 1 + 1/\beta$. Thus, $\beta = \frac{1}{\gamma - 1}$; Initially, we set $\gamma = 2.2$, which is the value obtained from our preliminary analysis of the ProgrammableWeb data. (iv) For parameter, since high clustering is a desirable property for our API network, we vary parameter $T \in [0, 0.2]$ such that the network clustering is maximized at $T = 0$, and it decreases almost linearly for $T = [0, 0.2)$.

Growth Aspect : For the growth aspect, it started with an empty network. Then, at time $t \geq 1$, API node i on the sorted node-list is placed on the circle with polar coordinates (r_i, θ_i). As the node i arrived at time t_i, it is connected to m existing nodes j in the network with the m smallest values of product $j\theta_{ij}$, where θ_{ij} is the angular distance between node i and j as shown in Figure 4.10 illustration. If T is set to 0, the new node i connects to the m hyperbolically closest nodes, if $T > 0$, the new node picks a randomly chosen existing node $j < i$, and given that i is not already connected to the j, i connects to j with probability $p(ij)$ defined in equation 4.7;

The procedure is repeated until i is connected to m nodes. The growth process will stop after all the API nodes N have joined the network. Algorithm 4.3 describes the $step - by - step$ procedure for constructing the PSO-based Web-API network.

- Procedure $Lines\ 1 - 6$ follows thesame node-list creation procedure used for BA network.

- In $Line\ 8$, the Web-API functional similarity matrix W is normalized. *Normalized graph Laplacian* method is applied for normalization because it produce a symmetric similarity weight, such that $w_{i,j} = w_{j,i}$, which is a desirable property in this case. In $Line\ 9$, the diagonal of the normalized matrix is set to zero to avoid self-loop.

- Instead of arranging the node over the similarity space using angular coordinates uniformly sampled at random in $[0, 2\pi]$ as done in the original model, θ_i is defined as the normalized Web-APIs *functionality similarity* – (detail of Web-APIs *functionality similarity* computation is described in section 3.2.3).

- In $Lines\ 10 - 11$, the matrix is scaled with scaling factor $(1 - \eta)2\pi$ to get the angular distance between two data point, such that $0 \leq \theta < 2\pi$, where η is the normalized similarity measure for each data point (Note that the higher the

functional similarity, the lower the distance, hence the scaling factor $(1 - \eta)2\pi$).

- Then, in $Line - 12$, dimension reduction technique described in (Tenenbaum, De Silva & Langford, 2000) is applied to reduce the matrix dimension from N to n, where $n = 2$ to get the angular coordinates θ for each data point.

- $Lines\ 13 - 22$ describe the network growing procedure in hyperbolic space. Initially, the first node (on-top of the ordered node-list) (L_1) is put into the network G_1 with radial coordinate ($r_1 = 0$) and the corresponding angular coordinate ($\theta_1 \in [0, 2\pi]$). For each node $t_i = 2, \ldots, N$, three key operations are performed: (i) node t_i is added to the network and assigned a radial coordinate $r_i = 2\log(t_i)$; (ii) the radial coordinate of every existing node $t_j < t_i$ is increased according to $r_j(t_i) = \beta r_j + (1 - \beta)r_i$ – the popularity fading parameter β is set based on the resultant γ value obtained in Chapter 3 . (iii) $p(i, j)$ is determined based on equation 4.7, and then new node t select a randomly chosen node $s < t$ that is not already connected to it and connect with probability $p(x_{i,j})$. Since, high clustering is one of the desirable properties for our network, parameter T is varied such that $0.2 \leq T \geq 0$

- The process is repeated until node i gets linked to m different nodes . The network growing process is repeated until N nodes are added and connected.

4.5 Network Analysis and Results

This section report the conducted experiments to analyse the Web-API networks proposed in this chapter. The focus here is to analysis the networks from both topological and user application perspectives. The Web-API networks proprieties are mapped with typical Web-API ecosystem properties (*discoverability* and *navigability*) using

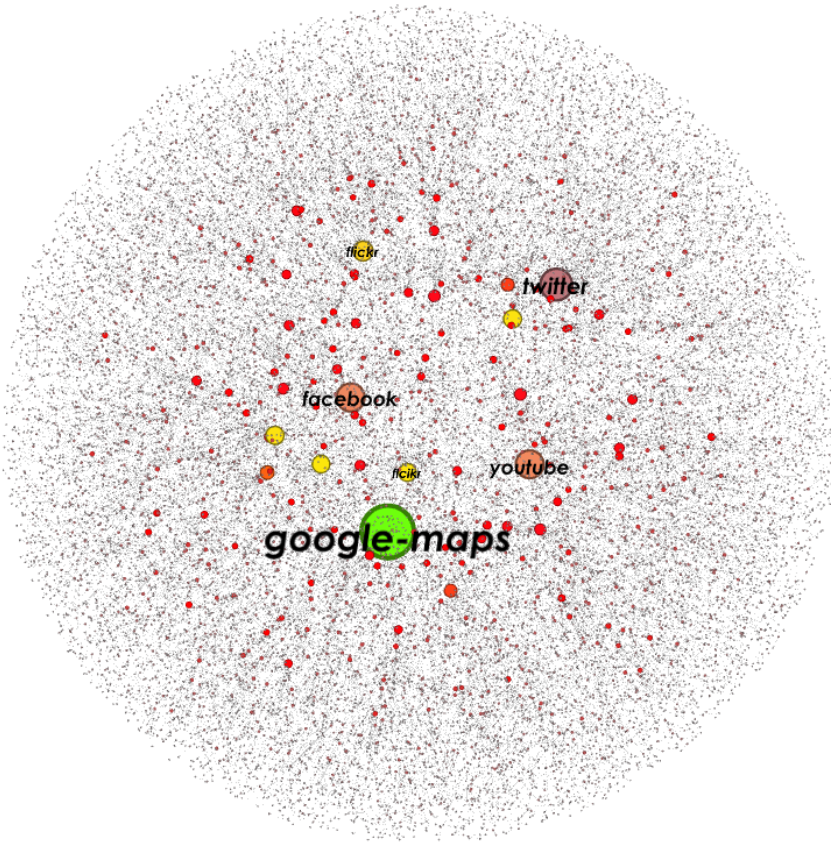


Figure 4.11: Overview of the PSO-Based Web-API Network

ProgrammableWeb as the case study. End-user activities in ProgrammableWeb such as *searching* , *browsing* and *visualization* is connected with the network properties.

4.5.1 Experimental Setup and Dataset

Five different Web-API networks are constructed : three evolving networks ($BA-based$, Fitness-based and $PSO-based$ *Web-API networks*) and two static networks as baselines: *Web-API affiliation network* and a data-driven, *Web-API correlation network* using thesame Web-API dataset described in Chapter 3.

Constructing a Correlation-Based Network from Web-Service Data

This section presents one of the baseline networks used in the evaluation of the evolving Web-API networks. It presents the data-driven correlation network , which exploits functional relationships captured by the textual descriptions of Web-APIs. For this network, the functional similarity between Web-APIs is exploited for creating links between related Web-API. A dense, *API-API correlation* network by inducing edge sets from Web-API *data points* using a distance *metric* $d(a_i, a_j)$ to measure the distance between node API nodes a_i and a_j − similar to the approach introduced in (Grady & Polimeni, 2010). The pairwise distance d_{a_i,a_j} between API a_i and a_j is computed as the *Euclidean* distance between their feature vectors learned in a *semantic space* . Thus, given n points $\{a_1, \ldots, a_n\}$ in $\mathbb{R}^d$, a network with n nodes is built, and set of edges connecting close (similar) nodes.

Note that nodes in this network could be either Mashups or APIs, and their *correlation coefficients*− edge weights E_{a_i,a_j} between node a_i and node a_j defined by *thresholded Gaussian Kernel weighing function* (Shuman, Narang, Frossard, Ortega & Vandergheynst, 2013; Grady & Polimeni, 2010).

$$E_{a_i,a_j} = \begin{cases} exp\left(-\frac{[d(a_i,a_j)]^2}{2\theta^2}\right), & \text{if } d_{a_i,a_j} \leq \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{4.13}$$

The *correlation coefficients* (edge weights) reflect a cor-relationship (through substitution, association, co-invocation and functionality) between Web services across the dataset. The assumption here is that Web service ecosystem could be represented as a fully connected network with positive weight indicating presence of connection between adjacent service nodes and *zero-valued* weight to indicate no connection.

The network construction procedure follows three algorithmic steps:

- First, a vector representation for each Web service data point including mashups

is generated by getting a representative geometric values of their description using *doc2vec*(Q. Le & Mikolov, 2014).

- To build connection between the embedding data, the distance $d(a_i, a_j)$ between the two feature vectors describing a_i and a_j is computed.

- Finally, nodes a_i and a_j are connected by an edge if they are $close$ such that edge set $\mathcal{E} = \{E_{a_i,a_j} \mid d(a_i, a_j) < \epsilon\}$, where ϵ is a *free* parameter. The resulting network is *naturally symmetric*− a feature that is important in network-based service discovery applications. ϵ is carefully selected through *grid search* to ensure that no similar or close node is left isolated. Figure 4.12 shows the visualization of the API-API network with $17,959$ nodes, and 137902 edges. We set the threshold ϵ very high to enable high (0.8 for our experiment) degree of correlation between connected node in the network . The figure shows the distribution of node degree (the number of connection that each node has) is *heterogeneous* such that majority of the nodes in the network have very few links, while some *hubs* and *spokes* also exist.

The above procedure could also be used to create a denser $API-API$ network. Another approached for inducing edges between unconnected, embedded data point is using the k−*nearest neighbour* to the node (Grady & Polimeni, 2010), where a link is created between a node and each of its its k−*nearest neighbour* nodes with a smaller distance than the rest of the node set. This approach gives a guarantee on the degree structure of the resulting network, such that degree of node i is greater or equal to its k−*nearest neighbour*. As a result of this, if $k > 0$, then there will be no isolated Web service node in the resulting service network.

Table 4.2 summarizes the statistics of each network constructed in this work. In contrast to the *affiliation network*, which only connects $1,525$ APIs, the BA-based, Fitness-based and PSO based networks connect $17,959$ Web-API data-points in the

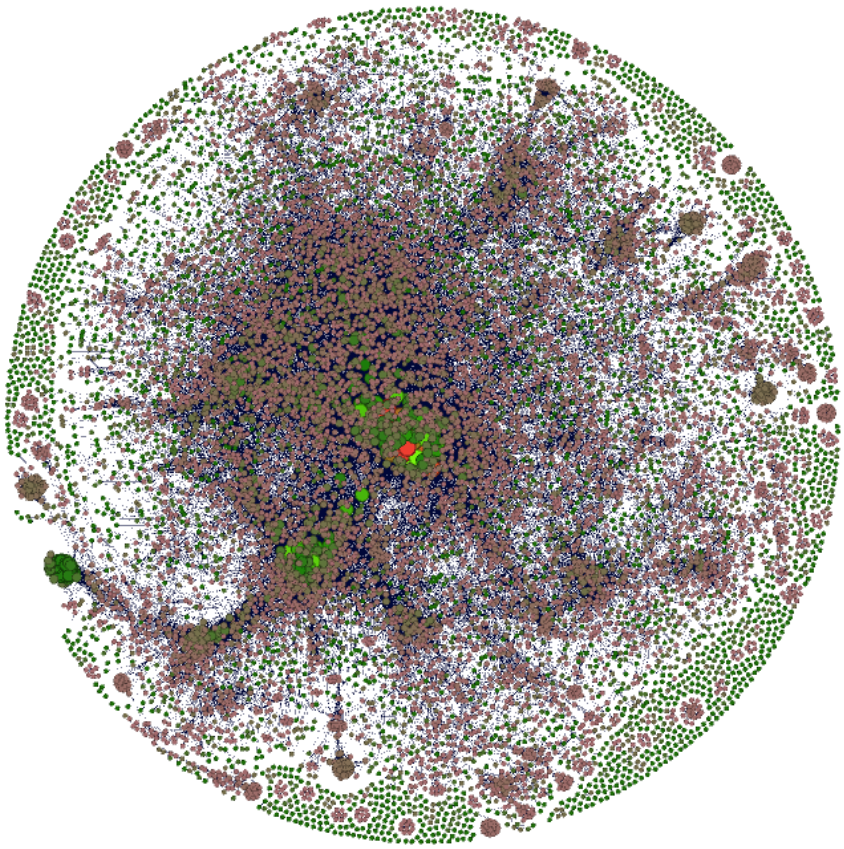


Figure 4.12: Web-API Correlation Network with $137,902$ number of edges and $\epsilon = 0.8$. The big light green and red discs are the hubs, while the visible brown patches are the clusters. 617 nodes with $k > 100$ are identified in the network.

ecosystem, thus, solving the Web-API isolation problem with $53,877$ total number of links per network. The network simulations are ran 10 times for each of the evolving networks and the average measure of the relevant network properties is taken. For *offline* analysis, we use *Gephi* and *Python Networkx*.

4.5.2 Mapping Web-API Ecosystem Properties with the API Networks Properties

Here, we quantitatively measure the topological properties of the evolving Web-API networks, and analyse the correlation between these properties and the ProgrammableWeb system properties.

Analysing Web-API Networks Degree Distributions

Following the same plotting, fitting and dynamic exponent γ estimation procedures described in section 3.2.1, a check is run to see if the proposed network models can preserve the API popularity information observed in the original dataset from ProgrammableWeb (as in the *affiliation network*). this is done by comparing the degree distributions of the networks with that of the *affiliation network*. The plausibility of the degree distributions is measured and *goodness-of-fit* test is conducted on the distributions .

Table 4.1: Summary of BA-based Vs BB-based Web-APIs Network Features

Comparison of PA driven and BB-based Web-API networks		
Networks	PA-based Web-API Net	BB-Based Web-API Net
Dynamic exponents	$\beta = 1/2$	$\beta(\eta) = \frac{\eta}{C}$
Clustering Coefficients	$\langle C_c \rangle \approx 0.006$	$\langle C_c \rangle \approx 0.048$
Network Diameters$\langle d \rangle$	$\langle d \rangle \approx 5.81$	$\langle d \rangle \approx 4.27$
Topology	*scale-free*	*scale-free*
Degree exponent(γ)	2.7	2.252

For the constructed Web-API network with $N = 16138$ and $m = 3$, and with η assigned to each node, the degree exponent is $\gamma = 2.25$, which follows the theoretical value. Fig. **??** shows a long-tailed degree distribution of the fitness-based Web-API network on linear-log scale. In the figure, nodes with large degree (high fitness nodes) form a plateau at the k region, the small degree regions occupy by nodes with low degree and low fitness values, establish a log-linear relations between degree distribution $p(k)$ and k ($\log p(k) \sim -\gamma \log k$, or $p(k) \sim k^{-\gamma}$). In Fig. 4.14, we extract information from the long-tail of the distribution by binning the data points and fitting the long-dashed-line, which represent $p(k) \sim k^{-2.252}$. As shown in the figure, the probability $p(k)$ is roughly proportional to $k^{-2.252}$. Hence, the network topology is nearly *scale-free*.

The numerical solution of $C = 1.255$. Hence, equation 4.4 predicts that the dynamic

exponent of each node i is different across the network, i.e. $\beta(\eta) = \eta_i/1.255$, where η_i is quenched and different for each node in the network. BB model is reduced to BA model when fitnesses for all nodes are taken to be equal. As shown in Table 4.1, $\eta = 1$ for each node in BA model and equation 4.4 gives $C = 2$, hence, $\beta = \frac{1}{2}$.

Figure 4.13 shows the results of the fitting for each network. We can see that both *power-law* and *log-normal* distribution models offer a good fit to the degree data (in CCDF) of Web-API affiliation (Aff), BA and PSO based networks , while the exponential and Poisson distribution models fit poorly to the data. For the correlation-based network, the degree data-points fit fairly well to *power-law*. We measure the plausibility of each distribution following the same procedure as in Section 3.2.1, Table 4.3 shows the resultant $p\text{-}values$ for each distribution model. Clearly, power-law is the most plausible fit for all the networks with Web-API affiliation (Aff), BA and PSO based networks having high $p\text{-}values$ scores 0.784 , 0.728 and 0.830 respectively. Moreover, the ***PSO network*** fit significantly to the log-normal distribution with $p\text{-}values$ = 0.636, validating the applicability of the network data in simulating real-world API interactions (?, ?). We apply equation 3.2 to estimate the *degree exponent* γ using MLE. The γ values for the *BA-based Network* and the PSO-based network are 2.85 and 3.01 (close to that of the Internet : γ = 3.42 (Barabási, 2016)) respectively . For both BA and PSO-based network, we estimate the lower cutoff for the scaling region as k_{min} to be 3, which is consistent with the number of links m used to grow the networks during simulation (the minimum degree). For m values between 2 – 10, both models produce scale-free network with degree exponents γ between 2.8 – 3.2 when fitted to power-law model. The exponent γ remains thesame for larger values of m.

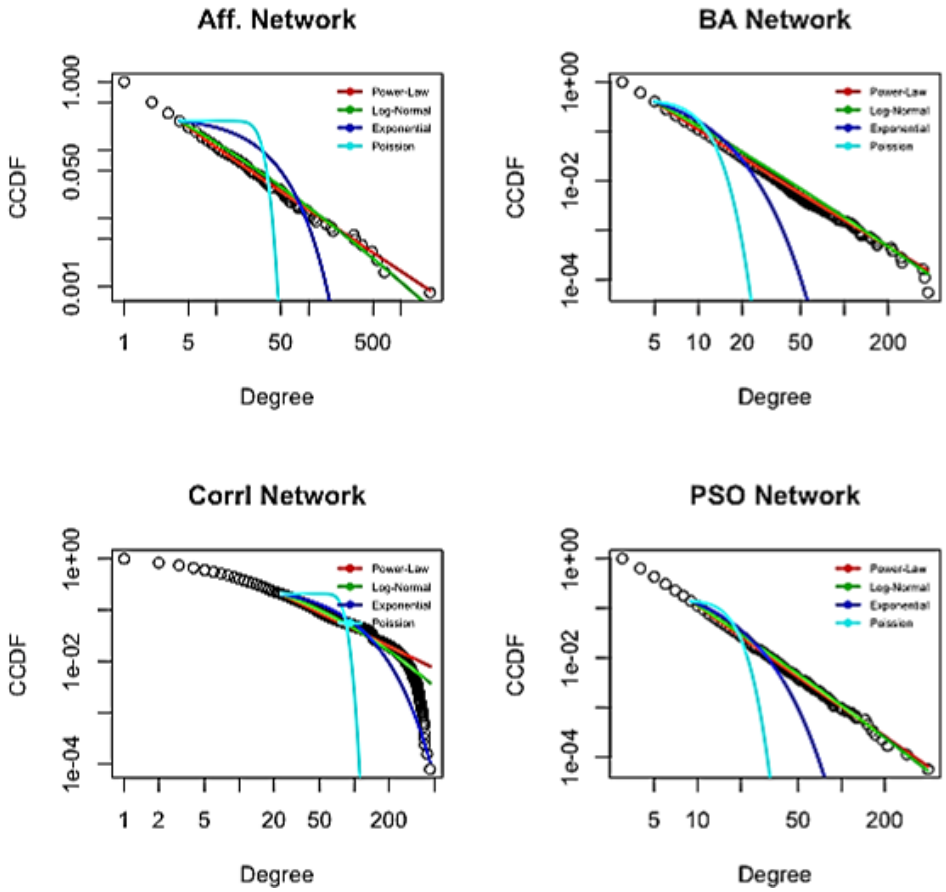


Figure 4.13: Fitting Power-law (PL), Log-normal, Exponential, and Poisson models to the Web-API Networks degree data.

Table 4.2: Web-APIs Networks Properties and Navigation Performance

Properties/Networks	Affiliation	Correlation	BA-based	PSO-based
Nodes (N)	7410	17959	17959	17959
Edges (m)	13231	137902	53877	53877
Node type	$mashup - API$	$API - API$	$API - API$	$API - API$
Network type	static-asymmetric	static-symmetric	dynamic	dynamic
Avg. Degree	2	21	3	3
Avg. Path Lenght	4.3	6.2	4.45	5.9
Diameter (D)	15	23	6	11
Degree Exponent (α)	2.1	-	2.7	3.1
Clustering (C)	0.18	0.55	0.006	0.72
No of Hubs $(k > 100)$	15	617	8	19
Navigation Eff (EC)	0.073	0.42	0.232	0.85
Avg. Closeness	0.303	0.133	0.226	0.175

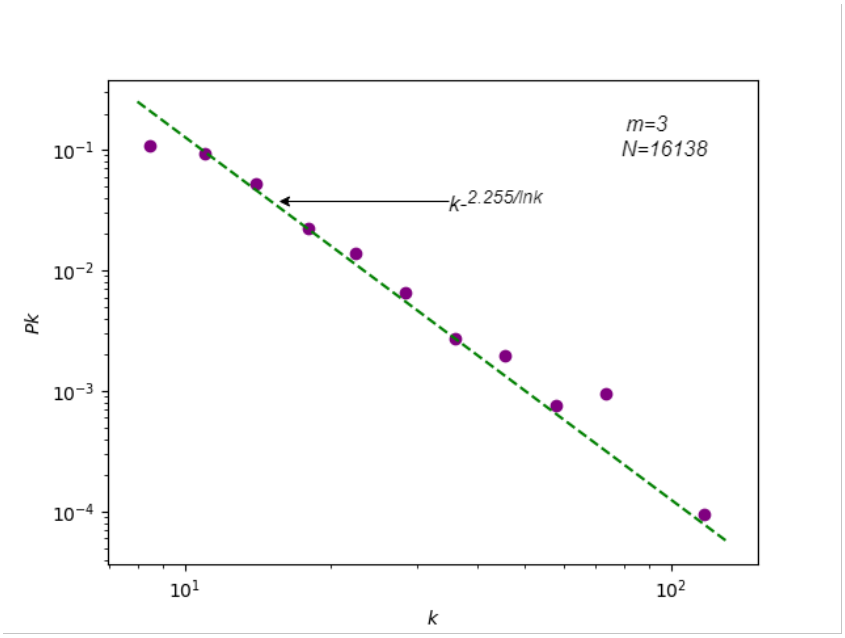


Figure 4.14: PDF plot with log binning of the Fitness-based Web-API network *degree distribution).*

Analysing Small-World Properties in Web-API Network and their Correlation with API Discovery

We measure the two specific properties in the Web-API networks that characterised small-worldness: *average path length* and *local clustering coefficient*.

Average Path Length : This is the mean distance between two nodes, averaged over all pairs of nodes in the Web-API networks. From discovery perspective, a short average

Table 4.3: Plausibility of fitting different distribution models to the Web-API Networks degree datasets

API Networks	***Power-law***			***Exponential***		***Log-normal***			***Poisson***	
	k_{min}	***p-value***	γ	***p-value***	γ	k_{min}	***p-value***	γ	***p-value***	γ
Affiliation	1	0.7841	2.2163	0.0000	-	1	0.687	-	0.0002	-
BA-based	3	0.8301	2.8531	0.00001	-	3	0.580	-	00000	-
PSO-based	3	0.7281	3.0157	0.00001	-	3	0.673	-	0000	-
Correlation-based	21	0.259	-	-	-	21	0.262	-	-	-

path length indicate high reachability of nodes (Borgatti, 2005), that is, most node can be reached from others through small number of links. Table 4.2 shows the average path length of each of the networks: Affiliation network , BA and PSO-based API networks average path length are 4.30 , 4.45 and 5.90 respectively.

Clustering Coefficient: We measure the *local clustering coefficient* (CC_i) of node i with degree k_i, which quantifies how close node $i's$ neighbours are to being a clique (a complete graph). CC_i is mathematically defined as :

$$CC_i = \frac{2|e_i|}{k_i(k_i - 1)} \tag{4.14}$$

where $|e_i|$ is the number of edges shared by the direct neighbors of node i (number of triangles formed by node i and any of its two neighbours) (Silva & Zhao, 2016). We can see that $CC_i \in [0, 1]$, if the neighborhood is fully connected, the clustering coefficient is 1, and a value close to 0 means that there are hardly any triangular connections in the neighborhood. In service discovery, high clustering values are desirable, the strongly interconnected nodes can be exploited to create local collection of specialized clusters which could aid discovery (W. Chen et al., 2015b). Fig. 4.15 shows the clustering coefficient distributions across the API networks. Clearly, the PSO network has very high number of nodes with CC_i scores close to 1, while the BA-based Web-API network has a CC_i scores are close to 0 (consistent with the empirical result in (Barabási, 2016)). Table 4.2 shows the *average clustering coefficients* (CC) for the networks. The *PSO-based* network has very high CC score with *average clustering coefficient* 0.72 and 0.55 respectively – an indication that many nodes are tightly connected by sharing common origins. On the other hand, the clustering for the BA-based API network is very low (close to 0), with average *average clustering coefficient*=0.006, which shows that most nodes did not form clique with their neighbours. This is again consistent with the results in (Barabási, 2016). The *affiliation* network has an *average clustering*

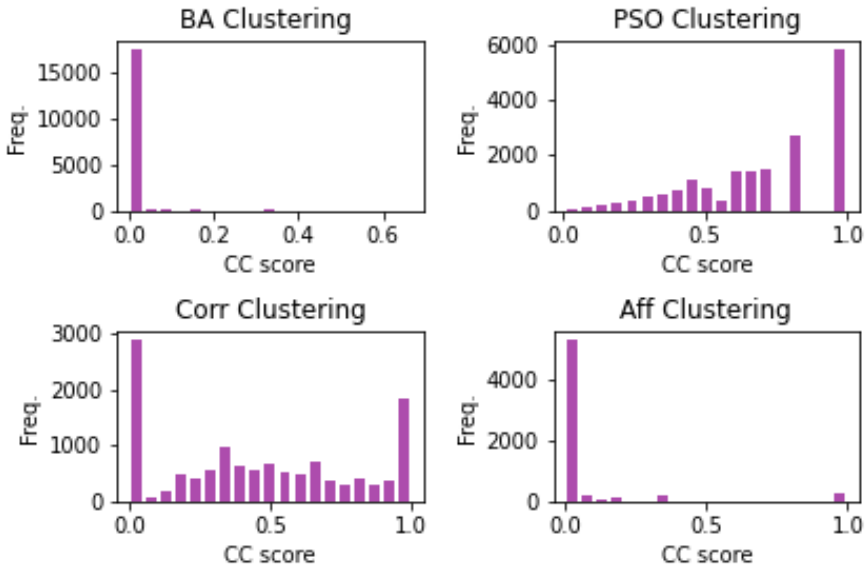


Figure 4.15: The clustering coefficient distribution of the Web-API networks.

coefficient= 0.18, an indication that nodes in our *popularity-similarity* API network is better connected than the APIs in ProgrammableWeb system .

Analysing Path-Based Centrality in Web-API Network and their Correlation with API Discovery

Centrality measures quantify how important nodes or edges are in a network. There are different types of centrality measures (Silva & Zhao, 2016). Here, we only focus on *closeness* as a path-based centrality measure because of its relevance to API discovery. We measure the *Closeness* centrality of nodes in Web-API networks to reflects the closeness between a node and other nodes in the network. Thus, the closer a node is to all other nodes in the network, the higher the centrality of the node is. For the Web-API network, the closeness centrality $C_{cl}(i)$ of node i is defined as the inverse of the average shortest path length of the node i to all other nodes:

$$C_{cl}(i) = \frac{N-1}{\sum_{j \neq i} d_{ij}}, \tag{4.15}$$

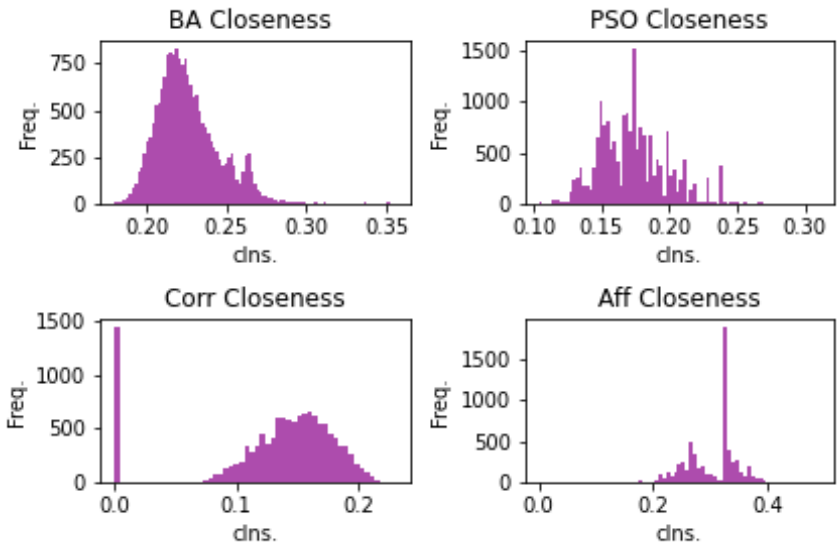


Figure 4.16: Closeness distribution for Web-API Networks.

This simply means we measure for node i the average shortest path length from i to all other nodes j in the Web-API networks, then take the inverse of *farness*, where d_{ij} is the shortest path length between nodes i and j. Nodes with low closeness scores have short distances from others, and so will tend to be discover quicker, assuming that *search* originates from all other nodes with equal probability, and also assuming that whoever is searching manages to travel along shortest paths. Figure 4.16 shows the closeness distribution of the Web-API networks. We note that even-though some nodes in the *affiliation* network have high closeness scores (between 0.2 - 0.4), the percentage of highly $'close'$ nodes is small compare to our proposed networks due to the few number of nodes involved in this network. As reported in Table 4.2, the average closeness scores are 0.303, 0.226, 0.175 and 0.133 for *affiliation* , *BA-based*, *PSO-based* and *Correlation* Web-API networks respectively.

Navigability of Web-API Networks

From API discovery standpoint, a highly navigable API-network is desirable. For example, in a highly connected network, users can explore the network characteristics

such as shortest paths to navigate the network starting from the *hubs*. Thus, we measure the navigation performance of the Web-API network using the distance between the network nodes. We consider *navigation* as a simple process of progressing from a source node to the next node with the closest geodesic distance to a desired target node, then terminating when the target is reached. We adopt the *navigation efficiency* formulation described in (Seguin, Van Den Heuvel & Zalesky, 2018) which used the inverse of shortest-path length to measure the navigation performance. First, we compute the navigation efficiency E_{ij} for transversing between nodes i and j in a given Web-API network to be inversely proportional to the navigation length l_{ij}, that is $E_{ij} = \frac{1}{l_{ij}}$, where $l_{ij} = l_{iu} + \cdots + l_{vj}$ where $\{u, \ldots, v\}$ are sequence of intermediate service nodes visited during the navigation process. In case where there is no path between i and j, $l_{ij} = +\infty$ with resulting $E_{ij} = 0$. We compute the average navigation efficiency for each API network G as follows:

$$E(G) = \frac{1}{N(N-1)} \sum_{i \neq j \in G} \frac{1}{l_{ij}}. \tag{4.16}$$

Using the information in the Web-API networks, we compute the navigation matrix $\{l_{ij}\}$ as the shortest path length between two generic nodes i and j. This metric is similar to network global efficiency which is used to generalize the *small-world behaviour* (Barabási, 2016). We also compute the local navigation efficiency as the average of efficiency of local *subgraphs* : $E_{loc} = \frac{1}{N} \sum_{i \in G} E(G_i)$

Table 4.2 shows both E_G and E_{loc} for the Web-API networks. Both *PSO-based* and *Correlation* Web-API network show relatively high local efficiency and moderate global efficiency –the *PSO-based* Web-API network has 85% and the correlation network has 42.1% local efficiency.

4.6 Chapter Summary

In summary, we conclude from the results of our analysis that the proposed evolving Web-API networks not only solve the isolation problem associated with a typical Web-API ecosystem but they also exhibit common topological properties found in real world networks system like WWW, internet, biological and social networks (Barabási, 2016; Borgatti, 2005; Albert et al., 1999) which can be exploited in API discovery. Both the BA and PSO-based Web-API evolving networks preserved the system and network properties of the API ecosystem (the ProgrammbleWeb modelled as Affiliation network). The PSO-based, Web-API network enables the integration of functional similarity into the network construction and thus induces properties such as high navigability and clustering coefficient which can be exploited in API discovery application. The measures of the PSO network indicate that the network clearly exhibit similar properties commonly found in a classical *small-world* network. Results presented in this work will not only provide insight into the topology of the Web-API ecosystems but also serve as a practical guide for designing an evolving-network-based solution for Web service discovery.

Chapter 5

Complex Network-Based Web Service for Web-API Discovery

The emergence of Service Oriented Computing as an effective computing paradigm for assembling complex and composite Web applications from distributed, standalone application components like Web-APIs, have helped in simplifying software development processes, increasing service reusability and reducing software development cost and time (Metrouh & Mokhati, 2013). There have been a rapid growth in the number and popularity of Web services due to the continual developments in Web service technology, and the widespread adoption of *RESTful architecture* and cloud computing. The number of service directories, portals and marketplaces where services providers can be advertised their services is also increasing. Other emerging computing technologies and platforms like cloud , big data technologies and mobile computing are leveraging and exploring various service functionalities and benefits to facilitate their respective operations (Bouguettaya et al., 2017; Tan, Fan, Ghoneim, Hossain & Dustdar, 2016). Web-APIs such as *Google-Maps*, *Twilio* and *Twitter* APIs form the building block of many mobile and web-based applications. They provide means for simplifying cross-organisational interoperability, and enable the integration of business

functionalities over the Internet. Because of the wide and continuous adoption of Web services, it is expected that they will continue to secure a core role in various computing domains and increase in numbers, and thereby promoting the emerging service economy (Tan et al., 2016). Modern Web services with features such as *RESTful Web services*, *JSON data*, and/or *JavaScript interface* are usually called *Web-APIs* [1] in order to distinguish them from the traditional SOAP-based Web service. These Web-APIs can be published, discovered, selected by API consumers e.g. software developers. They are capable of communicating with both applications and users, and can be used to create new, value-added application – known as composition services or *mashups* –, which combine multiple Web-APIs of diverse functionalities from different sources to satisfy complex user requirements. This process shortens software development life cycle and forms the basis for the formation of the so-called Web service ecosystem (Barros & Dumas, 2006a; Lyu et al., 2014), where new services emerge, some old ones perish, and service vendors and developers collaborate to develop innovative software solutions. A typical representation of Web service ecosystem is ProgrammableWeb [2], which is currently the largest online Web-API directory, with over 19,000 Web-APIs belonging to more than 400 predefined categories, and over 7,000 mashups as at June 2020. The obsolescence of some existing Web-APIs and the emergence of new ones coupled with the dynamic interactions that exist among objects of the ecosystem over time drive Web-API ecosystems evolution (Huang et al., 2012b).

Various research efforts have been invested into advancing service computing domain, particularly, in the context of Web service discovery. The processes for discovering, selecting and composing existing Web-APIs have become more practical and reoccurring, especially when developers aim for fast and delivery of software applications. Service discovery as one of the key processes (N. Zhang et al., 2018) for building service-based

[1] https://en.wikipedia.org/wiki/Web_API. Note that in this paper, we coin "Web" and "APIs" together as one term "Web-APIs" to emphasize the atomicity of this term.

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

systems(SBS), generally, involves the process of searching/finding set of candidate services that satisfy specific user's requirements (Metrouh & Mokhati, 2013). Due to the challenges associated with service discovery tasks, various approaches have been proposed to handle or improved existing service discovery frameworks. Most of the existing service registries are based on discovery approaches such as keyword-similarity-based (Cong et al., 2015; He et al., 2017) and semantic-based discovery (W. Chen et al., 2017; Lamine, Zghal, Mrissa & Guegan, 2017; J. Wang et al., 2017; N. Zhang et al., 2018). Keyword-based technique mainly discover services via matching of user's queries with service description or keywords using common information retrieval techniques such as *Term-frequency-Inverse-Document-Frequency* (TF-IDF) (Dong et al., 2004) and *Vector space model* (Platzer & Dustdar, 2005). On the other hand, semantic-based approach incorporate ontological concepts (Lamine et al., 2017), semantic information for both users and services with keyword-based to improve discovery performance (N. Zhang et al., 2018).

However, despite the outstanding progress made so far, the discovery and uptake of Web-APIs on a Web scale has been significantly less than initially anticipated. The continual increase in the number of APIs on the internet coupled with numerous number of APIs with similar functionalities, makes it very challenging for API consumers to discover suitable ones for software development. Most Web-APIs published on the Web are never used; only few of them have ever been discovered by users or invoked (W. Chen & Paik, 2013) in mashups. For instance, about 75% of Web-APIs in ProgrammableWeb have never been invoked in any composition (Duan & Tian, 2017). Moreover, the ever-increasing number of Web-APIs on the internet coupled with their diversity, poses a new, challenging search/discovery problem; that is, finding suitable Web-APIs out of the tens of thousands present on the internet. For a service consumer who intends to compose a new-value added service (mashup), the first step is to search some online Web-APIs repositories like ProgrammableWeb either by using functional

descriptions of require API components or using generic descriptive service requests, and then select appropriate existing Web-API from the repository. However, it is very challenging to sieve through large number of candidate Web-APIs covering a wide range of functionalities in different repositories, and select suitable Web-APIs that match the exact user requirements especially when dealing with a complex mashup-oriented Web-API requirements. As service consumers' needs and software requirements are evolving and becoming more complex, it is more difficult to find a standalone service that can satisfy the requirements. Eventhough, existing discovery approaches such as the semantic and keyword-based approaches have been able to achieved considerable success in tackling service discovery challenges, however, recent studies (Weiss & G.R, 2010; Duan & Tian, 2017; Huang et al., 2012a) show that the problem still persist and main issues are yet to be resolved, with most service yet to be discovered. This is because existing approaches do not only have their unique downsides which limit their performances especially when the user software requirements are complex, but also ignore some specific information in there design that help in promoting/enhancing service discovery. A lot of efforts have been focused on indexing service repository and matching algorithms with little or no attention the services social dimensions and support for efficient user interaction with the service discovery systems. For example, it is often difficult for users (especially new users with little or no software experience) to specify high-quality query or know what keyword to use that better describe the require service function when using a typical keyword-based discovery system. And for semantic-based approaches, the lack of semantic information for modern web services (especially for Restful APIs) and the complexity of building new ontology for service discovery are the major setbacks (N. Zhang et al., 2018). As discussed in the earlier chapter, another major reason that limits Web service discoverability (especially in the commonly used API registries, hubs and portals) is that, Web-APIs registered on these online directories such as *ProgrammbleWeb.com* are in general *isolated*, as they are

registered by diverse providers independently and progressively (W. Chen et al., 2017; Huang et al., 2014b; Duan & Tian, 2017). These techniques consider web services as isolated functional island with no direct link or interaction among each other, and ignore the contribution of the Web service social dimension. Because of these isolated service islands, service discovery faces the following challenges (W. Chen et al., 2015a). One of them is that most techniques to service discovery ignore interactions with service consumers, resulting in a high usability barrier for service users . Service consumers can not find services by clicking on links that interest them, like they can on Web pages. Due to the issue of service isolation, guiding service consumers to find services, starting with the service at hand and expanding to peer services that may be integrated into more complex functions, remains a difficult task.

Recent work on service discovery explore social networks (Hafsi et al., 2020; W. Chen et al., 2017; Fallatah et al., 2014; Bianchini, De Antonellis & Melchiori, 2014; Kalaï, Zayani & Amous, 2015; Maamar, Faci et al., 2011; Yao, Wang, Sheng, Benatallah & Huang, 2018; Feng et al., 2015) to support service discovery frameworks. The integration of two domains, social computing and service-oriented computing creates a new service discovery scheme and concept called Social Web Services (SWS). Incorporating the social aspect into Web services can enhance their exposure and benefit them in becoming active entities that can interact with each other through collaborative, competitive and substitution processes. With social Web service, services can live in a global interlinked network of services where they can better collaborate to satisfy complex consumer's needs. The majority of the existing works relied on limited network data acquired from services repositories such as the co-invocation data and service historical usage data to create the service networks, hence, the problem of service isolation still persist.

This chapter address the task of Web service discovery from complex network perspective by developing an evolving, complex network-based, Web-APIs discovery service

that leverages the capabilities of *Google custom search APIs* [3] to recommend both composite and single Web-APIs to users based on user's service requirements. The complex network-based discovery service explore various complex network properties to support service selection for both simple and complex or composite service requirements . The chapter build on the findings in the previous chapters including the results of the network analysis conducted on Web service ecosystem and the application of the evolving networks constructed in Chapter 4 in solving service discovery. The chapter presents the demonstration of how the evolving Web-API network can be applied to facilitate service discovery process through a prototype evolving complex network-based, mashup-oriented Web-API discovery service.

The network-based service discovery approach presented in this chapter aims to leverage both topological information of the proposed evolving Web service networks, and the Web service system properties to attain better service discovery performance. The prototype design of the complex-network-based service leveraged *Google custom search API* feature to facilitate node ranking based on term frequency, functionality and node popularity information. A strategy for aggregating candidate APIs from a pool of search results based on the keywords in user's query and the ranking position of individual API in the search results is also introduced. To evaluate the accuracy of our framework, the users' service queries are modelled by exploiting word features and structure of original service profiles using Latent Dirichlet Allocation (LDA) (Blei et al., 2003). To validate the proposed discovery approach, the popular programmableWeb dataset in a period of fourteen years (2005-2020) is used.

The rest of the chapter is structured as follows. Section 5.1 presents the background concepts and motivation for the approach presented in this chapter. Section 52. present the types, processing and structure of the dataset used for the experiments in this chapter. Section 5.3 elaborates on the proposed approach; the sections also present various

[3] https://developers.google.com/custom-search/v1/overview

underlying techniques such as the aggregation technique and the page-Rank feature adopted for facilitating node ranking . Section 5.4 presents the extensive experiments, including a comprehensive analysis of results, using a real-world mashup-API invocation dataset from ProgrammableWeb. The section present the effectiveness of the proposed approach by conducting extensive experiments on a real-world dataset crawled from *programmableweb.com*. Compared with existing service discovery methods, experimental results show that our approach significantly improves API discoverability in terms of precision and usefulness . Section 5.5 concludes the chapter.

5.1 Background and Motivation

This section provides some backgrounds about application of complex network theory in modelling Web service system, follow by the motivating example of how this can help solve Web-APIs isolation and improve service discovery.

5.1.1 Complex Network Applications

The method applied in this chapter is inspired by recent development in complex network theory and its application. Over the past 20 years, complex networks have been extensively studied in the network science domain, and several significant discoveries have been made including the well-acclaimed small-world networks (Watts & Strogatz, 1998) and scale-free networks (Barabási & Albert, 1999). Many social, biological and communication systems can be described or modelled using complex network models whose nodes represents entities and edges represents the interactions among the entities. Network scientists have used network models to address two related tasks (Pham et al., 2015): (i) Studying the emergence of topological properties in complex networks and investigating possible mechanisms underlying the network formation. (ii) Modelling of dynamical processes involved in complex network systems in a way that enables

effective exploitation of the known topological features. For the first tasks, various mechanisms that governs network topology and evolution such as *preferential attachment* (PA) and *growth* have been investigated and are found ubiquitous among many real world networks. For instance, the topology of the Internet and the World Wide Web have been investigated using evolving network models and shown to be fundamentally governed by the PA and growth mechanisms (Albert et al., 1999; Barabási, 2016). For the second task, which involves modelling the dynamics of complex systems, several network models that exploit certain attraction mechanisms and can capture common topological features of complex networks have been proposed (Barabási, 2016; Y. Cao et al., 2006; Caldarelli, Capocci et al., 2002). The Barabási-Albert (BA) model (Barabási, 2016) is the most widely known PA-based evolving network model, and serves as the basis for several other models. The BA family of models generally exploits *popularity* as the main dimension of attractiveness which underlies PA phenomenon, and explain the emergence of scale-free structure characterised by heavy-tailed degree distributions commonly found in growing networks. In order to overcome the limitation of Web service discovery due to service isolation, this chapter presents how complex network theory can be exploited to model the service interactions in their ecosystem, thereby globally connect them.

5.1.2 Searchability and Navigability of Complex Networks

This research work is further motivated by key findings in network science that connect certain topological characteristics of complex networks like *small-worldness* and *scale-freeness* with functions such as *searchability* (Rosvall, Grönlund, Minnhagen & Sneppen, 2005; Adamic, Lukose, Puniyani & Huberman, 2001; Watts, Dodds & Newman, 2002; Liben-Nowell, Novak, Kumar, Raghavan & Tomkins, 2005) and *navigability* of complex networks (Boguna et al., 2009; Boguná & Krioukov, 2009; X. Sun & Zhuge,

2014) which we consider important features for effective, user-aware and interactive discovery systems. For instance, the famous Milgram's *"six degrees of separation"* phenomenon and *small-world* paradigm (Travers & Milgram, 1977) revealed something fascinating in the structure of complex network systems−that without a global knowledge of the network topology, an *object* can reach the target node using, on average, 5.2 intermediate nodes, which practically validates the manifestation of social networks as *small-worlds*. Following this interesting finding, researchers have conducted various experiments to show that small-world phenomena exist in various real-world networks including internet, WWW, email network, biological, transportation , citation , and co-author networks (Albert & Barabási, 2002). Real-world networks with small-world property exhibit certain topological properties including *short average path length* and relatively *high clustering coefficient*, and several network generating models exist to model the properties. Recent works (Rosvall et al., 2005; Adamic et al., 2001; Watts et al., 2002; Liben-Nowell et al., 2005) show that small-world and scale-free properties of networks can be exploited to improve the network node's *searchability* – that is, the property of being able to discover a target node efficiently through shortest-path with local information. Similarly, there are algorithms (Boguna et al., 2009; Boguná & Krioukov, 2009; X. Sun & Zhuge, 2014) like *greedy routing* that exploit network properties like clustering coefficient,shortest-path-length and scale-free node degree distributions to improved network *navigability* – a measure that describes efficient transversing from a source node to the next node which is closest in proximity to a desired target node, and stop if the target node is reached. We aim to replicate these developments in our complex network-based Web service discovery framework to improve Web service discoverability and enable effective user-interaction with Web service discovery system. We use the two different types of network models that can generate network structures and properties− like scale-free, small-world property and high clustering – similar to those of real-world networks to construct Web service networks with high navigation

efficiency (E) and success ratio (E_r) (Seguin et al., 2018).

5.1.3 Exploiting Web-API's Functionality and Sociability for Its Discovery

Web-API's *functionality* is a set of functional properties that describe its operation signatures including its input/output schema usually captured in the service textual descriptions, while the Web-API *sociability* is its property or ability to interact with other related Web-APIs (W. Chen et al., 2015a). Service models including Semantic Web services, RESTful APIs only consider service functionality but not their social aspect , and most Web service discovery approaches focused on using Web services functional descriptions to facilitate discovery process and neglect the interactions among the services. Service sociability can be enabled using network models to create social links among services, and several authors have emphasized the importance of such social connections for Web service discovery (Maamar, Faci et al., 2011; Maamar, Hacid & Huhns, 2011; W. Chen & Paik, 2013).

The network-based discovery approach presented in this chapter is based on the assumption that an evolving Web service social network that captures service relationships or interactions, and preserves their social attributes such as service popularity in their ecosystem, can be constructed to facilitate continuous service publication and discovery. Therefore, by connecting Web services that are independently published into a global social network, both social and functional properties of Web services can be exploited to improve service discoverability.

5.1.4 Motivation Example

Consider a new service consumer, who wants to leverage different Web-APIs from different domains (say *Location, Jobs, Message*) to create a mashup that allows users

to get current job vacancies that are close to a specified location, and display the job vacancies as text message or notification on a device. Assume that the user enters different functional requirements to discover most suitable set of APIs for the mashup; First, the user query ProgrammableWeb directory by entering *" recent jobs vacancies close to location and message"*, the search results was 0, then, the user conduct another search with only the keywords *" recent job vacancy location message"*, the search result still returns nothing. Then, user decided to use a combination of two key terms at a time, using *" recent job"*, *"job vacancies"*, *"job location"*, *"job message"*. 5, 4, 31, 8 sets of Web-APIs were returned for each of the query respectively.
The following observations can be deduced from the simple search procedures and the results:

- The search results of the first and second queries reflect the impact of query length on the performance of ProgrammableWeb especially for complex user's requirement , that is, ProgrammableWeb performance decreases with increase query length.

- There is clear vocabulary gap between the user's requirements (queries) and the service descriptions created by the providers. Many relevant Web APIs are missing with respect to each request, and many irrelevant Web-APIs were included in the search results.

- Manual long search and filtering through results maybe be required for users to get results close to their requirements. User may not know what is the best choice of words or combination of words to use in the query.

Other problem with the above illustration may include the lack of quality description, and the subjectivity of service provider functional descriptions (Zhong et al., 2016). For instance, in ProgrammableWeb, mashup *"AllJobs.biz"* have 5 component Web-APIs

- (*indeed, GeoNames, Simply Hired Jobs, Bing Maps Locations, Juju Publisher*) that captured user's initial functional requirements (say we use the first query to search the ProgrammableWeb: *"**recent jobs vacancies** close to **location** and **message**"*). However, the mashup was not included in the first and second search results, even-though it is very relevant to the user's requirement. The reason behind this shortcoming lies in the provider's description of the mashup : *"AllJobs.biz is a job search aggregator which lists **latest job vacancies** from a multitude of industries."*. Though the mashup includes location required APIs (Geoname and Bing Maps) and other relevant functionalities , the PW discovery system could not return it because of the vocabulary differences between the mashup's description and the search terms. Moreover, when queries *"jobs vacancies"* and *"latest jobs vacancies"* were used, the mashup came up because the terms in user's query were directly included in the providers description. Replacing "latest" with "recent" also provides no result for query *"latest jobs vacancies"*. These issues are addressed in this chapter.

5.1.5 Problem Formulation

Consider a typical mashup-oriented Web-APIs discovery problem defined as follows : Given a user's mashup query or request (usually a less detailed, ad-hoc functional description), the mashup-oriented Web-APIs discovery algorithm need to receive the user query and return ranked list of Web-APIs related to the functional elements described in the query, where higher ranked Web-APIs in the list are potential candidates for composing mashup that meets the user request. The mashup-oriented discovery framework can be deployed via an active Web domain or an online repository like the ProgrammableWeb where both Web-API users and providers can interact with the framework through a Web interface. Based on this description, the network-based, mashup-oriented discovery task can be represented as a typical information retrieval

task that includes ranking and *link-as-you-go* as sub-tasks, where a newly published service is connected with other related services mainly based on its functional and social properties.

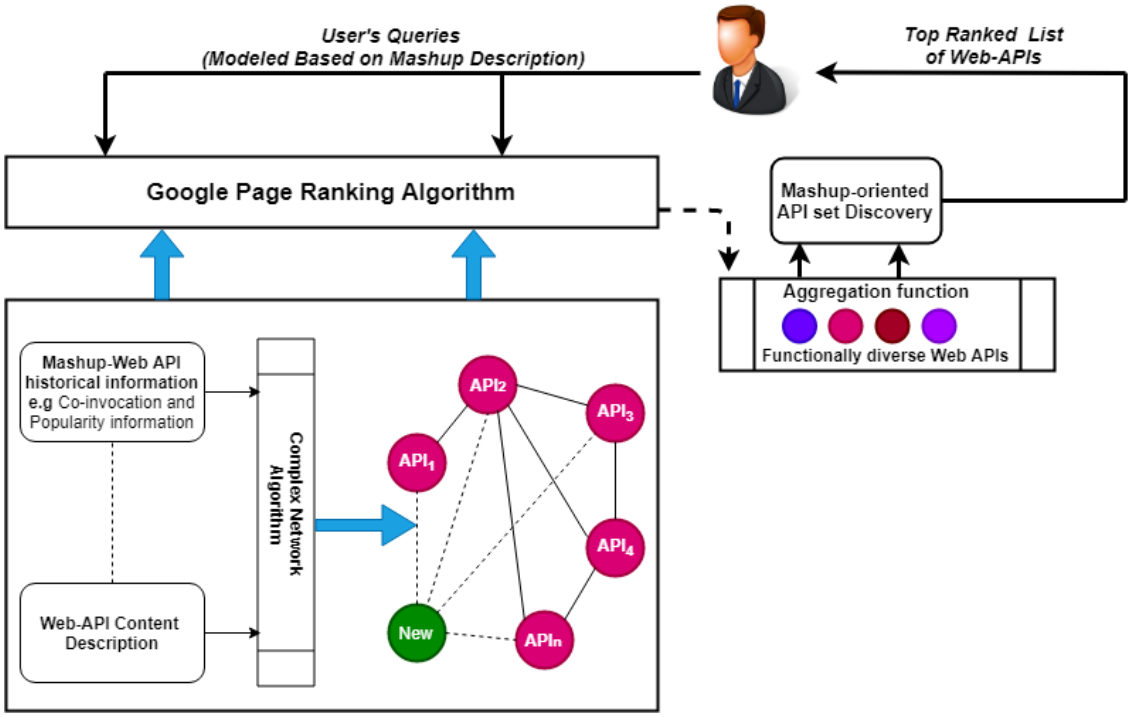


Figure 5.1: Complex network-based Web-API Discovery System using *Google Custom Search API*.

In order to solve this problem, an evolving, popularity-similarity, network-based Web-API discovery framework is proposed. Figure 5.1 shows the components of the proposed framework. These include 3 key components: (i) the complex network component with popularity (preferential attachment) and similarity as the network drivers and the Web-API textual profiles as node attributes. (ii) Incorporating page ranking service for node ranking process (iii) Aggregating diverse Web-APIs to facilitate mashup-oriented API discovery. Further descriptions of these components and the workflow of the framework based on Algorithm 5.1 are presented as follows:

Algorithm 5.1: Mashup-Oriented Web-API Discovery Algorithm

Input:

1: $\mathcal{G}(\mathcal{V}, \mathcal{E})$: the Web-API social network
2: $Q_t = \{t_1, \ldots, t_n\}$ the n number of keywords that define the mashup the user wants to compose.
3: k : the number of top ranked APIs that will be selected per $t_i \in Q_t$ search result

Output:

1: $\mathcal{L}_m$: Ranked list of selected component Web-APIs for Mashup m

Procedure

1: **Remove** stop words from Q_t
2: **stem** Q_t
3: $B \leftarrow$ empty array
4: **for** i = 1 to n **do**
5: $R = \mathcal{G} \leftarrow Gcse(t_i)$
6: **append** $R[:k]$ to B
7: **end for**
8: $\mathcal{L}_m \leftarrow Aggregate_k(B)$ using *ranked-choice voting into a single list*
9: **return** $\mathcal{L}_m$

5.2 Data Processing

Both raw and synthetic data are used for the experiment conducted in this chapter. The experiment and analysis conducted in this chapter is based on the data acquired from one of the largest Web-API registry–*ProgrammableWeb*. PW is used as the $testbed$ for the network-based Web-API discovery service. The descriptive raw data of Web-APIs and mashups from the registry, spanning from $June$ 2005 to $January$ 2020. Since the backend database of $ProgrammableWeb$ is publicly inaccessible, only the web pages can be employed for acquiring the data. Hence, Web-APIs data from ProgrammableWeb web pages were crawled. The acquired web pages are then separated into two groups: Web-APIs and mashups. The textual information for each Web-API includes *name*, *descriptions*, *publication date*, and *category* ; Likewise, each mashup

also contains the above metadata plus the list of Web-APIs invoked within it. For the initial pre-processing, the mashup-API interaction matrix was extracted. Redundant mashup points were removed, mashups with no information other than their names were identified and discarded. After the pre-processing, the number of mashups is then reduced to 5882 mashups. Even though there are over 17959 Web-APIs in our dataset, only 1,525 are involved in the mashup-API interaction matrix, which is the total number of APIs invoked by the mashups (i.e. only 8.5% of the total number of Web-API were involved mashup-API interaction matrix). This indicates that only a few Web-APIs are frequently invoked in the mashups with majority of the APIs isolated from this community, therefore are rarely used. Moreover, there are only 11,287 mashup-API interactions which indicates a very low matrix density of 1.26×10^{-3}. About 80% of the Web-APIs involved in the interaction are consumed less than 5 times. These statistics confirm the meagre in the uptake of Web-APIs and the lack of interactions within most related APIs in the ecosystem.

5.2.1 Pseudomashups – Generating synthetic mashups

In order to have quality, objective and highly dense dataset as ground truth for the experimental purpose, synthetic mashups called *Pseudomashups* are generated . Each pseudomashup contains at least 3 component Web-APIs from the original *mashups* dataset with their descriptions generated from the composite functional keywords from each of the components. The *pseudomashups* is constructed by exploiting the historical usage of APIs within mashups, and the features of the original descriptions (for both APIs and mashups) crawled from *ProgrammableWeb.com*. The following procedure describe the step-by-step processes involve in generating the synthetic mashups.

- First, the service interactions in *ProgrammableWeb* are modelled in the form of an affiliation network, which depicts the invocation relationships between mashups

and Web-APIs.

- Then both the mashups invocation distribution and Web-APIs popularity (degree) distribution from the network are extracted, where the Web-APIs popularity distribution is the number of times a particular Web-API is consumed or invoked by mashups (similar to degree distribution).

- An ordered list $\mathcal{L}$ of all Web-APIs in the ecosystem is then created following similar node ordering strategy for the network construction procedure (described in chapter 4).

- The, a weighted sampling process (W. Wu, Li, Chen & Zhang, 2017) is directly applied. where the normalized popularity information is used as the sampling distribution:

  - First, the Web-API degree distribution is normalized to achieve a probability distribution p, where p follows thesame sequence as in the ordered API candidate list $\mathcal{L}$.

  - Then, i number of Web-APIs are chosen from $\mathcal{L}$ based on the p, where i follows the mashup invocation distribution.

- Based on this strategy , it is possible to replicate thesame mashup structure similar to the original data but more comprehensive. Since mashups descriptions are fundamentally generated from their component's descriptions, new descriptions are generated for the each *pseudomashup* by simply merging the textual descriptions of their respective component APIs selected through the sampling process. We were able to create addition 6000 sets of *pseudo-mashups* which were used as *ground-truth* in validating our approach.

- Finally, to remove noise and redundancy, further preprocessing is conducted

on the description of each pseudomashup using natural language preprocessing techniques. First, the tokenization of the descriptions is done and meaningless or redundant terms are discarded. The root terms for each description are retained.

Moreover, instead of directly using the service descriptions for user's query, a more objective mashup queries are generated for the experiment based on the original (since the original dataset is limited and some data points lack quality descriptions). To do this, Latent Dirichlet allocation (Blei et al., 2003) was employed to construct the query for each the *pseudo-mashup*. Technically, the functional requirement of each pseudomashup is modelled as query using LDA by feeding collection of tokens generated from the pseudomashup description documents into a Gibbs sampler with predefined number of topics as described in (Blei et al., 2003), and then generate topic distribution for each document, and also word per topic distribution. This is done based on the assumption that each topic represents a particular functional component of a pseudomashup. Therefore, for each topic assigned to a pseudomashup document, a collection of dominant words (based on the probability assigned to the words) is retrieved. For example, mashup document A with Topic 0 may contain [*0.032"intentionally" + 0.022"left" + 0.021"fork" + 0.021"blank" +0.020"resource" + 0.005"user"* ...]. This shows the top 10 words that contributed to Topic 0 of a document (mashup) with their degree of importance as weight. Finally, the document per word dictionary for each mashup is generated. This represents a map of pseudomashups with respective probabilistic-based generated queries.

5.2.2 Refining APIs descriptions for Web-API network nodes

The performance of keyword-based service discovery systems like ProgrammableWeb usually rely heavily on the quality service descriptions created by the providers (Tapia, Torres & Astudillo, 2011; Zhong et al., 2018). However, service descriptions created

by the providers are usually highly subjective, limited to the providers' opinion of the services functions. This often leads to vocabulary gap between user's service functional requirement (define in form of query) and the providers' original service descriptions, making it difficult for many existing service discovery algorithms to capture potential services relevant to the user's requirement. Service consumers can use combination of terms totally different to the original description to search for a particular service. For instance, a developer may use an informal term *eatery* to search for *restaurant*, while the former will return just a single API in ProgrammableWeb, the latter term will return more that a dozen APIs. The challenge of vocabulary gap is more pronounce when searching for mashups or component services for mashups development. Normally, the description of a particular mashup should reflect the key functional features of the mashup's component Web-APIs. For example, a mashup *SongsNearMe* with description *"a web application that allows a user to discover new music based on what is popular at a local bar and college pub"* would draw components based on the key words **discover, music, location, college, streaming** and maybe **real time**. However, if the description is lacking key terms for the component services, it would be difficult for description-based discovery systems to retrieve the candidate services. Furthermore, mashup profile should be consistent with the current state of the ecosystem. As Web-APIs evolves, their names, functions and invocation history information normally change over time. For instance, in ProgrammableWeb, some Web-APIs either have outdated profiles or have become obsolete. These changes could lead to inconsistency in the information provided in service registries. After the initial pre-processing of the datasets, about 20% mashups still contain information about either depreciated APIs or APIs invoked that have been further decomposed.

Since the proposed evolving network-based, Web-API discovery approach also explores service functional descriptions as one of the nodes attributes used to facilitate node ranking and retrieval, an attempt is made to improve the quality of the descriptions. To

address the discrepancies in the Web-API descriptions and improve the quality, similar probabilistic topic-modelling approach introduced in (Zhong et al., 2016) is employed. The approach is based on the assumption that the mashup descriptions are generated by its component APIs. Following the procedure described in the work, the degree of similarity between each Web-API functional features and the topic feature of the mashups that invoked them is determined. then, the resulting similarity weights is sorted in descending order. Then the Web-APIs descriptions are augmented with closely similar functional information contained in the mashup descriptions where the APIs were invoked .

5.3 Proposed Approach

This section introduces the proposed evolving, complex network-based Web-API discovery framework. First, a typical Web service discovery problem considered in this work is defined, and then the components of framework are described. As shown in Figure 5.1, the proposed Web-API discovery framework consists of three major components: (i) An evolving complex-network of Web-APIs. (ii) the integration of Web-API network with Google search functionality (iii) an aggregation function for combining services with diverse functionalities based on users query. We describe in details of the components of the framework in following subsections:

5.3.1 Network-Based Web-API Discovery with Google Custom Search API

This section demonstrates how to exploit the Web-API social network and semantics of the Web-APIs descriptions to facilitate mashup-oriented Web-API discovery.

The proposed Web-API discovery service prototype utilizes the popularity-similarity

Web-API network to connect all APIs into a global network. The network is constructed following the procedures described in chapter 5 with set of the network parameters that are defined based on the results of the network analysis (described in chapter 4) performed on the Web-API ecosystem datasets. In order to allow users interact with the discovery framework, both the nodes $\mathcal{V}$ – the Web-APIs with their descriptions, and edges $\mathcal{E}$ – as the social interactions among Web-APIs of the network $\mathcal{G}$, are published via an active Web domain. Then, the capability of Google-page ranking feature is Incorporated into the framework through the invocation *Google Custom Search* API ($Gsce$), which is aim at facilitating Web-API nodes ranking of the network. Therefore, user's can search for relevant mashup components on the network using the Google search interface. This simplifies the retrieval and display of the search results.

Algorithm 1 provides the workflow of how each of the components and their associated processes are interconnected. Given a user mashup requirement Q_t with n number of keywords that define the mashup the user wants to compose, after further preprocessing including pruning, stemming and tokenization, the resulting $t_i \in Q_t$, where $0 < i \leq n$, is used to search the network $\mathcal{G}$ via the $Gsce$ interface. For each search process, top k Web-API is retrieved, and stacked sequentially on preceding search result to get an array of list B.

Then, an *aggregation function* inspired by *ranked-choice voting procedure* [4] is defined to generate a single ranked list of candidate Web-APIs for the user's mashup requirement based on frequency of occurrence and the position of each Web-APIs in the ordered lists B. This is done to determine each Web-API position in the final $top-k$, cumulative ranked list $\mathcal{L}_m$ for Q_t. For instance, addressing the problem described in section 5.1.4, if the network-based discovery service is queried using pair terms (t_i, t_{i+1}), (t_i, t_{i+2}) and (t_i, t_{i+3}) where $t_i \in Q_t$, and get $b_1 = \{api_1, api_2, api_3, api_4\}$, $b_2 = \{api_2, api_5, api_6, api_3\}$, $b_3 = \{api_2, api_5, api_6, api_3\}$ as $top-k$ lists for each pair

[4] https://en.wikipedia.org/wiki/Ranked_voting

term query respectively. Then, the following procedure explains how the *aggregation strategy* works:

- First, a weight, which is computed as the inverted-rank of each elements in lists b_1, b_2, b_3, is assigned to each element in the list so as to preserve that rank 1 is best. For the example, the result of the above example will be : b_1 = $\{api_1 : weight$ = $1, api_2 : weight$ = $\frac{1}{2}, api_3 : weight$ = $\frac{1}{3}, api_4 : weight$ = $\frac{1}{4}\}$ and thesame process would be applied to b_2, b_3.

- Then, in order to get the cumulative rank score for each element in the lists, the mean weight for each API across all lists (b_1, b_2, b_3) is computed. For the example, the cumulative rank for each element in B for the example above would be $\{api_1 : weight$ = $1.5, api_2 : weight$ = $1.5, api_3 : weight$ = $0.58, api_4 : weight$ = $1.25, api_5 : weight$ = $0.5, api_6 : weight$ = $0.33, api_7 : weight$ = $0.33, api_8 : weight$ = $0.25\}$

- Then, elements in the list are ranked based on each API cumulative weight score to create an ordered list of aggregate Web-API. The list : $(api_1, api_2), api_4, api_3, api_5,$
$(api_6, api_7), api_8$

- For rank ties, such as the $(api_1 api_2)$ and (api_6, api_7), the popularity information of the APIs in the list is used as tie breaker. Using thesame rank choice voting procedure, the lowest scoring element from the lists can be eliminated, and then the scores recalculated until there is a winner.

5.4 Experiments and Results

This section reports the extensive experiments conducted to demonstrate the performance of the proposed discovery framework on ProgrammableWeb dataset. Table 5.1

Table 5.1: Summarize Features of the Experimental Dataset.

Statistics	**Values**
Number of Web-APIs	17,959
Number of original mashups	5882
Average number of Web-APIs invoked by Mashups	2.1
Number of unique terms for mashup description	22 ,626
Number of *Pseudomashup* used for the experiments	6000
mashup-API Interaction Matrix Density	1.26×10^{-3}

shows the stats of our datasets. The experiments aim to answer the following three research questions:

1. How does the proposed evolving network-based, Web-API discovery service performs in comparison with the state-of-the-art service discovery methods in both the single, API-oriented discovery and mashup-oriented discovery tasks ?

2. How does the popularity-similarity based evolving network Web-API discovery perform in comparison with random-based Web-API network using thesame *Google page-ranking* feature for both networks?

3. How does m (number of links) parameter setting affects the Web-API discovery performance using thesame schema of the preferential-attachment based evolving Web-API network ?

5.4.1 Evaluation Metrics

Since the proposed Web-API discovery approach returns ranked lists of Web-APIs, commonly used ranked-based evaluation metrics for information retrieval systems (Schröder, Thiele & Lehner, 2011) are used to evaluate the quality of the discovery results and the overall performance of proposed API discovery solution.

Mean Average Precision (MAP@K) :

MAP is the mean of Average Precision. Precision P also known as *true positive accuracy* (Schröder et al., 2011) is the ratio between the discovered APIs in the ranked list that are relevant and the total number of APIs in the retrieved result. Precision estimates the probability that a recommended Web-API is the one preferred for the user's service composition or API request (in the case the pre-defined pseudomashup request). P@K is a precision measured by considering only the top-k elements in the ranked list. Mean Average Precision(MAP) at *top-K*Web-APIs in the ranking list is defined as :

$$MAP@K = \frac{1}{|P_m|} \sum_{m \epsilon P_m} \frac{1}{N_m} \sum_{r=1}^{K} \left(\frac{N_r}{r} . I(r) \right), \tag{5.1}$$

where P_m is the set of mashups in the test set (the *pseudomashups*). N_m represent the number of component APIs of *pseudomashup* m , and $|P_m|$ is the cardinality of P_m. N_r denotes the number of component APIs of m that occurred in the top r APIs of the ranking list. $I(r)$ is a function equalling to 1 when the API at ranking position r is a component API of *pseudomashup* m, otherwise equal to 0. For each pseudomashup m, we run the description as user's query, we get N_r number of actual Web-APIs components of m in the top r ranking list.

Normalized Discounted Cumulative Gain (NDCG@N)

Normalized Discounted Cumulative Gain (NDCG) (Järvelin & Kekäläinen, 2002) measures the quality of a ranked list based on the relevance of the items, discounted by the $\log$ of their position on the ranking list. In this case, the usage of the metric is based on two assumptions: (i) that *highly relevant* Web-APIs is more useful than *marginally relevant* Web-APIs. (ii) that the lower the rank position of a relevant Web-API in the ranking list, the less useful it is for the service consumer, since it is less likely to be

examined by the consumer. Therefore, gain or usefulness of Web-API is discounted at every lower rank, penalizing relevant APIs at the lower ranks. We evaluate the Web-API retrieval results by computing the NDCG at *top-K* which considers the subset of a discovery result from rank 1 to N using the following equation:

$$NDCG@K = \frac{1}{|P_m|} \sum_{m \epsilon P_m} \frac{1}{S_m} \sum_{r=1}^{K} \frac{2^{I(r)} - 1}{\log_2(1+r)} \tag{5.2}$$

where S_m represents the ideal maximum Discounted Cumulative Gain (DCG) score that can be achieved for m . where r is the position of a Web-API in the search result ranked list and $I(r)$ is the relevance score of the r_{th} result in Web-API on the ranking list, and $0 \leq I(r) \leq 1$. For our experiment, the value $I(r)$ is taken to be the normalized popularity value of API in r_{th} position. The higher values of *NDCG* indicates better ranked lists and therefore better correctness.

5.4.2 Baseline Methods

The performance of the proposed evolving complex network, Web-API discovery service is compared with five related information retrieval baseline methods:

- **TF-IDF** (B. Xia et al., 2014) : For this approach , Web-APIs are discovered or recommended based on the similarities between the APIs in the API repository and the mashup request provided by the service consumer. The term frequency of each service document is extracted, and the Term-Frequency Document (TDM) matrix is created from the Web-API corpus. Then, the matrix is transformed to a TF-IDF weighted matrix. Web-APIs are then recommended based on their TF-IDF weight similarities with the mashup request or query. The similarity can be measure by Kullback-Leibler divergence.

- ***ProgrammableWeb (PW) :*** ProgrammableWeb recommend Web-APIs to users

mainly based on the popularity of the each API, and the functional similarity of the APIs and the user request (similarity between user's requests and the predefined description with tags used in programmableWeb repository).

- ***PoP-K :*** This method discover Web-APIs based on popularity. The popularity of an API is measured by the usage frequency by mashups from historical information. Only the top N Web-APIs in relative the API categories (as in programmableWeb.com) are recommended according to the mashup query. After mapping the user query with its knowledge based, programmableWeb simply calculates the number of times each API is consumed in the mashups and ranks them based on their popularity. The system then recommends Web APIs based on their popularity and the degree of similarity between use's query and Web APIs descriptions.

- **LDA** (C. Li, Zhang, Huai, Guo & Sun, 2013): The LDA-based service discovery approach utilizes probabilistic model to analyse Web-API documents and generate topics-based representation for the Web-APIs. After estimating the model parameters, the relevance score of Web-APIs can be determined with respect to user Web-API request by estimating the similarity between the user request topic distribution and the topic distribution of the Web-APIs. Finally, the recommendation list is generated based on the relevance scores.

- **Random Web-API Network (R-Net) (Barabási, 2016)**: To evaluate the effect of popularity information in our service discovery service, the popularity-similarity based, evolving Web-API network is compared with the randomly generated Web-API network. The $Erds-Rnyi$ random network model discussed in (Barabási, 2016) is used to generate a random-network that connects all Web-APIs. The following steps are followed in generating the random-network of Web-APIs:

1. First, the network is initialized with N isolated Web-API nodes, where N is the total number of Web-APIs in the original ProgrammableWeb dataset.
2. Then, a node pair is selected at random with a random number between 0 and 1. If the number exceeds probability p, the selected node pair is connected with an edge, otherwise, they remain disconnected.
3. Repeat step (2) for each of the $N(N-1)/2$ node pairs.

The same google ranking feature employed in the proposed discovery architecture is also adopted for the random network-based discovery baseline approach to evaluate the impact the popularity and similarity information in the proposed network-based discovery solution.

5.4.3 Results and Analysis

This section discuss the results and analysis of the experiments conducted in this chapter, and provide answers to the research questions.

Performance Comparison (RQ-1)

The effectiveness of the proposed popularity-similarity network-based Web-API discovery approach is examine in this section. The proposed approach is compared with other competing Web service discovery baseline approaches.

The *pseudomashups* dataset is used as the groundtruth for this experiment. Using the *pseudomashups* descriptions as the service composition request, search request is send through to the Web-API network for relevant component Web-APIs for each user mashup requirements, and then the Average precision of the ranked candidate list is measure against the groundtruth. More specifically, the following observations are made:

Table 5.2: **Web-API Discovery Performance By Different Methods**

Methods	MAP@5	MAP@10	MAP@15	NDCG@5	NDCG@10	NDCG@15
TFIDF	0.5845	0.5839	0.5771	0.5679	0.5696	0.5703
LDA	0.5865	0.5792	0.5769	0.5735	0.5744	0.5940
PoP-K	0.5882	0.5824	0.5786	0.5771	0.5784	0.5789
PW	0.6004	0.6185	0.6155	0.6155	0.6149	0.6112
PoP-API Net	**0.6213**	**0.6184**	**0.6155**	**0.6079**	**0.6109**	**0.6127**

- Table 5.2 reports the performance comparisons of the experimental results with a step 5 (i.e. top-k= 5/10/15) for the competing approaches on the evaluation metrics. Clearly, the proposed approach exhibits improvements (in terms of precision and usefulness of the discovered Web-APIs to mashup requests) over the competing services discovery baselines across the ranking positions. The popularity-driven Web-API network coupled with the *Google* page ranking feature significantly improves the relevance of candidate Web-APIs that were retrieved and the precision with respect to the groundtruth.

- The MAP and $NDCG$ results for K at 3, 5, 7, 10, 15 are shown in Figures 5.2 and 5.3 respectively, in all instance, the proposed Web-API network-based discovery service gives quite good results, while all the baseline methods are sometimes seriously inefficient especially for the higher K values. For instance, the relevance and usefulness of discovered Web-API in the ranked results (which is measured using the NDCG metric) with respect to mashup query decrease significantly for all the baseline approaches at top 10 and 15, but increases for the proposed approach.

  Because ProgrammableWeb includes functionality based categorisation and gives preference to popular APIs in each categories when recommending APIs, it

performs relative well for simple queries but poorly for complex mashup requests (queries with multiple functional requirements). The advantage of the evolving, complex network-based Web API discovery service to PW indicates that the social dimension introduced to the discovery framework and API popularity information have significant impact in increasing discoverability of the Web services. TF-IDF shows poor performance because it only considers functional keywords and cannot capture semantic similarity between two different keywords. Moreover, the mean average precision (MAP) of the proposed significantly higher than that of LDA, TF-IDF, PoP-K and ProgrammableWeb. The proposed network-based discovery approach outperforms vanilla LDA-based discovery approach because it does not only rely on the semantics of the Web-API node descriptions but also incorporates other attributes like popularity, network/social features and node semantics/functionality similarity .

In summary, the performance gain of the proposed approach can attributed to the ability of the approach to exploit both the functional and social relationships between Web-APIs.

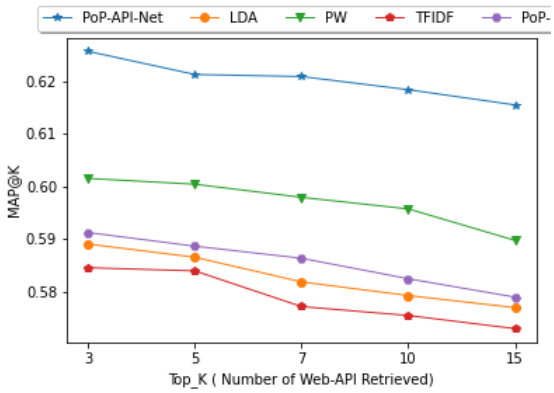


Figure 5.2: Mean average precision for *top-K* APIs discovery results

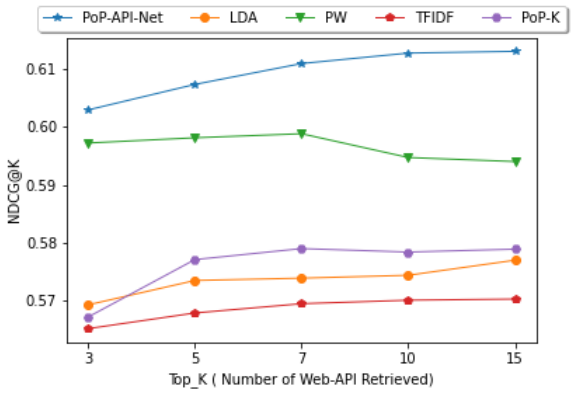


Figure 5.3: Normalized discounted cumulative gain for *top-K* APIs discovery results

Impact of Preferential Attachment on Service Discovery (RQ-2)

As discussed earlier, one of the key driving mechanism of the proposed Web-API evolving complex network is preferential attachment, where Web-APIs connect preferentially to popular or already well-connected APIs. In this part, the impact of popularity (simulated by PA mechanism) is evaluated from discovery perspective. The analysis of how the popularity information introduced into the network construction affects the Web-API ranking process of the ranking function incorporated with the Web-API network-based discovery framework is conducted. This is done by comparing the discovery performance of a random Web-API network to the proposed popularity-based, evolving Web-API network. Using thesame number of Web-API nodes and attributes, a random Web-API network (R-Net) is constructed, following the $Erds - Rnyi$ random network procedures as described in section 5.4.2 and use thesame Google-page ranking feature as our proposed approach for node ranking.

As shown in Figures 5.4 and 5.5, the popularity-based evolving Web-API network performs better than the random Web-API network in terms of precision and relevance of discovery results with respect to the mashup requirement. This is because : (i) the few very high degree Web-API nodes (popular Web-APIs that are most known and frequently consumed) seem to be at the center of everything, while the low degree nodes connect to the rest of the population through their high degree neighbors. From the results, it can be observed that the node ranking feature considers both the functional descriptions and popularity information in ranking the candidate Web-APIs for each query. The more connection a candidate Web-API has and the closer it is to the mashup query, the higher is rank position in the candidate list. It is worth nothing that the popularity information does not only play a part in the automated discovery process but also helps users in interacting with the Web-API network. For instance, users can further *zoom-in* into the *hubs* (popular nodes) in the network and explore the Web-APIs

connected to them (similar to surfing the WWW by following interesting links).

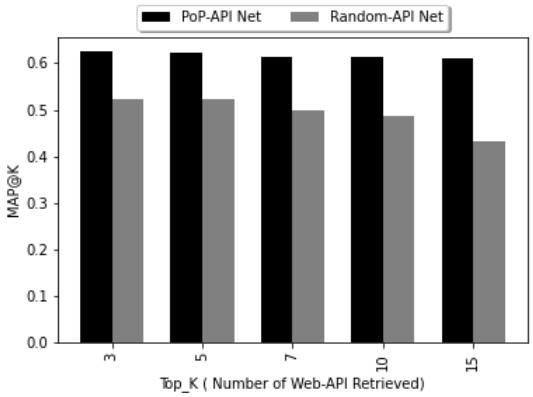


Figure 5.4: MAP@K for Pop-API Network vs. Random-API Network

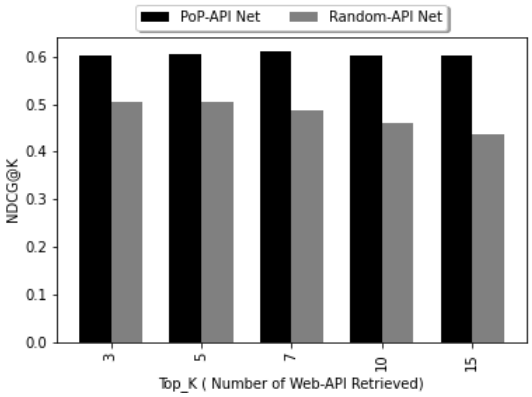


Figure 5.5: NDCG@K for Pop-API Network vs. Random-API Network

Impact of parameter m (RQ-3)

For this experiment, the impact of parameter m on the discovery performance of the evolving Web-API network is evaluated by simply varying the value of m from 1 to 10. Recall that parameter m represents the number of links added per node at each time step of the network growing process. As value of parameter m increases, the average number of links acquired by each Web-API node (average degree) in the network also increases. Figures 5.6 and 5.7 show the discovery performances of the proposed approach with varying m.The results show that a moderate increase in value of parameter m (from 1 to 5) improves the discovery performance of our approach. However, at m = 10, the precision/usefulness of the results begins to drop. This is because as m increases the probability for Web-APIs nodes with a higher degree to interact with other nodes becomes higher, that is, the influence of the popular nodes in the network becomes more apparent, and the average degree for all nodes in the network increases resulting to improvement in local connectivity. However, when the value of is set to m to 10, the precision of the discovery results decreases due to the noise introduced by the high value m. Moreover, as the value m increases, the network navigation becomes more

difficult, and users will have more links and candidate Web-APIs to explore making selection of desired or relevant Web-APIs more challenging.

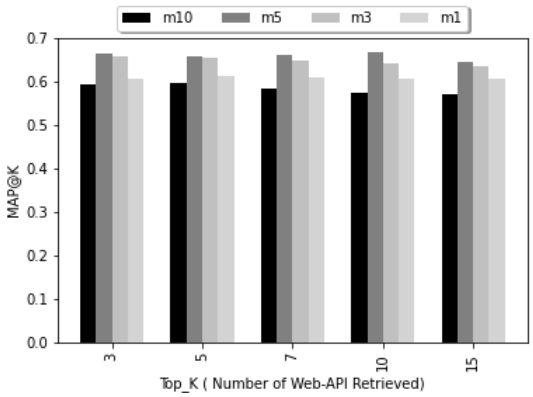


Figure 5.6: Effect of parameter m on *MAP*

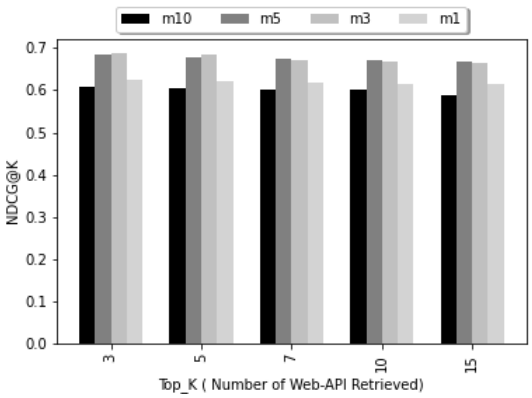


Figure 5.7: Effect of parameter m on *NDCG*

5.5 Chapter Summary

The rapid and continual increase in the number and diversity of Web-APIs currently available on the Web have made it more challenging for software developer to find most appropriate Web-APIs to speed-up software development. At the moment, Web-API consumers including mashup developers normally rely on Web-APIs repositories such as *ProgrammableWeb* and *Mashapes* to discover API of their interest. However, these registries are considered ineffective because: (a) Web APIs registered on these registries are in general *isolated*, as they are registered by diverse providers independently and progressively, without considering relevant dynamic information or continuous social interactions that exist among the services, which could influence their discovery (b) they cannot effectively respond to complex, mashup-oriented Web-API requests. Therefore, there is need for a new Web-API discovery approach that can address these challenges and increase Web-API discoverability. Existing approaches have mainly focus on increasing the accuracy of discovery using hybrid of semantic and syntactic approaches

without addressing the fundamental problems that are limiting the API discoverability. Most of these approaches ignored the very important that Web-API social relationship in their discovery. This chapter addressed the above challenges from complex network perspective using the a popularity-similarity Web-API complex network to facilitate API discovery. The chapter presented a complex-network based discovery service for Web service that leverages an online *Google custom search service* for recommending Web-APIs for mashup development. *Google Page-Ranking* feature was used to facilitate node ranking based on term frequency, functionality and node popularity information. An aggregation function was introduced to systematically retrieve diverse Web-APIs to satisfy complex user request with multiple functional requirements. The chapter demonstrated the effectiveness of the proposed complex-network based discovery approach by presenting results of extensive experiments conducted using both synthetic and real-world Web-API datasets. The evaluation results indicated that the proposed connecting Web-APIs into global social Web service network with popularity-similarity complex network can indeed improve API discovery. This is evident in the performance gain of the proposed approach with respect to other competing information retrieval approaches. The performance edge of the proposed approach with respect to the baseline approaches can be attributed to introduction of the social dimension into the discovery process, and the ability of the approach to exploit both the functional and social relationships between Web-APIs.

Chapter 6

Conclusion and Future Directions

Web-API have had a huge influence on the Web in terms of fostering a worldwide distributed service economy. Despite their impressive development, their Web adoption has been much lower than expected. Poor adoption has been attributed to service isolation, a lack of social ties among related services, insufficient trade-offs between the expressivity and semantics of service descriptions, and poor scalability, which results in exponentially increased search time in vast search areas. This thesis addressed various challenges associated with Web-API discovery from complex network perspective. It propose the used of network analysis technique and network models to study existing Web service ecosystem, and connect all Web-APIs into a global social network. The resulting network can then utilized to incorporate Web-APIs social dimension into their discovery application; an ingredient that have been missing in the existing discovery solutions. Specifically, the thesis propose linking isolated Web-API functional islands into a global social API network to enable exploration of Web-API social properties in Web service applications, and introduce a network-based discovery service that exploits the network, which demonstrate the effectiveness of the proposed linked-as-you go complex-network-based discovery service.

In general, the contributions of this thesis can be summarized into three folds; first, in

Chapter 3 of the thesis presented a method for investigating the drivers of a typical Web service ecosystem , and study its topological and social characteristics using network analysis technique with statistics. Second, in Chapter 4, the thesis presented the proposed evolving Web-API complex networks with details of their construction and evaluation procedures. And in Chapter 5, the thesis presented the complex-network-based Web-API discovery framework and demonstrated the effectiveness of the framework in comparison to the conventional Web-API discovery approach. This chapter concludes the thesis by summarizing the thesis contributions, the key results and outlining some issues for future research. Section 6.1 provides a detailed discussion on the contributions of this thesis. Section 6.2 discusses the limitations and possible improvements of proposed research. Section 6.2 also presents an overview of future research directions.

6.1 Thesis Contributions

The research emphasized four top-level challenges associated with the current Web service discovery approaches. These challenges include: i) the isolation of Web services in their ecosystem; ii) the poor scaling mechanism for the system ; iii) the lack of social social relationships among related Web services; and iv) the neglect of Web service social dimension in their discovery solutions. These research challenges are addressed by defining fine-grain research questions including: i) "What" stimulates service-service interactions in Web service ecosystem? and is social network behaviour universally existing in Web service ecosystem? – these questions are addressed by investigating the dimensions of attractiveness in Web service ecosystem, and studying the topology of the system. ii) How to construct a global evolving social networks of Web services such that it preserves the Web service system properties and reflect relevant properties of real-world network systems like internet and WWW? – these questions are addressed from complex network perspective using both network analysis, complex

network theory and other recent findings in network science to study and construct an evolving Web-API social network. And finally, the thesis examine how to exploit the Web service social network to facilitate the discovery and selection of component services for mashups?– To address this question, this thesis proposed a complex-network based Web-API discovery services that exploited properties of the proposed Web-API network to improve Web-API discoverability . The main contributions of this thesis are summarized below.

6.1.1 A Complex Network Analysis of Web-API ecosystem

From complex network perspective, the Web-API ecosystem (as used in this thesis) consist of Web-APIs and evolves as a complex network system. Multiple Web-APIs can be quickly composed into a Webpage or application called $mashup$. This shortened software development life cycle leads to the formation of the ecosystem, where new APIs and mashups emerge, some old ones perish, and Web-API vendors and developers collaborate to develop innovative software solutions. The perishing of some existing Web-APIs and the emergence of new ones coupled with their dynamic collaborations drive the evolution of this service ecosystem over time. The relationships among APIs and mashups have become increasingly complicated and dynamic as the ecosystem has grown over time, much like those in a complex network. As a result, complex network analysis can be an effective technique for examining the ecosystem's drivers including its topological, static, and dynamical characteristics.

Chapter 3 presented and discussed a complex-network analysis oriented methodology for investigating the topological and dynamical mechanism of the ProgrammableWeb registry (as a template for Web service ecosystem). The Chapter presented a comprehensive exploration of the ecosystem, including an examination of the topology in order to gain a clear knowledge of the current usage pattern and the use of dynamic metrics

to track the ecosystem's evolution. First, the topology of the Web-API ecosystem was studied by investigating mashups and Web-APIs interactions, and analyzing their popularity distributions. The analysis followed three key steps: *visualization*, *model fitting*, and comparison with existing *classical models* including Power-law, Poisson, exponential and log-normal distributions. All the metrics used helped established that the Web-API ecosystem shares similar topological and dynamical properties as the well-known , related real-world network systems like Internet and WWW. For example properties such as small-wordiness commonly found in the real-world complex network systems are also found in ecosystem. Second, for the investigation of the dynamical drivers of the ecosystem, the Chapter presented an approach to quantitatively measure the dimension of attractiveness within the ecosystem: *Preferential Attachment* and *Similarity*. Both dimensions are deemed as the drivers of many real-world networks and were also established to be the key drivers of the Web-API ecosystem. The significant power-law distribution of mashups number per Web-APIs established that there is preferential attachment for the API selection and consumption, confirming that API consumers tend to select the popular Web-API more. This is attributed to the fact that API users often believe that by reusing popular APIs, they may learn from previous usage and create new mashups with potentially higher quality. For link formation and dynamical/evolving mechanisms, *preferential attachment* and the pairwise similarity between Web-APIs were measured and established as significant drivers of API interactions and link formation .

6.1.2 Constructing and Evaluating Web-API Networks

Chapter 4 presented the construction and evaluation of the proposed evolving Web-API networks with clear theoretical backgrounds and step-by-step algorithmic descriptions of how to grow the networks. The Chapter addressed key issue of Web-API *isolation*

and enhancing API sociability in their ecosystem by connecting all Web-APIs into a global social network. Three API networks ; i.) Popularity-based Web-API network; ii.) Fitness-based Web-API network and iii.) Popularity-Similarity Web-API network; are constructed based on the identified dimensions of attractiveness in Web-API ecosystem, which include *Popularity* (simulated by preferential attachment), Similarity (captured by API functional similarity) , and Fitness (computed using random-walk as the correlation values of Web-APIs in the ecosystem). The networks are evaluated using network and system metrics. The network metrics are used to analyse the topological properties of the API networks, and the properties are then compared with that of the real-network system including that of the real world Web-API ecosystem to validate preservation of specific properties of the API ecosystem. The Web-API networks proprieties are mapped with typical Web-API ecosystem properties (*discoverability* and *navigability*) using ProgrammableWeb as case study. End-user activities in ProgrammableWeb such as *searching* , *browsing* and *visualization* are then connected with the API network properties. The findings of the analysis indicated that the proposed evolving Web-API networks not only solved the isolation problem associated with a typical Web-API ecosystem but they also exhibited common topological properties found in real world networks system like WWW, internet, biological and social networks, which are later exploited in API discovery in Chapter 5. The Web-API evolving networks preserved the system and network properties of the API ecosystem (the ProgrammableWeb modelled as Affiliation network). The PSO-based, Web-API network enables the integration of functional similarity into the network construction and thus induces properties such as high navigability and clustering coefficient which can be exploited in API discovery application.

6.1.3 Application of Web-API Network in API Discovery

Chapter 5 presented and discussed the application of the evolving Web-API network for Web-API discovery. The Chapter proposed a complex-network-based Web-API discovery service to demonstrate how the evolving Web-API network can be applied to facilitate service discovery process. For this purpose, a prototype evolving complex network-based, mashup-oriented Web-API discovery service is designed with detailed architectural and algorithmic description provided. The proposed discovery framework contains three key components: (i) the complex network component with popularity (preferential attachment) and similarity as the network drivers and the Web-API textual profiles as node attributes. (ii) Incorporating page ranking service for node ranking process (iii) Aggregating diverse Web-APIs to facilitate mashup-oriented API discovery. The experimental results indicated that the social network-based service discovery solution can indeed be used to improve service discoverability.

6.2 Limitations and Future Direction

Although the solutions mentioned above effectively met the research objectives raised in this thesis, there are certain limitations that could be addressed in the future. This section outlines those limitations and some of the prospective enhancements for future research on the offered solutions.

- **Enhancing Complex-Network-Based Web-API Discovery Performance with User's Feedback:** For the network-based Web-API discovery approach , one limitation is that the approach did not include side information such as API user's feedbacks (either negative or positive), and context information, and QoS that might be relevant to improve API discovery service performance. Incorporating

this information could further enhance the the quality of API discovery application performance. In the future, interactions between Web-API users might be examined and a trust model can be incorporated to exploit the effect of user's feedbacks to improve the quality of Web-API link formation for discovery purpose. Another path to examine in the future study is to model the social influence on the Web-API network and identify representative nodes in order to get insight into the network. Another area of research is to come up with some methods for defining service requirements for the service-based economy on a global basis.

- **Exploiting Community Formation in Web-API Network for Discovery Process:** In the future, community structure in Web-API complex network would be explored to advance the network application with advanced features such as community-awareness introduced into the discovery solution.

- **Ranking Web-API in Web-API Social Networks with Motif-based PageRank:** The ranking and API aggregation approach used in the thesis is limited in terms of API result indexing, in the future, a new ranking approach that better explore the network structure to facilitate discovery would be considered. A motif-based ranking approach (Zhao, Xu et al., 2019) would be explored to investigate how the network structure influences the discovery performance.

References

Adamic, L. A., Lukose, R. M., Puniyani, A. R. & Huberman, B. A. (2001). Search in power-law networks. *Physical review E*, *64*(4), 046135.

Adeleye, O., Yu, J., Yongchareon, S. & Han, Y. (2018). Constructing and evaluating an evolving web-api network for service discovery. In *International conference on service-oriented computing* (pp. 603–617).

Adeleye, O., Yu, J., Yongchareon, S., Sheng, Q. Z. & Yang, L. H. (2019). A fitness-based evolving network for web-apis discovery. In *Proceedings of the australasian computer science week multiconference* (pp. 1–10).

Adeleye, O., Yu, J., Yongchareon Yongchareon, S., Han, Y. & Sheng, Q. (2020). Complex network-based web service for web-api discovery. In *Proceedings of the australasian computer science week multiconference* (pp. 1–10).

Agarwal, N., Sikka, G. & Awasthi, L. K. (2020). Web service clustering approaches to enhance service discovery: A review. In *The international conference on recent innovations in computing* (pp. 23–35).

Akkiraju, R., Farrell, J., Miller, J. A., Nagarajan, M., Sheth, A. P. & Verma, K. (2005). Web service semantics-wsdl-s.

Albert, R. & Barabási, A.-L. (2002). Statistical mechanics of complex networks. *Reviews of modern physics*, *74*(1), 47.

Albert, R., Jeong, H. & Barabási, A.-L. (1999). Internet: Diameter of the world-wide web. *nature*, *401*(6749), 130.

Alessandro, M. & Vittorio, C. C. (2018). Leveraging the nonuniform pso network model as a benchmark for performance evaluation in community detection and link prediction. *New Journal of Physics*, *20*(6), 063022.

Aznag, M., Quafafou, M. & Jarir, Z. (2014). Leveraging formal concept analysis with topic correlation for service clustering and discovery. In *2014 ieee international conference on web services* (pp. 153–160).

Balke, W.-T. & Wagner, M. (2003). Towards personalized selection of web services. In *Www (alternate paper tracks)* (pp. 20–24).

Bano, M., Zowghi, D., Ikram, N. & Niazi, M. (2013). What makes service oriented requirements engineering challenging? a qualitative study. *IET software*, *8*(4), 154–160.

Barabási, A. (2016).
In *Network science, cambridge university press.*

Barabási, A.-L. (2012). Network science: Luck or reason. *Nature*, *489*(7417), 507.

Barabási, A.-L. & Albert, R. (1999). Emergence of scaling in random networks. *science*, *286*(5439), 509–512.

Barabási, A.-L. & Bonabeau, E. (2003). Scale-free networks. *Scientific american*, *288*(5), 60–69.

Barros, A. P. & Dumas, M. (2006a). The rise of web service ecosystems. *IT Professional*, *8*(5), 31–37.

Barros, A. P. & Dumas, M. (2006b). The rise of web service ecosystems. *IT professional*, *8*(5), 31–37.

Bedogne, C. & Rodgers, G. (2006). Complex growing networks with intrinsic vertex fitness. *Physical Review E*, *74*(4), 046115.

Bell, M., Perera, S., Piraveenan, M., Bliemer, M., Latty, T. & Reid, C. (2017). Network growth models: A behavioural basis for attachment proportional to fitness. *Scientific reports*, *7*(1), 1–11.

Bell, M. G. H., Perera, S., Piraveenan, M., Bliemer, M. C. J., Latty, T. & Reid, C. (2017). Network growth models: A behavioural basis for attachment proportional to fitness. *CoRR*, *abs/1702.04046*.

Benesty, J., Chen, J., Huang, Y. & Cohen, I. (2009). Pearson correlation coefficient. In *Noise reduction in speech processing* (pp. 1–4). Springer.

Benslimane, D., Dustdar, S. & Sheth, A. (2008). Services mashups: The new generation of web applications. *IEEE Internet Computing*, *12*(5), 13–15.

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), 19.

Bianchini, D., De Antonellis, V. & Melchiori, M. (2014). Link-based viewing of multiple web api repositories. In *International conference on database and expert systems applications* (pp. 362–376).

Bianconi, G. & Barabási, A.-L. (2001a). Bose-einstein condensation in complex networks. *Physical review letters*, *86*(24), 5632.

Bianconi, G. & Barabási, A.-L. (2001b). Competition and multiscaling in evolving networks. *EPL (Europhysics Letters)*, *54*(4), 436.

Blei, D. M., Ng, A. Y. & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993–1022.

Bobadilla, J., Ortega, F., Hernando, A. & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-based systems*, *46*, 109–132.

Boguná, M. & Krioukov, D. (2009). Navigating ultrasmall worlds in ultrashort time. *Physical review letters*, *102*(5), 058701.

Boguna, M., Krioukov, D. & Claffy, K. C. (2009). Navigability of complex networks. *Nature Physics*, *5*(1), 74–80.

Borgatti, S. P. (2005). Centrality and network flow. *Social networks*, *27*(1), 55–71.

Botangen, K. A. (2020). *Towards the adaptability of service-based systems* (Unpublished doctoral dissertation). Auckland University of Technology.

Bouguettaya, A., Sheng, Q. Z. & Daniel, F. (2014). *Advanced web services*. Springer.

Bouguettaya, A., Singh, M., Huhns, M., Sheng, Q. Z., Dong, H., Yu, Q., ... others

(2017). A service computing manifesto: the next 10 years. *Communications of the ACM*, *60*(4), 64–72.

Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H. F., ... Winer, D. (2000). W3c note: Simple object access protocol (soap) 1.1. *World Wide Web Consortium*.

Bullmore, E. & Sporns, O. (2009). Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature reviews neuroscience*, *10*(3), 186–198.

Caldarelli, G., Capocc, A., Rios, P. D. L. & Munoz, M. A. (2002). Scale-free networks from varying vertex intrinsic fitness." physical review letters. , *258702*, 89.25.

Caldarelli, G., Capocci, A., De Los Rios, P. & Munoz, M. A. (2002). Scale-free networks from varying vertex intrinsic fitness. *Physical review letters*, *89*(25), 258702.

Cao, B., Liu, X., Rahman, M. M., Li, B., Liu, J. & Tang, M. (2017). Integrated content and network-based service clustering and web apis recommendation for mashup development. *IEEE Transactions on Services Computing*.

Cao, Y., Wang, G., Jiang, Q. & Han, Z. (2006). A neighbourhood evolving network model. *Physics Letters A*, *349*(6), 462–466.

Cardoso, J. (2007). *Semantic web services: Theory, tools and applications: Theory, tools and applications*. IGI Global.

Carpineto, C. & Romano, G. (2012). A survey of automatic query expansion in information retrieval. *Acm Computing Surveys (CSUR)*, *44*(1), 1–50.

Carstens, B., Jensen, M., Spaniel, M. & Hermansen, A. (2017). *Vertex similarity in graphs using feature learning (2017).*

Casati, F. & Shan, M.-C. (2001). Dynamic and adaptive composition of e-services. *Information systems*, *26*(3), 143–163.

Cassar, G., Barnaghi, P. & Moessner, K. (2013). Probabilistic matchmaking methods for automated service discovery. *IEEE Transactions on Services Computing*, *7*(4), 654–666.

Cervantes, H. & Hall, R. S. (2003). Automating service dependency management in a service-oriented component model. In *Icse cbse workshop.*

Channabasavaiah, K., Holley, K. & Tuggle, E. (2003). Migrating to a service-oriented architecture. *IBM DeveloperWorks*, *16*, 727–728.

Chattopadhyay, S. & Murthy, C. (2017a). Generation of power-law networks by employing various attachment schemes: Structural properties emulating real world networks. *Information Sciences*, *397*, 219–242.

Chattopadhyay, S. & Murthy, C. A. (2017b). Generation of power-law networks by employing various attachment schemes: Structural properties emulating real world networks. *Inf. Sci.*, *397*, 219–242.

Chen, F., Li, M., Wu, H. & Xie, L. (2017). Web service discovery among large service pools utilising semantic similarity and clustering. *Enterprise Information Systems*, *11*(3), 452–469.

Chen, F., Lu, C., Wu, H. & Li, M. (2017). A semantic similarity measure integrating

multiple conceptual relationships for web service discovery. *Expert Systems with Applications*, *67*, 19–31.

Chen, L., Wang, Y., Yu, Q., Zheng, Z. & Wu, J. (2013). Wt-lda: user tagging augmented lda for web service clustering. In *International conference on service-oriented computing* (pp. 162–176).

Chen, L., Yang, G., Zhu, W., Zhang, Y. & Yang, Z. (2013). Clustering facilitated web services discovery model based on supervised term weighting and adaptive metric learning. *International journal of Web engineering and technology*, *8*(1), 58–80.

Chen, W. & Paik, I. (2013). Improving efficiency of service discovery using linked data-based service publication. *Information Systems Frontiers*, *15*(4), 613–625.

Chen, W., Paik, I. & Hung, P. C. (2015a). Constructing a global social service network for better quality of web service discovery. *IEEE transactions on services computing*, *8*(2), 284–298.

Chen, W., Paik, I. & Hung, P. C. K. (2015b). Constructing a global social service network for better quality of web service discovery. *IEEE Trans. Services Computing*, *8*(2), 284–298.

Chen, W., Paik, I. & Yen, N. Y. (2017). Discovering internal social relationship for influence-aware service recommendation. *Multimedia Tools Appl.*, *76*(18), 18193–18220.

Chen, Y. (2018). *Service-oriented computing and system integration.* Kendall Hunt Publishing, Dubuque, IA.

Choromański, K., Matuszak, M. & Mikisz, J. (2013). Scale-free graph with preferential attachment and evolving internal vertex structure. *Journal of Statistical Physics*, *151*(6), 1175–1183.

Christensen, E., Curbera, F., Meredith, G., Weerawarana, S. et al. (2001). *Web services description language (wsdl) 1.1.* Citeseer.

Clauset, A., Shalizi, C. R. & Newman, M. E. J. (2009). Power-law distributions in empirical data. *SIAM Review*, *51*(4), 661–703.

Cohen, R., Havlin, S. & Ben-Avraham, D. (2003). Structural properties of scale free networks. *Handbook of graphs and networks*, *4*.

Cong, Z., Fernandez, A., Billhardt, H. & Lujak, M. (2015). Service discovery acceleration with hierarchical clustering. *Information Systems Frontiers*, *17*(4), 799–808.

Crasso, M., Zunino, A. & Campo, M. (2011). A survey of approaches to web service discovery in service-oriented architectures. *Journal of Database Management (JDM)*, *22*(1), 102–132.

Daniel, F. & Matera, M. (2014a). *Mashups: concepts, models and architectures*. Springer.

Daniel, F. & Matera, M. (2014b). Quality in mashup development. In *Mashups* (pp. 269–291). Springer.

De Bruijn, J., Bussler, C., Domingue, J., Fensel, D., Hepp, M., Kifer, M., ... others (2005). Web service modeling ontology (wsmo). *Interface*, *5*(1), 50.

Dillon, T. S., Wu, C. & Chang, E. (2007). Reference architectural styles for service-oriented computing. In *Ifip international conference on network and parallel*

computing (pp. 543–555).

Dong, X., Halevy, A., Madhavan, J., Nemes, E. & Zhang, J. (2004). Similarity search for web services. In *Proceedings of the thirtieth international conference on very large data bases-volume 30* (pp. 372–383).

Dorogovtsev, S. N. & Mendes, J. F. (2002). Evolution of networks. *Advances in physics*, *51*(4), 1079–1187.

Duan, L. & Tian, H. (2017). Collaborative web service discovery and recommendation based on social link. *Future Internet*, *9*(4), 63.

Dumais, S. T. (2004). Latent semantic analysis. *Annual review of information science and technology*, *38*(1), 188–230.

Dusseault, L. & Snell, J. (2010). *Patch method for http* (Tech. Rep.). RFC 5789, March.

Easley, D. & Kleinberg, J. (2010). *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press.

Eikermann, R., Look, M., Roth, A., Rumpe, B. & Wortmann, A. (2017). Architecting cloud services for the digital me in a privacy-aware environment. In *Software architecture for big data and the cloud* (pp. 207–226). Elsevier.

Elshater, Y., Elgazzar, K. & Martin, P. (2015). godiscovery: Web service discovery made efficient. In *2015 ieee international conference on web services* (pp. 711–716).

Erdős, P. & Rényi, A. (1961). On the strength of connectedness of a random graph. *Acta Mathematica Hungarica*, *12*(1), 261–267.

Erdos, P., Rényi, A. et al. (1960). On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, *5*(1), 17–60.

Fallatah, H., Bentahar, J. & Asl, E. K. (2014). Social network-based framework for web services discovery. In *Future internet of things and cloud (ficloud), 2014 international conference on* (pp. 159–166).

Feng, Z., Lan, B., Zhang, Z. & Chen, S. (2015). A study of semantic web services network. *The Computer Journal*, *58*(6), 1293–1305.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. & Berners-Lee, T. (1999). *Hypertext transfer protocol–http/1.1*. RFC 2616, june.

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*. University of California, Irvine.

Garriga, M., De Renzis, A., Lizarralde, I., Flores, A., Mateos, C., Cechich, A. & Zunino, A. (2018). A structural-semantic web service selection approach to improve retrievability of web services. *Information Systems Frontiers*, *20*(6), 1319–1344.

Goland, Y., Whitehead, E., Faizi, A., Carter, S. & Jensen, D. (1999). *Http extensions for distributed authoring–webdav* (Tech. Rep.). RFC 2518, IETF, Feb.

Goldberg, Y. & Levy, O. (2014). word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Grady, L. J. & Polimeni, J. R. (2010). *Discrete calculus: Applied analysis on graphs for computational science*. Springer Science & Business Media.

Haas, H. & Brown, A. (2004). Web services glossary. *W3C Working Group Note (11 February 2004)*, *9*, 784–786.

Hadley, M. J. (2006). *Web application description language (wadl).* Sun Microsystems, Inc.

Hafsi, A., Gamha, Y., Njima, C. B. & Romdhane, L. B. (2020). Big-swsdm: Bipartite graph based social web service discovery model. In *International conference on business information systems* (pp. 307–318).

Halevy, A., Nemes, E., Dong, X., Madhavan, J. & Zhang, J. (2004). Similarity search for web services. In *Proceedings of the 30th vldb conference* (pp. 372–383).

Hamilton, W. L., Ying, R. & Leskovec, J. (2017). Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*.

Hao, Y., Zhang, Y. & Cao, J. (2010). Web services discovery and rank: An information retrieval approach. *Future generation computer systems*, *26*(8), 1053–1062.

He, Q., Yan, J., Jin, H. & Yang, Y. (2014). Quality-aware service selection for service-based systems based on iterative multi-attribute combinatorial auction. *IEEE Transactions on Software Engineering*, *40*(2), 192–215.

He, Q., Zhou, R., Zhang, X., Wang, Y., Ye, D., Chen, F., ... Yang, Y. (2017). Keyword search for building service-based systems. *IEEE Transactions on Software Engineering*(1), 1–1.

Hébert-Dufresne, L., Allard, A., Marceau, V., Noël, P.-A. & Dubé, L. J. (2011). Structural preferential attachment: Network organization beyond the link. *Physical review letters*, *107*(15), 158702.

Hobold, G. C. & Siqueira, F. (2012). Discovery of semantic web services compositions based on sawsdl annotations. In *2012 ieee 19th international conference on web services* (pp. 280–287).

Hoschek, W. (2002). Web service discovery processing steps. *ICWI*, *2*, 255–262.

Huang, K., Fan, Y. & Tan, W. (2012a). An empirical study of programmable web: A network analysis on a service-mashup system. In *2012 ieee 19th international conference on web services* (pp. 552–559).

Huang, K., Fan, Y. & Tan, W. (2012b). An empirical study of programmable web: A network analysis on a service-mashup system. In *2012 IEEE 19th international conference on web services, honolulu, hi, usa, june 24-29, 2012* (pp. 552–559).

Huang, K., Fan, Y. & Tan, W. (2014a). Recommendation in an evolving service ecosystem based on network prediction. *IEEE Transactions on Automation Science and Engineering*, *11*(3), 906–920.

Huang, K., Fan, Y. & Tan, W. (2014b). Recommendation in an evolving service ecosystem based on network prediction. *IEEE Trans. Automation Science and Engineering*, *11*(3), 906–920.

Huhns, M. N. & Singh, M. P. (2005). Service-oriented computing: Key concepts and principles. *IEEE Internet computing*, *9*(1), 75–81.

Jalal, S., Yadav, D. K. & Negi, C. S. (2019). Web service discovery with incorporation of web services clustering. *International Journal of Computers and Applications*, 1–12.

Järvelin, K. & Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, *20*(4), 422–446.

Jiang, W., Lee, D. & Hu, S. (2012). Large-scale longitudinal analysis of soap-based and

restful web services. In *2012 ieee 19th international conference on web services* (pp. 218–225).

Jordan, M. I. & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255–260.

Kalaï, A., Zayani, C. A. & Amous, I. (2015). User's social profile–based web services discovery. In *Service-oriented computing and applications (soca), 2015 ieee 8th international conference on* (pp. 2–9).

Kamath, S. & Ananthanarayana, V. (2014). Similarity analysis of service descriptions for efficient web service discovery. In *2014 international conference on data science and advanced analytics (dsaa)* (pp. 142–148).

Kanagasabai, R. et al. (2013). Semantic web service discovery: state-of-the-art and research challenges. *Personal and ubiquitous computing*, *17*(8), 1741–1752.

Kavitha, R. & Anuvelavan, S. (2015). Weather master: mobile application of cyclone disaster refinement forecast system in location based on gis using geo-algorithm. *Int. J. Sci. Eng. Res*, *6*, 88–93.

Keller, U., Lara, R., Lausen, H., Polleres, A. & Fensel, D. (2005). Automatic location of services. In *European semantic web conference* (pp. 1–16).

Kerrigan, M. (2006). Web service selection mechanisms in the web service execution environment (wsmx). In *Proceedings of the 2006 acm symposium on applied computing* (pp. 1664–1668).

Kiefer, C. & Bernstein, A. (2008). The creation and evaluation of isparql strategies for matchmaking. In *European semantic web conference* (pp. 463–477).

Klemm, K. & Eguiluz, V. M. (2002). Highly clustered scale-free networks. *Physical Review E*, *65*(3), 036123.

Klusch, M., Fries, B. & Sycara, K. (2006). Automated semantic web service discovery with owls-mx. In *Proceedings of the fifth international joint conference on autonomous agents and multiagent systems* (pp. 915–922).

Klusch, M., Fries, B. & Sycara, K. (2009). Owls-mx: A hybrid semantic web service matchmaker for owl-s services. *Journal of Web Semantics*, *7*(2), 121–133.

Klusch, M. & Kapahnke, P. (2012). The isem matchmaker: A flexible approach for adaptive hybrid semantic service selection. *Journal of Web Semantics*, *15*, 1–14.

Klusch, M., Kapahnke, P., Schulte, S., Lecue, F. & Bernstein, A. (2016). Semantic web service search: a brief survey. *KI-Künstliche Intelligenz*, *30*(2), 139–147.

Koch, C. (2005). A new blueprint for the enterprise. *CIO Magazine*, *5*(4), 1–8.

Kokash, N. (2006). A comparison of web service interface similarity measures. In *Stairs* (pp. 220–231).

Kokash, N., Birukou, A. & D'Andrea, V. (2007). Web service discovery based on past user experience. In *International conference on business information systems* (pp. 95–107).

Konstas, I., Stathopoulos, V. & Jose, J. M. (2009). On social networks and collaborative recommendation. In *Proceedings of the 32nd international acm sigir conference on research and development in information retrieval* (pp. 195–202).

Kourtesis, D. & Paraskakis, I. (2008). Combining sawsdl, owl-dl and uddi for semantically enhanced web service discovery. In *European semantic web conference* (pp.

614–628).

Krafzig, D., Banke, K. & Slama, D. (2005). *Enterprise soa: service-oriented architecture best practices*. Prentice Hall Professional.

Kreger, H. (2003). Fulfilling the web services promise. *Communications of the ACM*, *46*(6), 29–ff.

Krioukov, D., Papadopoulos, F., Kitsak, M., Vahdat, A. & Boguná, M. (2010). Hyperbolic geometry of complex networks. *Physical Review E*, *82*(3), 036106.

Kritikos, K. (2008). Qos-based web service description and discovery.

Kritikos, K. & Plexousakis, D. (2009). Requirements for qos-based web service description and discovery. *IEEE Transactions on Services Computing*, *2*(4), 320–337.

Kuck, J. & Gnasa, M. (2007). Context-sensitive service discovery meets information retrieval. In *Fifth annual ieee international conference on pervasive computing and communications workshops (percomw'07)* (pp. 601–605).

Kumara, B. T., Paik, I., Chen, W. & Ryu, K. H. (2014). Web service clustering using a hybrid term-similarity measure with ontology learning. *International Journal of Web Services Research (IJWSR)*, *11*(2), 24–45.

Kumara, B. T., Paik, I., Koswatte, K. R. & Chen, W. (2014). Ontology learning with complex data type for web service clustering. In *2014 ieee symposium on computational intelligence and data mining (cidm)* (pp. 129–136).

Küster, U., König-Ries, B., Klein, M. & Stern, M. (2007). Diane: A matchmaking-centered framework for automated service discovery, composition, binding, and invocation on the web. *International Journal of Electronic Commerce*, *12*(2), 41–68.

Lamine, S. B. A. B., Zghal, H. B., Mrissa, M. & Guegan, C. G. (2017). An ontology-based approach for personalized restful web service discovery. *Procedia Computer Science*, *112*, 2127–2136.

Latora, V. & Marchiori, M. (2001). Efficient behavior of small-world networks. *Physical review letters*, *87*(19), 198701.

Le, D.-N., Nguyen, V.-Q. & Goh, A. (2009). Matching wsdl and owl-s web services. In *2009 ieee international conference on semantic computing* (pp. 197–202).

Le, Q. & Mikolov, T. (2014). Distributed representations of sentences and documents. In *International conference on machine learning* (pp. 1188–1196).

Lee, J., Niko, D. L., Hwang, H., Park, M. & Kim, C. (2011). A gis-based design for a smartphone disaster information service application. In *2011 first acis/jnu international conference on computers, networks, systems and industrial engineering* (pp. 338–341).

Leicht, E. A., Holme, P. & Newman, M. E. (2006). Vertex similarity in networks. *Physical Review E*, *73*(2), 026120.

Li, C., Zhang, R., Huai, J., Guo, X. & Sun, H. (2013). A probabilistic approach for web service discovery. In *2013 ieee international conference on services computing* (pp. 49–56).

Li, J., Zaman, N. & Li, H. (2015). A decentralized locality-preserving context-aware

service discovery framework for internet of things. In *2015 ieee international conference on services computing* (pp. 317–323).

Li, L. & Horrocks, I. (2004). A software framework for matchmaking based on semantic web technology. *International Journal of Electronic Commerce*, *8*(4), 39–60.

Li, Z., He, K., Wang, J. & Zhang, N. (2014). An on-demand services discovery approach based on topic clustering. *Journal of Internet Technology*, *15*(4), 543–555.

Liang, T., Chen, L., Ying, H. & Wu, J. (2014). Co-clustering wsdl documents to bootstrap service discovery. In *2014 ieee 7th international conference on service-oriented computing and applications* (pp. 215–222).

Liben-Nowell, D., Novak, J., Kumar, R., Raghavan, P. & Tomkins, A. (2005). Geographic routing in social networks. *Proceedings of the National Academy of Sciences*, *102*(33), 11623–11628.

Lin, S.-Y., Lai, C.-H., Wu, C.-H. & Lo, C.-C. (2014). A trustworthy qos-based collaborative filtering approach for web service discovery. *Journal of Systems and Software*, *93*, 217–228.

Lizarralde, I., Mateos, C., Rodriguez, J. M. & Zunino, A. (2019). Exploiting named entity recognition for improving syntactic-based web service discovery. *Journal of Information Science*, *45*(3), 398–415.

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.

Lizarralde, I., Rodriguez, J. M., Mateos, C. & Zunino, A. (2017). Word embeddings for improving rest services discoverability. In *2017 xliii latin american computer conference (clei)* (pp. 1–8).

Lu, W., Cai, Y., Che, X. & Lu, Y. (2016). Joint semantic similarity assessment with raw corpus and structured ontology for semantic-oriented service discovery. *Personal and Ubiquitous Computing*, *20*(3), 311–323.

Ludwig, H. & Petrie, C. J. (2006). S ssion summary-"cross cutting concerns". In *Proc. of dagstuhl seminar* (Vol. 5462).

Lyu, S., Liu, J., Tang, M., Kang, G., Cao, B. & Duan, Y. (2014). Three-level views of the web service network: An empirical study based on programmableweb. In *Big data (bigdata congress), 2014 ieee international congress on* (pp. 374–381).

Ma, J., Zhang, Y. & He, J. (2008a). Efficiently finding web services using a clustering semantic approach. In *Proceedings of the 2008 international workshop on context enabled source and service selection, integration and adaptation: organized with the 17th international world wide web conference (www 2008)* (pp. 1–8).

Ma, J., Zhang, Y. & He, J. (2008b). Web services discovery based on latent semantic approach. In *2008 ieee international conference on web services* (pp. 740–747).

Maamar, Z. (2003). Commerce, e-commerce, and m-commerce: what comes next? *Communications of the ACM*, *46*(12), 251–257.

Maamar, Z. & Badr, Y. (2009). Social networks as a service in modern enterprises. In *2009 international conference on the current trends in information technology (ctit)* (pp. 1–5).

Maamar, Z., Faci, N., Wives, L., Badr, Y., Santos, P. & de Oliveira, J. P. M. (2011).

Using social networks for web services discovery. *IEEE internet computing*, *15*(4), 48–54.

Maamar, Z., Hacid, H. & Huhns, M. N. (2011). Why web services need social networks. *IEEE Internet Computing*, *15*(2), 90–94.

Maamar, Z., Mostefaoui, S. K. & Mahmoud, Q. H. (2005). Context for personalized web services. In *Proceedings of the 38th annual hawaii international conference on system sciences* (pp. 166b–166b).

Maamar, Z., Wives, L. K., Badr, Y. & Elnaffar, S. (2009). Even web services can socialize: A new service-oriented social networking model. In *2009 international conference on intelligent networking and collaborative systems* (pp. 24–30).

Maamar, Z., Wives, L. K., Badr, Y., Elnaffar, S., Boukadi, K. & Faci, N. (2011). Linkedws: A novel web services discovery model based on the metaphor of "social networks". *Simulation Modelling Practice and Theory*, *19*(1), 121–132.

MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P. F., Metz, R. & Hamilton, B. A. (2006). Reference model for service oriented architecture 1.0. *OASIS standard*, *12*(S 18).

Manikrao, U. S. & Prabhakar, T. (2005). Dynamic selection of web services with recommendation system. In *International conference on next generation web services practices (nwesp'05)* (pp. 5–pp).

Manning, C. D., Raghavan, P. et al. (2008). *Schü tze h. introduction to information retrieval.* Cambridge University Press Cambridge.

McDonald, D. W. (2003). Recommending collaboration with social networks: a comparative evaluation. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 593–600).

McPherson, M., Smith-Lovin, L. & Cook, J. M. (2001). Birds of a feather: Homophily in social networks. *Annual review of sociology*, *27*(1), 415–444.

Medjahed, B. (2004). *Semantic web enabled composition of web services* (Unpublished doctoral dissertation). Virginia Tech.

Medjahed, B. & Atif, Y. (2007). Context-based matching for web service composition. *Distributed and Parallel Databases*, *21*(1), 5–37.

Metrouh, A. & Mokhati, F. (2013). Social web services discovery: A community-based approach. In *Proceedings of international conference on information integration and web-based applications & services* (pp. 275–279).

Miller, G. A. (1998). *Wordnet: An electronic lexical database*. MIT press.

Mitra, N., Lafon, Y. et al. (2003). Soap version 1.2 part 0: Primer. *W3C recommendation*, *24*, 12.

Mohebbi, K., Ibrahim, S., Khezrian, M., Munusamy, K. & Tabatabaei, S. G. H. (2010). A comparative evaluation of semantic web service discovery approaches. In *Proceedings of the 12th international conference on information integration and web-based applications & services* (pp. 33–39).

Mukhopadhyay, D. & Chougule, A. (2012). A survey on web service discovery approaches. In *Advances in computer science, engineering & applications* (pp. 1001–1012). Springer.

Muniruzzaman, A. (1957). On measures of location and dispersion and tests of

hypotheses in a pare to population. *Calcutta Statistical Association Bulletin*, *7*(3), 115–123.

Muscoloni, A. & Cannistraci, C. V. (2018). A nonuniform popularity-similarity optimization (npso) model to efficiently generate realistic complex networks with communities. *New Journal of Physics*, *20*(5), 052002.

Naim, H., Aznag, M., Quafafou, M. & Durand, N. (2016). Probabilistic approach for diversifying web services discovery and composition. In *2016 ieee international conference on web services (icws)* (pp. 73–80).

Nazir, S., Sapkota, B. & Vitvar, T. (2008). Improving web service discovery with personalized goal. In *International conference on web information systems and technologies* (pp. 266–277).

Newman, M. E. (2001). Clustering and preferential attachment in growing networks. *Physical review E*, *64*(2), 025102.

Newman, M. E. (2003). The structure and function of complex networks. *SIAM review*, *45*(2), 167–256.

Ngu, A. H., Carlson, M. P., Sheng, Q. Z. & Paik, H.-y. (2010). Semantic-based mashup of composite applications. *IEEE Transactions on Services Computing*, *3*(1), 2–15.

Nguyen, K. & Tran, D. A. (2012). Fitness-based generative models for power-law networks. In *Handbook of optimization in complex networks* (pp. 39–53). Springer.

Obidallah, W. J., Raahemi, B. & Ruhi, U. (2020). Clustering and association rules for web service discovery and recommendation: a systematic literature review. *SN Computer Science*, *1*(1), 1–33.

Paliwal, A. V., Adam, N. R. & Bornhovd, C. (2007). Web service discovery: Adding semantics through service request expansion and latent semantic indexing. In *Ieee international conference on services computing (scc 2007)* (pp. 106–113).

Palma, F., Gonzalez-Huerta, J., Moha, N., Guéhéneuc, Y.-G. & Tremblay, G. (2015). Are restful apis well-designed? detection of their linguistic (anti) patterns. In *International conference on service-oriented computing* (pp. 171–187).

Paolucci, M., Kawamura, T., Payne, T. R. & Sycara, K. (2002). Semantic matching of web services capabilities. In *International semantic web conference* (pp. 333–347).

Papadopoulos, F., Kitsak, M., Serrano, M. Á., Boguná, M. & Krioukov, D. (2012). Popularity versus similarity in growing networks. *Nature*, *489*(7417), 537.

Papazoglou, M. (2008). *Web services: principles and technology*. Pearson Education.

Papazoglou, M. P. (2003). Service-oriented computing: Concepts, characteristics and directions. In *Proceedings of the fourth international conference on web information systems engineering, 2003. wise 2003.* (pp. 3–12).

Papazoglou, M. P. & Dubray, J.-j. (2004). A survey of web service technologies.

Papazoglou, M. P., Traverso, P., Dustdar, S. & Leymann, F. (2007). Service-oriented computing: State of the art and research challenges. *Computer*, *40*(11), 38–45.

Patil, N. & Gopal, A. (2010). Ranking web-services based on qos for best-fit search. *International Journal of Computer Science & Communication*, *1*(2), 345–349.

Pautasso, C. (2014). Restful web services: principles, patterns, emerging technologies. In *Web services foundations* (pp. 31–51). Springer.

Pautasso, C., Zimmermann, O. & Leymann, F. (2008). Restful web services vs." big"'web services: making the right architectural decision. In *Proceedings of the 17th international conference on world wide web* (pp. 805–814).

Pham, T., Sheridan, P. & Shimodaira, H. (2015). Pafit: A statistical method for measuring preferential attachment in temporal complex networks. *PloS one*, *10*(9), e0137796.

Pham, T., Sheridan, P. & Shimodaira, H. (2016). Joint estimation of preferential attachment and node fitness in growing complex networks. In *Scientific reports, 6, 32558.*

Pham, T., Sheridan, P. & Shimodaira, H. (2017). Pafit: an r package for the non-parametric estimation of preferential attachment and node fitness in temporal complex networks. *arXiv preprint arXiv:1704.06017*.

Pilioura, T. & Tsalgatidou, A. (2009). Unified publication and discovery of semantic web services. *ACM Transactions on the Web (TWEB)*, *3*(3), 1–44.

Platzer, C. & Dustdar, S. (2005). *A vector space search engine forweb services*. IEEE.

Raj, R. J. R. & Sasipraba, T. (2010). Web service selection based on qos constraints. In *Trendz in information sciences & computing (tisc2010)* (pp. 156–162).

Rich, E. (1979). User modeling via stereotypes. *Cognitive science*, *3*(4), 329–354.

Richardson, L. & Ruby, S. (2008). *Restful web services*. " O'Reilly Media, Inc.".

Rodriguez, J. M., Crasso, M., Zunino, A. & Campo, M. (2010). Improving web service descriptions for effective service discovery. *Science of Computer Programming*, *75*(11), 1001–1021.

Rodriguez-Mier, P., Pedrinaci, C., Lama, M. & Mucientes, M. (2015). An integrated semantic web service discovery and composition framework. *IEEE transactions on services computing*, *9*(4), 537–550.

Roman, D., Kopeckỳ, J., Vitvar, T., Domingue, J. & Fensel, D. (2015). Wsmo-lite and hrests: Lightweight semantic annotations for web services and restful apis. *Journal of Web Semantics*, *31*, 39–58.

Rong, W. & Liu, K. (2010). A survey of context aware web service discovery: From user's perspective. In *2010 fifth ieee international symposium on service oriented system engineering* (pp. 15–22).

Ross, J. W. & Westerman, G. (2004). Preparing for utility computing: The role of it architecture and relationship management. *IBM systems journal*, *43*(1), 5–19.

Rostami, N. H., Kheirkhah, E. & Jalali, M. (2013). Web services composition methods and techniques: A review. *International Journal of Computer Science, Engineering & Information Technology*, *3*(6), 10–5121.

Rosvall, M., Grönlund, A., Minnhagen, P. & Sneppen, K. (2005). Searchability of networks. *Physical Review E*, *72*(4), 046117.

Schröder, G., Thiele, M. & Lehner, W. (2011). Setting goals and choosing metrics for recommender system evaluations. In *Ucersti2 workshop at the 5th acm conference on recommender systems, chicago, usa* (Vol. 23, p. 53).

Schulte, S. (2010). *Web service discovery based on semantic information-query formulation and adaptive matchmaking* (Unpublished doctoral dissertation). Technische Universität.

Seguin, C., Van Den Heuvel, M. P. & Zalesky, A. (2018). Navigation of brain networks. *Proceedings of the National Academy of Sciences*, *115*(24), 6297–6302.

Shao, L., Zhang, J., Wei, Y., Zhao, J., Xie, B. & Mei, H. (2007). Personalized qos prediction forweb services via collaborative filtering. In *Ieee international conference on web services (icws 2007)* (pp. 439–446).

Sheng, Q. Z., Benatallah, B., Dumas, M. & Mak, E. O.-Y. (2002). Self-serv: A platform for rapid composition of web services in a peer-to-peer environment. In *Vldb'02: Proceedings of the 28th international conference on very large databases* (pp. 1051–1054).

Sheng, Q. Z., Benatallah, B., Maamar, Z., Dumas, M. & Ngu, A. H. (2004). Enabling personalized composition and adaptive provisioning of web services. In *International conference on advanced information systems engineering* (pp. 322–337).

Sheng, Q. Z., Qiao, X., Vasilakos, A. V., Szabo, C., Bourne, S. & Xu, X. (2014). Web services composition: A decade's overview. *Information Sciences*, *280*, 218–238.

Sheth, A. P., Gomadam, K. & Lathem, J. (2007). Sa-rest: Semantically interoperable and easier-to-use services and mashups. *IEEE Internet Computing*, *11*(6), 91–94.

Sheugh, L. & Alizadeh, S. H. (2015). A note on pearson correlation coefficient as a metric of similarity in recommender system. In *2015 ai & robotics (iranopen)* (pp. 1–6).

Shi, M., Liu, J., Zhou, D., Tang, M. & Cao, B. (2017). We-lda: a word embeddings augmented lda model for web services clustering. In *2017 ieee international conference on web services (icws)* (pp. 9–16).

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A. & Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, *30*(3), 83–98.

Silva, T. C. & Zhao, L. (2016). *Machine learning in complex networks* (Vol. 2016). Springer.

Singh, M. P. & Huhns, M. N. (2005). *Service-oriented computing*. Wiley Online Library.

Snell, J., Tidwell, D. & Kulchenko, P. (2001). *Programming web services with soap: building distributed applications*. " O'Reilly Media, Inc.".

Sreenath, R. M. & Singh, M. P. (2004). Agent-based service selection. *Journal of Web Semantics*, *1*(3), 261–279.

Stavropoulos, T. G., Andreadis, S., Bassiliades, N., Vrakas, D. & Vlahavas, I. (2015). The tomaco hybrid matching framework for sawsdl semantic web services. *IEEE Transactions on Services Computing*, *9*(6), 954–967.

Stein, S., Barchewitz, K. & El Kharbili, M. (2008). Enabling business experts to discover web services for business process automation. In *Emerging web services technology, volume ii* (pp. 23–39). Springer.

Suda, B. (2003). Soap web services. *Retrieved June*, *29*, 2010.

Sukkar, M. (2010). *Design and implementation of a service discovery and recommendation architecture* (Unpublished doctoral dissertation). University of Waterloo.

Sun, J., Qu, H., Chakrabarti, D. & Faloutsos, C. (2005). Neighborhood formation and anomaly detection in bipartite graphs. In *Data mining, fifth ieee international conference on* (pp. 8–pp).

Sun, X. & Zhuge, H. (2014). Modeling and navigation of social information networks in metric spaces. *World Wide Web*, *17*(4), 649–670.

Swami Das, M., Govardhan, A. & Vijaya Lakshmi, D. (2020). Web service classification and prediction using rule-based approach with recommendations for quality improvements. In *Proceedings of the third international conference on computational intelligence and informatics* (pp. 311–323).

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.

Tapia, B., Torres, R. & Astudillo, H. (2011). Simplifying mashup component selection with a combined similarity-and social-based technique. In *Proceedings of the 5th international workshop on web apis and service mashups* (p. 8).

Tavares, A. L. & Valente, M. T. (2008). A gentle introduction to osgi. *ACM SIGSOFT Software Engineering Notes*, *33*(5), 1–5.

Telesford, Q. K., Joyce, K. E., Hayasaka, S., Burdette, J. H. & Laurienti, P. J. (2011). The ubiquity of small-world networks. *Brain connectivity*, *1*(5), 367–375.

Tenenbaum, J. B., De Silva, V. & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *science*, *290*(5500), 2319–2323.

Thiagarajan, R., Mayer, W. & Stumptner, M. (2009). Semantic service discovery by consistency-based matchmaking. In *Advances in data and web management* (pp. 492–505). Springer.

Tian, G., Sun, C., He, K.-q. & Ji, X.-m. (2016). Transferring auxiliary knowledge to enhance heterogeneous web service clustering. *International Journal of High Performance Computing and Networking*, *9*(1-2), 160–169.

Tian, G., Wang, J., He, K. et al. (2016). Leveraging auxiliary knowledge for web service clustering. *Chinese Journal of Electronics*, *25*(5), 858–865.

Tong, H., Faloutsos, C. & Pan, J.-Y. (2006). Fast random walk with restart and its applications. In *Sixth international conference on data mining (icdm'06)* (pp. 613–622).

Tran, V. X. & Tsuji, H. (2008). Qos based ranking for web services: Fuzzy approaches. In *2008 4th international conference on next generation web services practices* (pp. 77–82).

Travers, J. & Milgram, S. (1977). An experimental study of the small world problem. In *Social networks* (pp. 179–197). Elsevier.

Vaddi, S. & Mohanty, H. (2019). Webservice specification and discovery. In *Webservices* (pp. 25–51). Springer.

Vázquez, A. (2003). Growing network with local rules: Preferential attachment, clustering hierarchy, and degree correlations. *Physical Review E*, *67*(5), 056104.

Vechtomova, O. & Karamuftuoglu, M. (2007). Query expansion with terms selected using lexical cohesion analysis of documents. *Information processing & management*, *43*(4), 849–865.

Verma, K., Sivashanmugam, K., Sheth, A., Patil, A., Oundhakar, S. & Miller, J. (n.d.). Meteor–s wsdi: A scalable infrastructure of registries.

Wan, Y., Chen, L., Yu, Q., Liang, T. & Wu, J. (2016). Incorporating heterogeneous information for mashup discovery with consistent regularization. In *Pacific-asia conference on knowledge discovery and data mining* (pp. 436–448).

Wang, D., Cui, P. & Zhu, W. (2016). Structural deep network embedding. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 1225–1234).

Wang, H., Feng, Z., Chen, S., Xu, J. & Sui, Y. (2010). Constructing service network via classification and annotation. In *Service oriented system engineering (sose), 2010 fifth ieee international symposium on* (pp. 69–73).

Wang, J., Gao, P., Ma, Y., He, K. & Hung, P. C. (2017). A web service discovery approach based on common topic groups extraction. *IEEE Access*, *5*, 10193–10208.

Wang, Y., Lin, X., Wu, L. & Zhang, W. (2017). Effective multi-query expansions: Collaborative deep networks for robust landmark retrieval. *IEEE Transactions on Image Processing*, *26*(3), 1393–1404.

Wang, Y. & Stroulia, E. (2003). Flexible interface matching for web-service discovery. In *Proceedings of the fourth international conference on web information systems engineering, 2003. wise 2003.* (pp. 147–156).

Watts, D. J., Dodds, P. S. & Newman, M. E. (2002). Identity and search in social networks. *science*, *296*(5571), 1302–1305.

Watts, D. J. & Strogatz, S. H. (1998). Collective dynamics of 'small-world'networks. *nature*, *393*(6684), 440.

Wei, D., Wang, T., Wang, J. & Bernstein, A. (2011). Sawsdl-imatcher: A customizable and effective semantic web service matchmaker. *Journal of Web Semantics*, *9*(4), 402–417.

Weiss, M. & G.R, G. (2010). Modeling the mashup ecosystem: structure and growth. In *Rd management 40.1 (2010)* (p. 40-49).

Wieringa, R. (2005). Requirements researchers: are we really doing research? *Requirements Engineering*, *10*(4), 304–306.

Wu, C. (2012). Wsdl term tokenization methods for ir-style web services discovery. *Science of computer programming*, *77*(3), 355–374.

Wu, W., Li, B., Chen, L. & Zhang, C. (2017). Consistent weighted sampling made more practical. In *Proceedings of the 26th international conference on world wide web* (pp. 1035–1043).

Wu, Z., Deng, S. & Wu, J. (2014). *Service computing*. Elsevier.

Xia, B., Fan, Y., Tan, W., Huang, K., Zhang, J. & Wu, C. (2014). Category-aware api clustering and distributed recommendation for automatic mashup creation. *IEEE Transactions on Services Computing*, *8*(5), 674–687.

Xia, P., Zhang, L. & Li, F. (2015). Learning similarity with cosine similarity ensemble. *Information Sciences*, *307*, 39–52.

Xu, W., Cao, J., Hu, L., Wang, J. & Li, M. (2013). A social-aware service recommendation approach for mashup creation. In *2013 ieee 20th international conference on web services* (pp. 107–114).

Yang, Y., Ke, W., Wang, W. & Zhao, Y. (2019). Deep learning for web services classification. In *2019 ieee international conference on web services (icws)* (pp. 440–442).

Yao, L., Sheng, Q. Z., Ngu, A. H., Li, X. & Benatallah, B. (2015). Unveiling contextual similarity of things via mining human-thing interactions in the internet of things. *arXiv preprint arXiv:1512.08493*.

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*.

Yu, Q., Liu, X., Bouguettaya, A. & Medjahed, B. (2008). Deploying and managing web services: issues, solutions, and directions. *The VLDB Journal*, *17*(3), 537–572.

Zarei, B. & Gaedke, M. (2020). Disco: Web service discovery chatbot. *IADIS International Journal on WWW/Internet*, *18*(2).

Zhang, C., Zhu, D., Zhang, Y. & Yang, M. (2007). A web service discovery mechanism based on immune communication. In *2007 international conference on convergence information technology (iccit 2007)* (pp. 456–461).

Zhang, J. & Ackerman, M. S. (2005). Searching for expertise in social networks: a simulation of potential strategies. In *Proceedings of the 2005 international acm siggroup conference on supporting group work* (pp. 71–80).

Zhang, J., Tan, W., Alexander, J., Foster, I. & Madduri, R. (2011). Recommend-as-you-go: A novel approach supporting services-oriented scientific workflow reuse. In *2011 ieee international conference on services computing* (pp. 48–55).

Zhang, N., Wang, J., Ma, Y., He, K., Li, Z. & Liu, X. F. (2018). Web service discovery based on goal-oriented query expansion. *Journal of Systems and Software*, *142*, 73–91.

Zhao, H., Chen, J. & Xu, L. (2019). Semantic web service discovery based on lda clustering. In *International conference on web information systems and applications* (pp. 239–250).

Zhao, H., Xu, X., Song, Y., Lee, D. L., Chen, Z. & Gao, H. (2019). Ranking users in social networks with motif-based pagerank. *IEEE Transactions on Knowledge and Data Engineering*.

Zhong, Y., Fan, Y., Tan, W. & Zhang, J. (2016). Web service recommendation with reconstructed profile from mashup descriptions. *IEEE Transactions on Automation Science and Engineering*, *15*(2), 468–478.

Zhong, Y., Fan, Y., Tan, W. & Zhang, J. (2018). Web service recommendation with reconstructed profile from mashup descriptions. *IEEE Transactions on Automation Science and Engineering*, *15*(2), 468–478.

Zhou, T., Ren, J., Medo, M. & Zhang, Y.-C. (2007). Bipartite network projection and personal recommendation. *Physical review E*, *76*(4), 046115.

Zuev, K., Boguná, M., Bianconi, G. & Krioukov, D. (2015). Emergence of soft communities from geometric preferential attachment. *Scientific reports*, *5*, 9421.

Appendix A

Appendix