S3niffer : A text description-based service search system

Isaac Caicedo-Castro

UNIV. GRENOBLE ALPES

Universidad NACIONAL de Colombia

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES
ET DE L'UNIVERSITÉ NACIONALE DE COLOMBIE

Préparée dans le cadre d'une cotutelle entre l'Université Grenoble Alpes et de l'Université Nacionale de Colombie

Spécialité : **Informatique**

Arrêté ministérial : 2013

Présentée par

Isaac Bernardo Caicedo-Castro

Thèse dirigée par **Marie-Christine Fauvet**
et codirigée par **Helga Duarte-Amaya**

préparée au sein du **Laboratoire d'Informatique de Grenoble (LIG)**
et de **l'École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique (MSTII)**

S^{3}niffer: Un système de recherche de service basé sur leur description

S^{3}niffer: A text description-based service search system

Thèse soutenue publiquement le **12 mai 2015**,
devant le jury composé de :

Sylvie Calabretto
Professeur, INSA de Lyon, Présidente
Marlon Dumas
Professeur, Université de Tartu, Rapporteur
Mohand Boughanem
Professeur, Université Paul Sabatier, Rapporteur
Sylvie Calabretto
Professeur, INSA de Lyon, Examinateur
Marie-Christine Fauvet
Professeur, Université Joseph Fourier, Directeur de thèse
Helga Duarte-Amaya
Professeur, Université Nacionale de Colombie, Co-Directeur de thèse

UNIV. GRENOBLE ALPES

Universidad NACIONAL de Colombia

S^{3}niffer: A text description-based service search system

Isaac B. Caicedo-Castro

THESIS
Submitted for the degree of
Doctor of Philosophy

A doctoral thesis jointly supervised between the
Univ. Grenoble Alpes and the **Univ. Nacional de Colombia**

Major in **Informatics** at the Univ. Grenoble Alpes
Major in **Engineering** at the Univ. Nacional de Colombia

Thesis committee:
Professor Sylvie Calabretto, INSA de Lyon, Examiner, President
Professor Mohand Boughanem, Paul Sabatier University, Reviewer
Professor Marlon Dumas, University of Tartu, Reviewer
Professor Marie-Christine Fauvet, Univ. Joseph Fourier, Supervisor
Professor Helga Duarte-Amaya, Univ. Nacional de Colombia, Co-supervisor

S^{3}niffer (a Sniffer dog for Searching Services) is the main contribution of this research. This is a system designed to find services by comparing the similarity between their descriptions[1] and users' queries expressed in free text. The name of the system is inspired by sniffer dogs which are able to find preys. In this context, preys are services. The assumption of this research is that descriptions in OWL-S profiles provide the required information to apply IR models for discovering services.

[1] The word cloud in the figure was drawn by using a corpus composed by OWL-S profiles from the collection named OWLS-TC4 (http://projects.semwebcentral.org/projects/owls-tc/, retrieved on July of 2014)

To my beloved family, the jewels of my life, my son, my daughter, my wife, my mom, and my crazy dad: Isaac Miguel, María Emma, Ena, Rubby, and Jesus.

Abstract

In this research, we address the problem of retrieving services which fulfil users' need expressed in query in free text. Our goal is to cope the term mismatch problems which affect the effectiveness of service retrieval models applied in prior research on text descriptions-based service retrieval models. These problems are caused due to service descriptions are brief. Service providers use few terms to describe desired services, thereby, when these descriptions are different to the sentences in queries, term mismatch problems decrease the effectiveness in classical models which depend on the observable text features instead of the latent semantic features of the text.

We have applied a family of Information Retrieval (IR) models for the purpose of contributing to increase the effectiveness acquired with the models applied in prior research on service retrieval. Besides, we have conducted systematic experiments to compare our family of IR models with those used in the state-of-the-art in service discovery. From the outcomes of the experiments, we conclude that our model based on query expansion via a co-occurrence thesaurus outperforms the effectiveness of all the models studied in this research. Therefore, we have implemented this model in S^3niffer, which is a text description-based service search engine.

Keywords: *Information retrieval, Matrix factorisation, IR-based service discovery, Query expansion, Co-occurrence thesaurus.*

Résumé

Dans cette recherche, nous abordons le problème de le recherche de services qui répondent à des besoins des utilisateurs exprimés sous forme de requête en texte libre. Notre objectif est de résoudre les problèmes qui affectent l'efficacité des modèles de recherche d'information existant lorsqu'ils sont appliqués à la recherche de services dans un corpus rassemblant des descriptions standard de ces services.

Ces problèmes sont issus du fait que les descriptions des services sont brèves. En effet, les fournisseurs de services n'utilisent que quelques termes pour décrire les services souhaités. Ainsi, lorsque ces descriptions sont différentes des phrases dans les requêtes ce qui diminue l'efficacité des modèles classiques qui dépendent de traits observables au lieu de traits sémantiques latents du texte.

Nous avons adapté une famille de modèles de recherche d'information (IR) dans le but de contribuer à accrotre l'efficacité acquise avec les modèles existant concernant la découverte de services. En outre, nous avons mené des expériences systématiques afin de comparer notre famille de modèles IR avec ceux de l'état de l'art portant sur la découverte de service. Des résultats des expériences, nous concluons que notre modèle basé sur l'extension des requêtes via un thésaurus co-occurrence est plus efficace en terme des mesures classiques utilisées en IR que tous les modèles étudiés dans cette recherche. Par conséquent, nous avons mis en place ce modèle dans S^3niffer qui est un moteur de recherche de service basé sur leur description standard.

Resumen

En esta investigación, nosotros abordamos el problema de extraer servicios que satisfagan las necesidades de los usuarios, las cuales son expresadas en consultas en texto libre. Nuestro objetivo es resolver los problemas relacionados con la correspondencia de términos, los cuales afectan la efectividad de los modelos de extracción de servicios aplicados en previas investigaciones en extracción de servicios basada en descripciones de texto. Tales problemas son causados porque la descripción de servicios son breves. Los proveedores de servicio usan pocos términos para describir los servicios deseados, por ende, cuando las descripciones son diferentes a las frases en las consultas, los problemas de correspondencia de términos reducen la efectividad en modelos clásicos que dependen de las características observables del texto en vez de las características semánticas latentes.

Nosotros hemos aplicado una familia de modelos de extracción de información con el propósito de contribuir a incrementar la efectividad adquirida con los modelos aplicados en previas investigaciones en extracción de servicios. Además, nosotros realizamos experimentos para comparar nuestra familia de modelos de extracción de información con aquellos usados en el estado del arte en descubrimiento de servicios. De los resultados experimentales, se concluye que nuestro modelo basado en expansión de consulta via un tesauros de co-ocurrencias, supera la efectividad de todos los modelos estudiados en esta investigación. Por lo tanto, nosotros hemos implementado este modelo en S^3niffer, el cual es un motor de búsqueda de servicios basado en descripciones textuales.

Acknowledgements

When I finished reading the Crichton's book titled *The andromeda strain*, I realised that being only a software engineer would never be enough to find my happiness. Since then my dream was born, so, I wanted to become a researcher. Moreover, I realised that becoming a researcher would be one of the most challenging goals of my life. Indeed, doing my PhD research has been the most challenging activity that I ever done. It was exactly as hard as described by Michael Crichton in his book:

> ...scientific research was much like prospecting: you went out and you hunted, armed with your maps and your instruments, but in the end your preparations did not matter, or even your intuition. You needed your luck, and whatever benefits accrued to the diligent, through sheer, grinding hard work.[2]

However, many awesome persons have helped me to do my PhD research. First at all, I want to thank my precious son Isaac Miguel, my adorable daughter Maria Emma, and my beloved wife Ena. They're my most important people and the best motivation to do challenging things like this thesis. Everything I am and all that is mine belongs to them for ever.

I would like to thank my mother Rubby who encouraged me to endure a PhD, and has supported me during this adventure in many ways.

A very special thanks to my dear supervisor in France, Marie-Christine Fauvet. It has been a pleasure and honor to work under her guidance. I love her pragmatic

[2] Michael Crichton, *The andromeda strain*, Chapter 20: Routine, pg. 245, Harpercollings publishing, 1969.

sense of life and research, and her brilliant intellectual energy. I learned uncountable worthy things from her. Besides, she has managed to find typos and advice me to make this thesis a better work than it would have been without her help and support. Indeed, Marie-Christine has read very carefully every chapter and made sure the words make sense.

I thank to my supervisor in Colombia, Helga Duarte-Amaya, for giving me this opportunity, for all her advise, support, and for contacting Marie-Christine in order to agree a jointly supervision between the Université Grenoble Alpes and the Universidad Nacional de Colombia.

I also thank to my father in law Francisco Pernett, and my mother in law Olfa Agamez. They have helped me in many ways to complete this mission.

Thanks to Prof. Fabio Gonzales I've learned about machine learning, recommender systems, and information retrieval. I really enjoyed his wonderful classes and our fruitful talks about both topics and research in general. Thanks to him I found my Holy Grails, namely, machine learning and information retrieval.

I wish to thank to the talented members of the MRIR team who help me during my PhD research: my deepest appreciation goes to Dr. Philippe Mulhem, who always took a lot of time to discuss with me, gave me clues and advices about many aspects of a research in information retrieval. Thanks, Philippe, for keeping me on the right way. I thank to Dr. Jean-Pierre Chevallet, Prof. Ahmed Lbath, and Mohannad ALMasri for several discussions and advice which they gave me. When I arrived in France, Prof. Catherine Berrut became my angel. She helped me to find a comfortable flat, besides, she has always been caring of my family and me.

Many thanks to my sweet friends Vero Duprez, Ezequiel Ferrero, and the Moreira family: Beth, Maria Rita, and Alberto. They have been like my family in France.

Finally, I thank COLCIENCIAS-COLFUTURO (grant 528) and the Universidad de Córdoba (in Colombia) for their financial support.

Contents

List of Figures

List of Tables

Chapter 1
Introduction and research context

Contents

Abstract In this Chapter we state and motivate the problem addressed in this research, in general terms, retrieving services which satisfy users' requirements expressed in free text queries. In special, we delve into the problems which affect the effectiveness of the service retrieval process. Therefore, we aim at applying an Information Retrieval (IR) model which outperforms the effectiveness of models applied in prior research on service retrieval. The contribution of our research is threefold: 1) The application of a family of IR models for retrieving services. To the best of our knowledge, none of them have been used in prior research on service retrieval. 2) An empirical study conducted to identify the IR model with the highest effectiveness in the context of service retrieval. 3) The implementation of S^{3}niffer, which is a text description-based service retrieval system. Furthermore, we define the scope of our research.

1.1 Introduction

In this research we aim to cope the problem of finding Web services that fulfil users' requirements expressed in free text queries, i.e., queries composed by one or several terms instead of a specific language with operators. For users, building up query expressions with free text (e.g., I want to book a room for 3 nights) is easier than learning and using a specific language with operators (e.g., SPARQL, RDF query language, Structured Query Language, etc.). S^{3}niffer is the resulting tool of this research.

S^{3}niffer is a service search system based on an Information Retrieval (IR) model proposed in this research. S^{3}niffer is part of the system sketched in Figure 1.1. This system is designed to provide mobile users with services (Na-Lumpoon et al., 2012, 2013). For instance, services to book a room, or reserve the table in a restaurant located in a certain city, etc. The role of each module, and flow of information are detailed as follows:

1. User interaction and query management module aims at managing user connections and getting queries submitted by users and sent using their mobile device. Users' queries, identifications, and context information are received by this module in the data flow (1). This module extracts from the query all necessary information for the discovery, composition, and execution of services.
2. The module User management System is in charge of managing users' context and profile with respect of their privacy. This module receives the users' identifications in the data flow (2). Thereafter, this module sends forward by means of the data flow (3) users' queries, context and class of profile.
3. Discovery system: this is S^{3}niffer. Given a user's query (received in the data flow (3)), what are the services which may potentially meet the user's needs expressed in such query? The services which fulfil the user's needs are sent, in the data flow (4), to the next module.
4. Eventually the module Composition and orchestration system composes operations exposed by services returned by the discovery module, and then execute the resulting composite service. During its execution the service might require information from the interaction between the user and the system (see flow (5)).

Our research focuses only on the issues raised by the question in the third above mentioned item. Therefore, S^{3}niffer is the discovery system in Figure 1.1.

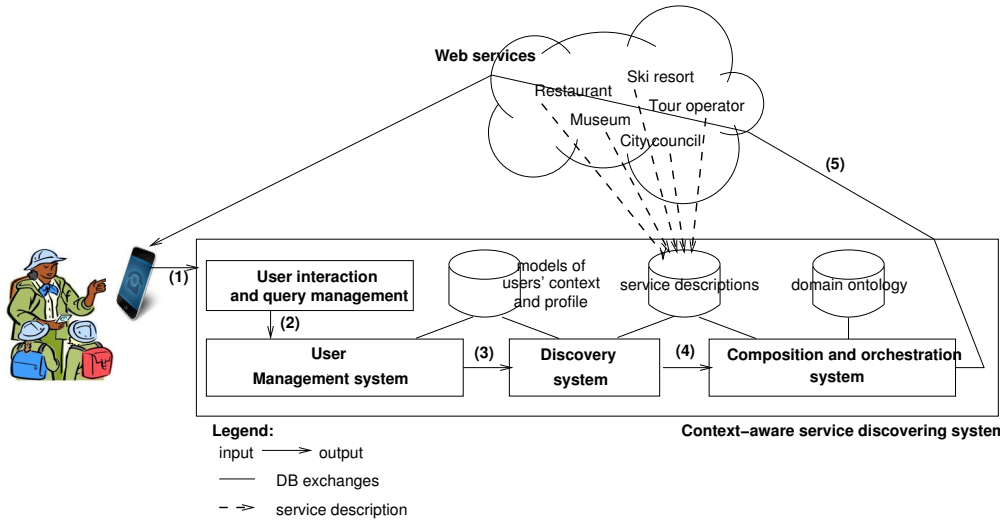

Fig. 1.1 Architecture of a system for discovering, execution and composition of services for mobile users

The remainder of this chapter is outlined as follows: In Section 1.2, we illustrate an example about a tourist, who uses the system depicted in Figure 1.1. With this example we aim to motivate the need of a system like S^{3}niffer to seek services in the context of the tourism domain. In Section 1.3, we present the mathematical notation followed in the rest of the thesis. In Section 1.4, we state the problem addressed in our research. Besides, we presents the aim of this thesis. In Section 1.5, we delve into the reasons which motivated this research. In Section 1.6, we explain the dataset, experiments, and the observed variables. In Section 1.7, we present contributions, assumptions, and limitations of the thesis. Finally, in Section1.8, we outline the next chapters of this work.

1.2 An illustrative example

Alice is an American tourist visiting Paris in France. She has forgotten to book a room. Thus, she picks up her smartphone and accesses the system above mentioned

(see Figure 1.1), and submits the query: I want to book a room for 3 nights from tonight. Moreover, the system captures Alice's context information, namely: coordinates = "48.2167° N, 2.3332° E" and date = "1/06/2014". Besides, the system has the following information about Alice's profile: name = "Alice", citizenship="USA", travelPurpose="tourism", gender="female".

Thereafter, S^{3}niffer searches services for booking a room. It gives to the Composition and Orchestration System (COS) a ranked list of candidate services for booking rooms in hotels. The service which has the highest rank in this list contains the following operation: BookARoom, this operation receives as parameters the name of the hotel, the number of nights and persons who shall stay in the room, the date and time to check-in, and the user's name and telephone number. As a result, the operation returns a confirmation whether the room has been booked or not.

COS executes this operation, and Alice successfully book a room in *Novotel hotel*. The Alice's context information is used by the COS to execute services.

At 4 PM, she wants to book a table at the finest restaurant in the city, and the direction to get there. Once again, she uses the same system and submits the query: I want to book a table for 2 people at the finest restaurant in the city, and the direction to the restaurant. At this time, Alice's profile has not been changed, and the system captures the following Alice's context information: coordinates = "48.8567° N, 2.3508° E" and date = "1/06/2014".

This query has two requirements, then the User Interaction and Query Management (UIQM) module splits this query in two subqueries. Therefore, the first query submitted to S^{3}niffer is I want to book a table at the finest restaurant in the city. The second query submitted to S^{3}niffer is the direction to the restaurant. S^{3}niffer shall send to the COS two ranked lists of candidate services, which correspond with each subquery. From the list of candidate services that may fulfil the first subquery, the one which has the highest rank contains the following operations:

- FindFinestRestaurant: This operation receives as a parameter the name of the city where the user is looking for the finest restaurant. The operation returns the name and the address of the restaurant.

- BookRestaurant: This operation receives as parameters the restaurant name, the number of persons, and the user's name and telephone number. As a result, the operation returns a confirmation whether the table has been booked or not.

In the another ranked list of candidate services that fulfils the second subquery, the one which has the highest rank contains the following operations:

- FromCoordinatesToCity: Given the geographical coordinates, this operation returns the name of the city where is allocated the coordinates of certain point of interest.
- CoordinatesFromAddress: Given an address, this operation returns its geographical coordinates.
- GetDirection: This operation provides instructions on how to reach a destination. This operation receives two parameters, the coordinates of the starting point, and the coordinates of the destination.

COS takes the operations of both services and compose them. The execution of the resulting composite service fulfils both Alice's needs (i.e., booking a table in the finest restaurant of the city, and knowing the direction to go there).

On that night, while she is enjoying a delightful dinner in *Le Meurice* restaurant, Alice is wondering about the weather in the next day. She needs this information to decide whether she will go to *Louvre museum* or *Euro Disney*. One more time, she uses the system and submits the following query: I want to buy a ticket for Euro Disney tomorrow if the weather forecast is sunny, otherwise, buy a ticket for Louvre museum. At this time, Alice's profile is still the same, however, her new context information is as follows: coordinates = "48.8651° N, 2.3280° E" and date = "1/06/2014".

Similar to the previous query, this one contains three requirements, therefore the UIQM module splits the query in three subqueries. The first subquery is to buy a ticket for Euro Disney tomorrow. The second subquery is the weather forecast is sunny. The last subquery is buy a ticket for Louvre museum. All three subqueries are sent to S^{3}niffer, thereby it sends three lists of services to the COS. From the list of candidate services that may fulfil the first subquery, the one which has the highest rank contains the following operation: BuyTickets4EuroDisney, this operation receives as

parameters the name of the customer, the number of required tickets, information of a credit card, etc. As a result, the operation returns a confirmation whether the transaction has been successfully finished or not.

In the another ranked list of candidate services that may fulfil the second subquery, the one which has the highest rank contains the following operation: GetWeatherForecast, this operation returns the weather for a given city of a certain country, and for a given date.

In the ranked list of candidate services that may fulfil the last subquery, the one which has the highest rank contains the following operation: BuyTickets4LouvreMuseum, this operation receives similar operation as the one to by tickets for *Euro Disney*, besides, the result of this operation is the same.

In the same fashion as before, the COS composes all operations of previous services. The execution of the resulting composite service fulfils Alice's requirements regarding her condition.

With the above mentioned system, users are able to consume services accessible on the Internet, from their mobile devices. Besides, service providers do not need to produce front-end applications, which serve as interfaces to access their services. This thesis addresses the problem of searching services for fulfilling specific users' requirements.

1.3 Mathematical notation

In this section we introduce the mathematical notation which shall be used in this thesis. Vectors and matrices are denoted by bold letters. However, uppercase letters denote matrices (e.g., $\mathbf{Y}$), while lowercase letters represent vectors (e.g., $\mathbf{x}$). A superscript T is used to denote the transpose of either matrices or vectors (e.g., $\mathbf{x}^T$ or $\mathbf{Y}^T$).

Vectors are assumed to be column vectors, hence, a vector with m elements is written as $\mathbf{w} = (w_1, \ldots, w_m)^T$, whereas the i^{th} element of a vector $\mathbf{w}$ is denoted as either w_i or $(\mathbf{w})_i$. Besides, $\mathbf{x}^T$ is a row vector. We will use the notation $a \in \mathbb{R}$ to indicate that a is a scalar, likewise, $\mathbf{x} \in \mathbb{R}^m$ is a m-dimensional vector, and $\mathbf{Y} \in \mathbb{R}^{m \times n}$ is a matrix, which has n column vectors, all of them belong to a

real-valued m-dimensional vector space. Moreover, an element of the matrix $\mathbf{Y}$, located in the i^{th} row and the j^{th} column, is denoted either Y_{ij} or $(\mathbf{Y})_{ij}$.

Finally, $\{b_i\}_{i=1}^{n}$ represents a set of n elements, where i indexes different elements that belong to it.

1.4 Problem statement and research aim

Given a query in free text, the problem of discovering services has been addressed with Information Retrieval (IR) models. In this context, collections of WSDL[1] documents have been preprocessed, indexed, and used to retrieve services which may fulfil a specific requirement. Such document contains a syntactically-based description. It includes service name, operations name and signature, and sometimes descriptions in natural language are also given, which are commentaries written by programmers.

Despite that WSDL is the standard service description, we have adopted OWL-S (Burstein et al., 2004) which is an OWL[2]-based format for describing, discovering, composing, enacting, and monitoring services. With OWL-S it is possible to describe WSDL-based services as well as those based on REST-ful[3] architecture.

With OWL-S, the functional description of a service is given in its profile in a way that it is suitable for a software agent searching for services (Burstein et al., 2004). Figure 1.2 depicts a chunk of code of a service profile described in a OWL-S document. Tags <profile:serviceName> and <profile:textDescription> introduce the name of the service and its description in free text, respectively. The problem is to measure the extent to which this information matches a query, which is the mean utilised by users in an attempt to communicate their needs to a system designed for discovering services. This problem is known in Computer Science as an IR problem.

[1] *Web Service Description Language*, see http://www.w3.org/TR/wsdl20/, retrieved on February of 2015

[2] *Web Ontology Language*, a.k.a. DAML-S, see http://www.w3.org/TR/owl-features/, retrieved on February of 2015

[3] Representational State Transfer, REST, see http://www.w3.org/2012/ldp/charter, retrieved on February of 2015

Thus, several service descriptions are collected from their respective OWL-S profiles. Herein the collection of service descriptions is denoted as $C = \{s_d\}_{d=1}^{n}$, where s_d is a text description of a service, and n is the number of services. Given a query Q, S^{3}niffer aims to rank each description s_d with a score, which the greatest score is for the most relevant services to the query. Relevant services are what the user would like to find, and their respective descriptions may be regarded as containing the expected answer to a given user's query. Nevertheless, a service is relevant whether, actually, it fulfils a specific user's need rather than its description match most of terms used in the query. The problem is to find a model based on a retrieval function f to score each service description with respect to the query in order to rank all the descriptions according their scores, i.e., $\forall s_d$ to compute $f(Q, s_d) \in \mathbb{R}$ such that $d = 1, \ldots, n$.

```
<profile:Profile>
...
<profile:serviceName>
  WorldwideHotelInfoService
</profile:serviceName>
<profile:textDescription>
  This service returns information of all famous
  hotels in the world.
</profile:textDescription>
...
</profile:Profile>
```

Fig. 1.2 Profile of a Web service in a OWL-S document

Service descriptions are sentences or paragraphs, which are briefer than usual documents (e.g., books, Web pages, etc.) (see Figure 1.2). Indeed, after removing all stop words in the collection called OWLS-TC4[4], it has:

- minimum 3 terms per description,
- 8 terms per description in the first quartile,
- a median of 9 terms per description,

[4] The OWL-S profiles from the collection named OWLS-TC4 (http://projects.semwebcentral.org/projects/owls-tc/, retrieved on July of 2014)

- 11 terms per description in the third quartile
- a mean of 9.99 terms per description,
- with a standard deviation of 4.07 terms, and
- maximum 50 terms per descriptions.

So, in the domain of service retrieval there are term mismatch problems (e.g., synonymy and homonymy problems) because of service descriptions are brief. Service providers use sentences (or sometimes short paragraphs) to describe desired services, hence, when these descriptions are different to the sentences in queries, this causes term mismatch problems in classical models which depend on the observable text features rather than the hidden semantic features (Ponte and Croft, 1998; Zhai and Lafferty, 2004). For instance, suppose a user submits the query: I want to book an apartment, and assuming the description of the desired service is: This service allows users to reserve a flat. In this example, terms such as book and reserve are synonyms as terms apartment and flat either. In this example, IR models based on the matching among terms (e.g., Vector Space Model or Boolean Retrieval Model) are not able to find services whose descriptions match that sort of query. As a consequence, in this context term mismatch problems affect the effectiveness of a text-based retrieval system applied on service discovery. A service retrieval system with high effectiveness increases the satisfaction of users, therefore, this encourages them to use the service retrieval system. While a system with low effectiveness disappoints users, hence, they might stop to use it permanently.

In this research, we aim to overcome the above mentioned term mismatch problems, in order to outperform the effectiveness of all IR models applied in prior research on text description-based service retrieval. By achieving this goal, our purpose is to implement this model in S^3niffer in order to encourage users to search services through S^3niffer.

1.5 Motivation

Software development is a complex challenging engineering activity. To cope this underlaying complexity, paradigms of software engineering adopt a reductionism approach (DeRemer and Kron, 1975; Heineman and Councill, 2001). With this

one, software production consists of assembling components. Each component accomplishes a specific functionality. Components hide their implementation details, and communicate with each others by exchanging messages. For instance, in case of the object-oriented paradigm, engineers design components called objects. Each one has attributes encapsulated by an interface composed by methods. Attribute values describe the state of an object, while an object performs an action when one of its methods is invoked. Through the component-based paradigm, engineers only need to know component interfaces. In this way, components are easy to reuse. Besides, software structured in components is easy to modify, fix, or extend. Thus, the component-based software is more flexible than monolithic-structured software.

Indeed, Service-Oriented Architecture (SOA) comes from the component-based paradigm. Thus, SOA addresses the challenges of developing software in heterogeneous distributed platforms. With SOA, software components may interoperate with each others through computer networks. So, an engineer produces software by assembling different components that are loosely coupled. This kind of components are known as services (Papazoglou and Heuvel, 2007).

Several technologies have been created for developing SOA products, e.g., Microsoft Distributed Component Object Model (DCOM) (Brown and Kindel, 1998), Sun Remote Method Invocation (RMI) (Microsystems, 1999), and Common Object Request Broker Architecture (CORBA) (Boldt, 1995). These technologies depend on specific platforms and protocols. For instance, DCOM works only on Microsoft Windows platforms, likewise RMI on Java Virtual Machines, and CORBA depends on its own stack of technologies and protocols. Nevertheless, with the inception of the Web, companies had become potential service providers. Since then, servers have worked on different platforms and protocols over the Internet. Thus, this heterogeneous environment caused challenges which are solved by Web service technology.

A Web service is a software component which can be used over the standard protocol of the Web, i.e., HyperText Transfer Protocol (HTTP) (Vaughan-Nichols, 2002). This technology enables the implementation of distributed software systems at big scale. At this scale, business processes involve usually various companies.

The execution of each business process implies invoking several services provided by these companies, in a loose couple way.

Web services have become the chosen implementation technology for realising the SOA aim of service sharing and interoperability. Currently, the Web service technology is the standard *de facto* to deliver services through the Web. Indeed, most of companies (including famous ones, such as Amazon, Facebook, etc.) provide their own services, which have become in important building blocks for business integration. This has caused the proliferation of services on the Web. For instance, in 2014, programmableWeb[5] provides 12,213 services in several categories and formats (e.g., SOAP and REST-ful).

So, nowadays there is a huge amount of services available on the Web, however, services are useless if these cannot be found and consumed. Besides, reviewing each service description to find the required is an infeasible task for users such as Alice. Thus, for users, searching services through the Web is an issue as challenging as seeking a needle in a haystack. Users would prefer a computer[6] to be able interpret their needs and estimate the relevance of services with respect to their expressed needs. This has motivated the prior research on IR-based models for searching services. Systems based on IR-based models match service descriptions with queries (see Chapter 2). Nevertheless, in all studies conducted on prior research, most of these models are evaluated with different test suites instead of a standard test data set with the same experimental setting like in other IR domains. Thereby, from a literature survey is not possible neither identify nor conclude which is the model with highest performance in service retrieval domain. Besides, currently, there is not a work which has completely compared at least the more remarkable prior research. This is due to lack of a standard test collection for IR-based service discovery. Therefore, it is unknown which of these models has the highest effectiveness. Hence, increasing the effectiveness of IR-based service discovery systems is still a research problem.

Effectiveness refers to the ability of a IR system to successfully estimate the degree of relevance of a service regarding user's needs. A service with the lowest

[5] ProgramableWeb, http://www.programmableweb.com/, retrieved on October of 2014
[6] Currently, due to the pervasive computing phenomenon, mobile devices are a kind of computer

relevance degree is irrelevant, whereas the one with the highest relevance degree is obviously the most relevant. Back to the above mentioned example, let us imagine that Alice does not find a service to book a room when she is using S^{3}niffer. As a consequence, she might feel disappointed. If this unfortunately scenario is frequently repeated, eventually, she might stop using S^{3}niffer due to its low effectiveness, and she will try another alternative to seek services. With this example, we have intuitively illustrated the more effective a service retrieval system is, more satisfied users are. Therefore, avoiding this kind of disappointing situations has motivated us to carry out this research.

1.6 Research approach

We apply IR models in order to overcome the term mismatch issue caused by synonymy problems, which are worse in the context of our research, due to service descriptions are briefer than usual documents (see a broader discussion in Section 1.4). The research approach of our research is experimental due to IR field is an empirical discipline of Computer Science (Manning et al., 2008, pg. 151). With experiments, the goal is to identify, through facts (evidences), which one of the models proposed and studied in this research have the superior performance. In our context, the performance of an IR model depends on its effectiveness to predict how much relevant is a service for a user.

To carry out an experiment to measure the effectiveness of an IR model a data set which contains the following things is necessary (Manning et al., 2008, pg. 152):

1. A collection of documents. In this context, each document is a OWL-S profile of a service. In this research, we have used the OWL-S profiles of the fourth version of the OWL-S service retrieval test collection named OWLS-TC4[7]. It contains the descriptions of 1083 Web services from 9 domains (i.e., education, medical care, food, travel, communication, economy, weapon, geography, and simulation). Each description is written in OWL-S 1.1. This collection is the unique one which exists in service retrieval domain which has judgement relev-

[7] OWL-S Service Retrieval Test Collection, http://projects.semwebcentral.org/projects/owls-tc/, retrieved on July of 2014

ance. Besides, previous versions of this collection have been used for carrying out experiments in related recent research (Cassar et al., 2011, 2013).

2. A set of users' needs, expressible as queries. OWLS-TC4 collection includes 42 queries.
3. A set of relevance judgments. Each query in the OWLS-TC4 collection is associated with their relevance judgment provided by several users. A pooling strategy (as used in TREC[8]) was conducted to collect the relevance judgment set. This set has been collected from the top-100 results of participants of the S3 contest[9] in 2008. The judgment relevance has been graded in four different levels, i.e., highly relevant (value 3), relevant (value 2), potentially relevant (value 1), and non-relevant (value 0).

The motivation to achieve the research aim (i.e., to propose a model with superior performance to discover services) is to increase the users' satisfaction (happiness) when they seek for services. In Section 1.5 we discussed the relationship between users' satisfaction and IR model effectiveness, namely, if the model applied in S^{3}niffer has a low effectiveness, therefore users might stop using S^{3}niffer. In the context of service retrieval, effectiveness is the measure of the ability of a IR-based service discovery system to satisfy users in terms of the relevance of service retrieved. This lead us to assume that the effectiveness of the IR model is the most important factor which affects the satisfaction of S^{3}niffer users. Hence, we aim at applying an IR model in S^{3}niffer which outperforms the effectiveness of models used in prior research on service retrieval. So, our approach consists of proposing a family of IR models, then conducting experiments to measure the effectiveness of each model in order to choose the one with the highest effectiveness, and compare it with those models used in the-state-of-the-art on service retrieval.

There are several effectiveness measures of an IR model. The most basic measures are *precision*, *recall*, and *F-measure* (Manning et al., 2008, pg. 142-143). The notions of precision and recall are used the first time in (Kent et al., 1955), and the F-measure is introduced in (Rijsbergen, 1979). On one side, *precision* is the fraction of retrieved services which are relevant according the judgment of the user.

[8] Text Retrieval Conference, http://trec.nist.gov/, retrieved on February of 2015
[9] Semantic Service Selection, http://www-ags.dfki.uni-sb.de/ klusch/s3/, retrieved on July of 2014

Retrieved services are those in a ranked list of search results for a given query. On the other side, *recall* is the fraction of relevant services which are retrieved. These measures are formally defined as follows:

Let $A = \{a_i\}_{i=1}^{k}$ be the set of retrieved services where a_i is the description which corresponds with the i^{th} ranked document. Let $B = \{b_i\}_{i=1}^{m_r}$ be the set of all relevant services for a given query. The mathematical formulas to compute precision (denoted as p) and recall (denoted as r) are as follows:

$$p = \frac{|A \cap B|}{|A|} \tag{1.1}$$

$$r = \frac{|A \cap B|}{|B|} \tag{1.2}$$

Moreover, F-measure is a trade-off between precision *versus* recall. This is the weighted harmonic mean of precision and recall:

$$F = \frac{1}{\frac{\alpha}{p} + \frac{1-\alpha}{r}} = \frac{pr}{(1-\alpha)p + \alpha r}$$

if $\alpha = \frac{1}{1+\beta^2}$ then

$$F_\beta = \frac{(\beta^2+1)pr}{\beta^2 p + r}$$

where $\alpha, \beta \in \mathbb{R}$ such that $0 \leq \alpha \leq 1$, thereby, $0 \leq \beta \leq \infty$. In the default *balanced F-measure* the weights for precision and recall are equal, hence, $\alpha = 0.5$ or $\beta = 1$. It is known as F_1, which is an abbreviation of $F_{\beta=1}$. Thus, the formula for F_1 is as follows:

$$F_1 = \frac{2pr}{p+r} \tag{1.3}$$

All previous measures of effectiveness are defined with respect to the set of retrieved services. As a consequence, a cutoff point is necessary to compute these measures, for instance, the precision at top ten candidate services. In a top ten services, all of them are sorted according their computed rank in descending order.

Nonetheless, precision does not perceive this order, thus, there is no difference whether only the last three services are judged as relevant, or if the first three are judged in this way.

The precision is not a good measure for the overall ranking effectiveness. Therefore, in the IR discipline another measure is used to compare two ranked lists more accurately. This measure is called *average precision* (Manning et al., 2008, pg. 146-147). It perceives any small change in the ranking of relevant services. The formulae to compute the average precision at k ($AP@k$) is as follows:

$$AP@k = \frac{1}{k} \sum_{i=1}^{k} \frac{i}{r_i} \tag{1.4}$$

where r_i is the rank of the i^{th} relevant service in the search result. This value is infinite if the i^{th} relevant service is not retrieved. For instance, suppose the relevance status of the top ten services is (○, ○, ●, ●, ●, ●, ○, ●, ○, ○), herein ● and ○ denote whether service at that rank is relevant or not. Assuming there are 20 relevant services for this example, on one hand, the precision and recall would be 5/10= 0.5 and 5/20=0.25, respectively. Both measures will not change if the 5 relevant services are retrieved in the top ten services in a different order, e.g., if the relevant services are in the end of the top ten list, therefore the precision and recall are sill 0.5 and 0.25, respectively. On the other hand, the $AP@10$ would be (1/3 + 2/4 + 3/5 + 4/6 + 5/8)/10 = 0.27, thus, if the same relevant services are retrieved in a different order, then the $AP@10$ will be different, e.g., if the relevant services are retrieved in the end of the top ten (○, ○, ○, ○, ○, ●, ●, ●, ●, ●), thereby the $AP@10$ is (1/5 + 2/6 + 3/7 + 4/8 + 5/9)/10 = 0.20.

In IR, experiments are carried out with a set of queries, hence, the method to evaluate an IR model consists of computing the mean of the $AP@k$ obtained with each query. This measure is called *Mean Average Precision* (MAP). Even though this measures the precision of an IR model, MAP considers the recall of a model. In other words, MAP is sensible to the recall measure.

All previous measures have the same shortcoming. These are useful only when the relevance of a service is binary (i.e., relevant or non-relevant), however, relevance might be judged at several levels (e.g., very relevant, relevant, or non-

relevant). Due to this, there is another approach defined measure of the overall ranking effectiveness of an IR model in case of various relevance levels, this measure is called *Normalised Discounted Cumulative Gain* (NDCG) (Järvelin and Kekäläinen, 2002). According to this measure, each service has an associated gain corresponding to its relevance level, i.e., a highly relevant service has a higher gain than a less relevant one. $NDCG@k$ is a better approach to measure the overall ranking effectiveness of the top k services with the sum of their gains. This sum is named as cumulative gain. $NDCG@k$ has two advantage over $MAP@k$ as follows:

1. It gives a higher weight to a highly ranked service in order to emphasise the importance of ranking services with high relevance values on the top k ranked list.
2. It normalises the overall discounted cumulative gain with its normalised factor. This is computed with the perfect ranking of the services for a given query. This makes $NDCG@k$ more comparable over different queries, whereas $MAP@k$ is an average value dominated by those $AP@k$ values from an easy query with many highly relevant documents.

Moreover, experiments have been conducted to study the stability and sensitivity of evaluation measures, from the outcomes is concluded that $NDCG@k$ is best for evaluating IR models (Sakai, 2007).

Because of the advantages of $NDCG@k$ over the other measures, we adopted $NDCG@k$ to measure the effectiveness of the IR models proposed for discovering services. Furthermore, the $NDCG@k$ is defined as follows: Let $\{Q_j\}_{j=1}^{l}$ be a set of l queries such that

$$NDCG@k = \frac{1}{l}\sum_{j=1}^{l} Z_{kj}\left(\sum_{i=1}^{k}\frac{2^{f(Q_j,s_i)}-1}{log_2(1+i)}\right) \tag{1.5}$$

where f is the retrieval function (see Section 1.4), Z_{kj} is the normalised factor such way that the best ranking's $NDCG@k$ for a given query Q_j is 1.

1.7 Contribution, assumptions, and limitations

The contribution of this research is threefold:

1. The application of a family of IR models for indexing and retrieving services. To the best of our knowledge, none of them have been used in prior research on service discovery. Two of them are based on Latent Semantic Indexing (LSI) via different matrix factorisation models. The other two models are based on query expansion.
2. An empirical study conducted to chose the IR model to be implemented in S^{3}niffer. The outcomes of this study are as follows:

   a. The results of our experiments suggest that extending queries with terms extracted from a co-occurrence thesaurus is the model which outperforms all the others, also studied in this work.
   b. We show as well, that the model based on Latent Dirichlet Allocation (LDA) has a lesser effectiveness than all the LSI models considered in this research.
   c. Our results reveal that there is no difference which is statistically significative in the effectiveness among the LSI-based models assessed in this research.
   d. Furthermore, LSI-based models have a similar effectiveness than expansion of queries with terms extracted from WordNet.

3. The implementation of S^{3}niffer, which is a text description-based service search system based on the IR model which outperforms the others in the above mentioned study.

In this research we assume:

1. Users express their needs through free text queries (e.g., I want to book a room for 3 nights) instead of a specific language with operators (e.g., JQuery).
2. Service descriptions are contained into OWL-S profiles. Hence, the models proposed in this thesis can be applied to services based on WSDL and REST-ful architecture, because both kind of services can be formatted in OWL-S.
3. Service descriptions are documents shorter than those used in other contexts, e.g., books, Web pages, etc. (see Figure 1.2).
4. Service descriptions and queries are expressed in English.

5. The collection OWLS-TC4[10] is representative for a text description-based service retrieval system in 9 domains including E-Tourism. Besides, previous versions of this collection have been used for carrying out experiments in related recent research (Cassar et al., 2011, 2013).

Finally, the limitations of the research are presented as follows:

1. We do not deal neither with composition nor execution of services.
2. We do not handle recommendations of services according either users' preferences or Quality-of-Services. Recommendation and retrieval of services are two different problems and research trends as we shall discuss in Section 2.2. In recommendation, the relevance estimation of services is based on users' preferences rather than queries. We focus in this research on service retrieval, hence, service recommendation is out of the scope of this research.
3. We do not deal with service crawling because we use the OWL-S service retrieval test collection named OWLS-TC4.
4. We do not infer terms from abbreviations found in OWL-S files. This is another topic that deserves to be investigated and it is out of the scope of the goal of this research.
5. We do not address scalability issues because we only concern the effectiveness of the service retrieval process. We assume there are enough computational resources to apply IR models for retrieving services.

1.8 Thesis outline

The thesis is outlined as follows:

- In Chapter 2, we discuss a complete survey of prior research on IR-based service discovery. Besides, we formulate the open research questions, which we address in this research in order to accomplish our aim (see Section 2.7).
- In Chapter 3, we delve into the proposed models to achieve the aim proposed in Section 1.4. First at all, we describe how S^{3}niffer preprocesses service de-

[10] OWL-S Service Retrieval Test Collection, http://projects.semwebcentral.org/projects/owls-tc/, retrieved on July of 2014

scriptions. After that, we explain two Latent Semantic Indexing (LSI) models via Non-negative Matrix Factorisation (NMF) and Minimising the Squared Error. Furthermore, we describe other two models based on query expansion. The first one based on natural language processing and extraction of synonyms from WordNet. The second one based on the expansion of queries with terms taken from a co-occurrence thesaurus.

- In Chapter 4, we describe the experimental setting of the evaluation conducted over the models described in previous chapter, besides the most relevant of prior research. Moreover, we present and analyse the results of the evaluation. Furthermore, we describe the implementation details of the proof-of-concept of S^{3}niffer.
- In Chapter 5, we summarise the thesis with emphasis upon the findings and the contributions made by the outcomes obtained in the research. Finally, we outline suggestions for further research.

Chapter 2
A literature review on text description-based service retrieval

Contents

Abstract In this Chapter we present and discuss the prior research on text description-based service retrieval. In some works have been applied Vector Space Model (VSM) (Platzer and Dustdar, 2005; Lee et al., 2007; Crasso et al., 2008; Wu, 2012; Crasso et al., 2014), however, this model is unable to capture relations between the terms in the queries and service descriptions. In other works, term mismatch problems have been addressed through the use of WordNet (Wang and Stroulia, 2003; Kokash et al., 2006), the application of Latent Semantic Indexing (LSI) (Sajjanhar et al., 2004; Paliwal et al., 2007; Ma et al., 2008; Wu et al., 2008, 2009; Pan and Zhang, 2009; Paliwal et al., 2012), Probabilistic LSI (PLSI) and Latent Dirichlet Allocation (LDA) (Cassar et al., 2011, 2013). From the literature survey, we state the open questions in the state-of-the-art on service retrieval: **Q1**: Which model has the best effectiveness between LDA and classical LSI? **Q2**: Is it possible to increase the effectiveness of LSI with other matrix factorisation mod-

els? **Q3**: Which kind of model has the best effectiveness between LSI-based models or query expansion?

2.1 Introduction

In 1998 was the inception of SOAP-based Web services[1] (Snell et al., 2002, pg. 152). Distributed systems at larger scale than ever are developed due to this technology.

Searching for services became an urgent need. This happened because of the increasing adoption of this technology. As a consequence, in 2000 the standard UDDI[2] is proposed for discovering services[3]. It specifies an interface for publishing and looking for services. However, since 2006 this standard is discontinued by Microsoft, IBM, and SAP[4]. Nowadays, any public large-scale UDDI registry does not longer exist.

UDDI has two drawbacks:

1. On one hand, publishing advertisements of services is easier than using UDDI. Advertisements are automatically indexed by search engines (e.g. Google, Yahoo, etc.). On the other hand, with UDDI, users must keep updated the inputs of services.
2. On the other hand, the information stored within UDDI registries is not necessarily friendly enough for being read by humans. Indeed, in the literature about service discovery, authors advice that human users should not access the UDDI repository directly (Chappell and Jewell, 2002, pg. 96–97). Otherwise, users must check the uncountable entries to find the needed services. This is a time-consuming and tedious task.

Thus, discovering services is a motivating research challenge. In Section 2.2 we present the research trends in service discovery, however, we focus in the rest of

[1] A brief history of SOAP, http://www.xml.com/pub/a/ws/2001/04/04/soap.html, retrieved on February of 2015
[2] Universal Description Discovery and Integration, http://uddi.xml.org, retrieved on February of 2015
[3] History of UDDI, http://uddi.xml.org/milestones, retrieved on February of 2015
[4] http://www.infoworld.com/d/architecture/microsoft-ibm-sap-discontinue-uddi-registry-effort-777, retrieved on February of 2015

this chapter only in text description-based service retrieval (a.k.a., IR-based service discovery, or only service retrieval), because it is the main topic of our research. In Sections 2.3, 2.4, 2.5, and 2.6 we delve into each Information Retrieval (IR) model applied in prior research on text description-based service retrieval. In Section 2.7 we discuss what has been done in the past to identify the open questions in prior research. Finally, in Section 2.8 we summarise the chapter.

2.2 Research trends in service discovery

There are three trends in prior research about service discovery:

1. **Recommender Systems-based service discovery**. Given the users' preferences, the problem addressed in this trend is to suggest services to a user. Users' preferences are descriptions of long term needs dynamically updated and based on the behaviour of a user community. The recommendation of a service depends on users' preferences (or profiles) but not on queries they submit (Manikrao and Prabhakar, 2005; Birukou et al., 2007b,a; Kokash et al., 2007; Chan et al., 2010b,a, 2011, 2012). Besides users' preferences, another approach takes into account Quality of Service (QoS) values to compute recommendations of services (Zheng et al., 2009; Zheng and Lyu, 2010; Jiang et al., 2011; Chen et al., 2013).
2. **Semantic service discovery**. In this approach, ontologies describe services, then users submit queries in the form of a service, in a specific format such as SPARQL (Paolucci et al., 2002; Li and Horrocks, 2003; Sivashanmugam et al., 2004; Mouhoub et al., 2014). Services are retrieved if the concepts of its inputs and outputs match those of inputs and outputs of the service requested (Ngan and Kanagasabai, 2013).

   Other approaches in this stream are hybrid. For instance, the matchmakers called OWLS-MX (Klusch et al., 2006, 2009a), WSMO-MX (Klusch and Kaufer, 2009), and SAWSDL-MX (Klusch et al., 2009b) (herein "X" stands for five different versions, namely, M0-M4). These matchmakers combine logic-based semantic matching with token-based similarity techniques.

   Another hybrid matchmaker is known as iMatcher (Kiefer and Bernstein, 2008). It combines functional properties matching and non-functional properties match-

ing. Besides, this matchmaker uses logical reasoning in ontologies about services, statistics, and indexing techniques to match between requests and services.

3. **Text description-based service retrieval**. Currently, IR has embraced the analysis of several kind of objects in many mediums (e.g., text, images, audio, video, etc.) including Web services. In this thesis, our aim is in the direction of this stream. The problem that we address is to retrieve the most relevant services for a given query (see Section 1.4). In this context, a query is used in order to communicate a need to a service discovery system. The difference between this stream and the previous one (i.e., semantic service discovery) is the kind of query. In text description-based service retrieval, queries are composed by key terms, whereas in the semantic service discovery stream the queries have a specific format. The rest of this chapter, we delve into the state-of-the-art on text description-based service retrieval.

The IR models applied in prior research on service retrieval are as follows:

1. Vector space model (VSM) (Salton et al., 1975),
2. Latent Semantic Indexing (LSI) (Deerwester, 1988; Deerwester et al., 1990),
3. Probabilistic Latent Semantic Indexing (PLSI) (Hofmann, 1999), or
4. Latent Dirichlet Allocation (LDA) (Blei et al., 2003).

The general architecture of a text description-based service retrieval system is depicted in Figure 2.1. The components of this kind of systems are described as follows:

- **Preprocessor of queries**: This component carries out the following steps over each query:
  1. remove punctuations and symbols,
  2. change to lowercase all terms,
  3. stem or lemmatise each term, and
  4. remove stop words.
- **Preprocessor of descriptions**: The collection of service descriptions can not be indexed directly. Firstly, each service must be preprocessed in order to provide the most relevant information to the indexer component. In prior research, service descriptions are given either WSDL documents or OWL-S profiles. In both

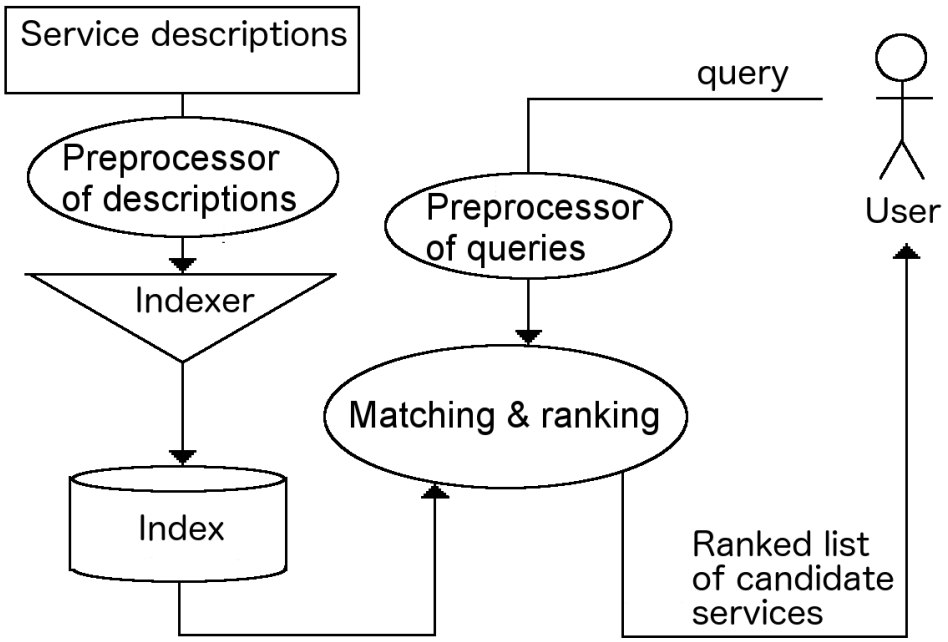

Fig. 2.1 Architecture of a text description-based service retrieval system

cases, the relevant terms are extracted from some sections of a WSDL document or an OWL-S profile. For example, in an OWL-S profile (see Figure 1.2, page 8) the relevant terms which describe a service may be extracted from tags <profile:serviceName> and <profile:textDescription> (Cassar et al., 2011, 2013). Once the terms are extracted, the **preprocessor of descriptions** carries out the same steps which **preprocessor of queries** executes over the terms of the query.

- **Indexer**: Given the preprocessed collection of service descriptions, this component makes the index of service descriptions. The index is a real-valued vector space, whose dimensions correspond to terms used to describe services in VSM (see Section 2.3), latent features in models based on LSI (see Section 2.4), or latent topics in LDA (see Section 2.6). Therefore, in the index, each service description has a vector representation.
- **Matching and ranking**: This component computes the retrieval function between the query and each service description. With the retrieval function is assigned a score to each service description. The greater the score of a service

is, the service is predicted to be more relevant to a user's query. First, this component compute the vector representation of the query according the indexing model (e.g. VSM or LSI). Thereafter, the cosine similarity is applied as retrieval function in order to compute the similarity of the vector representation of a query and each service description.

The main components of this architecture are in charge of indexing and ranking service descriptions. In the next Sections we emphasise and discuss how a collection of service descriptions are indexed and ranked, according every above mentioned models (i.e., VSM, LSI, PLSI, and LDA).

2.3 VSM-based service retrieval

In Vector Space Model (VSM) the index is a multidimensional real-valued vector space (Salton et al., 1975). Each vector into the space represents an indexed service description. Each unique term corresponds with a dimension of the vector space. Each element of a vector is a feature of a service description computed by the Term Frequency (TF) and Inverse Document Frequency (IDF). Let $TF_{td} \in \mathbb{R}$ be the number of times which the term ω_t appears into the service description s_d, and $IDF_t \in \mathbb{R}$ is defined as follows:

$$IDF_t = log\left(\frac{n}{DF_t}\right) \tag{2.1}$$

where n is the number of services, DF_t is the document frequency, which is the number of service descriptions that contain the term ω_t. According to the equation 2.1, IDF is high for terms used in few descriptions, however, IDF is low for those terms which are frequently used in many descriptions. In other words, IDF gives a lower weight to those terms which trend to be commonly used to describe many services. Such terms are less important during the retrieval process because this kind of terms do not contribute to highlight the differences between descriptions. Then, terms with low IDF are almost similar to stop words in particulars domains.

In VSM the TF and IDF define the weight of a term ω_t in a service description s_d as follows:

$$TFIDF_{td} = TF_{td} \times IDF_t \tag{2.2}$$

The $TFIDF_{td}$ of a term ω_t into a service description s_d is high, when the term frequently appears in few descriptions. Whereas $TFIDF_{td}$ is low if the term ω_t appears fewer times in a description, or whether it appears in many descriptions. $TFIDF_{td}$ is the lowest one for those terms which appears in almost all descriptions.

We have denoted by $C = \{s_d \subset V\}_{d=1}^{n}$ a collection of service descriptions (see Section 1.4), where n is the number of services, s_d is the text document which describes the d^{th} service, and $V = \{\omega_t\}_{t=1}^{m}$ is the vocabulary composed by the m significative terms (i.e., those terms which are not stop words) used to describe all services. Besides, we have denoted the user query as $Q \subset V'$, where V' is the vocabulary composed by the terms used to submit queries.

Thus, VSM is called in this way because vectors represent service descriptions (or documents in other domains). Let $Y_c = \{\mathbf{y}_i \in \mathbb{R}^m\}_{i=1}^{n}$ be a set of vectors, where each m-dimensional vector $\mathbf{y}_i$ represents the corresponding service description s_d, and m is the number of terms used to describe services. Let $\mathbf{Y}$ be a vector space, where the input vectors are given as column vectors. In other words, $\mathbf{Y}$ is an $m \times n$ matrix such that $\mathbf{Y} = (\mathbf{y}_1 \ldots \mathbf{y}_n)$, where the vector $\mathbf{y}_d$ is the d^{th} column of the matrix. Henceforth, the matrix $\mathbf{Y}$ shall be called a term-description matrix. In other domains of IR, $\mathbf{Y}$ is known as a term-document matrix in text retrieval, or a bag of visual features in image retrieval[5]. When VSM is applied to service retrieval, this matrix is the index of services descriptions, and is computed with the equation 2.2 as follows:

$$Y_{td} = TFIDF_{td} \tag{2.3}$$

[5] The bag of visual features is computed with a method which is different to the one used to compute a term-document matrix. However, with a bag of visual features the aim is to represent images through a set of vectors.

Moreover, in VSM a query is represented by a vector in the same m-dimensional real-valued vector space used to index the service descriptions. Let $\mathbf{q} \in \mathbb{R}^m$ be a vector which represents the query Q. This vector is computed with the equation 2.2. We can note that during the computation of this vector, a VSM-based system takes into account only terms that appear in the query, which are also part of the vocabulary composed by the m terms used to describe the services (i.e. terms that belong to the set $Q \cap V$).

VSM is unable to capture relations between terms submitted in the query and terms used to describe services. As a consequence, term mismatch problems affect the effectiveness of VSM. For instance, in the worse scenario of synonymy problems, the d^{th} service description s_d satisfies the query Q, but both sets of terms (s_d and Q) are disjoint (i.e., $s_d \cap Q = \emptyset$).

Given the vector $\mathbf{q}$ and the term-description matrix $\mathbf{Y}$, the retrieval function f most commonly used in prior research on VSM applied to service retrieval is the cosine of the angle between $\mathbf{q}$ and each column-vector of $\mathbf{Y}$ as follows:

$$f(Q, s_d) = cos(\mathbf{q}, \mathbf{y}_d) = \frac{\mathbf{q}^T \mathbf{y}_d}{\|\mathbf{q}\| \|\mathbf{y}_d\|} \tag{2.4}$$

where $d = 1, 2, \ldots, n$. The cosine similarity varies from 0 up to 1 because all elements in vectors are positive real numbers. Then, each time a VSM-based system receives a query, it computes the cosine similarity (i.e., the retrieval function) between the query vector and each description vector. Finally, the system sorts the resulting scores and it selects the top-k services descriptions, where $k \in \mathbb{N}$.

2.4 LSI-based service retrieval

Latent Semantic Indexing (LSI) (Deerwester, 1988; Deerwester et al., 1990) is an IR model proposed to index a collection of text documents (i.e., service descriptions in this context) according to their latent or hidden factors which explain the meaning of their contents, i.e., their semantics aspects.

LSI aims at reducing the vector space (the original index) in another of lower dimensionality. This new space is known as latent semantic space, such that through

hidden semantic factors are modelled the respective orthogonal projections of all column-vectors in a term-description matrix $\mathbf{Y}$ onto a semantic space.

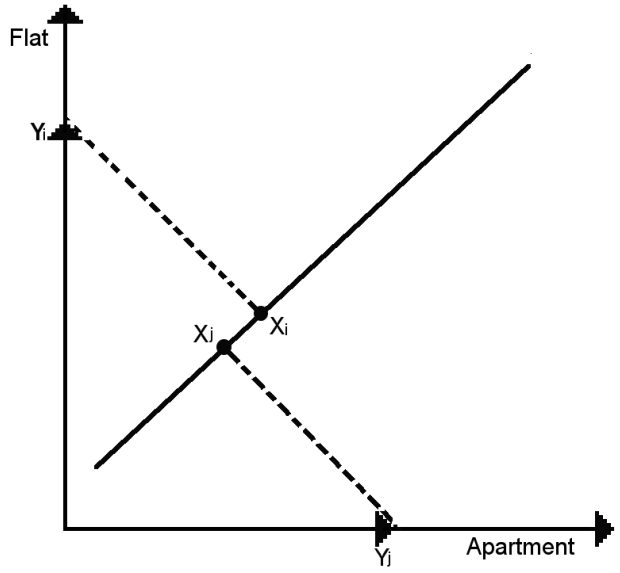

Fig. 2.2 Projection of two vectors of the columns of $\mathbf{Y} \in \mathbb{R}^{2\times n}$ in a lower dimensional space (i.e., the latent semantic space)

In Figure 2.2 we show an example where term mismatch problems affect the matching based on terms in VSM. For sake of simplicity, in this example we have considered two services, each of them is described by one term. Let us suppose that both services provide the same functionality, thereby, both terms *flat* and *apartment* are synonyms in the example. Figure 2.2 depicts two vectors, $\mathbf{y}_i$ and $\mathbf{y}_j$, which represent two descriptions onto a two-dimensional space. The similarity between both vectors is zero although the retrieval function is cosine or inner product. This is because both descriptions do not have terms in common, albeit the terms in the descriptions have the same meaning. Nevertheless, their projected vectors, $\mathbf{x}_i$ and $\mathbf{x}_j$, are near to each other in the latent semantic space (a one-dimensional space in the example, i.e., a straight right). The elements of each projected vector are the latent factors associated to their descriptions. Some basic

aspects of linguistic (e.g., synonymy and polysemy) can be captured in latent factors (Deerwester, 1988; Deerwester et al., 1990).

With LSI, a set of r latent factors are inferred from patterns found in the $TFIDF$ features of terms for each description. The number of factors are less than the number of either terms or descriptions. These factors explain all $TFIDF$ features by characterising descriptions and terms.

In case of service descriptions, latent factors may measure dimensions, which might be uninterpretable but meaningful. On the other side, for a given term, latent factors measure its occurrence in descriptions related to the corresponding hidden factors.

A joint latent semantic space represents service descriptions and terms. The inner product between their vector representations computes the $TFIDF$ feature. It measures the extent of the relationship between service descriptions and terms. Thus, latent factors are linear combinations of the original $TFIDF$ features. Let $\mathbf{x}_d \in \mathbb{R}^r$ be the representation of the description s_d, and $\mathbf{w}_t \in \mathbb{R}^r$ be the representation of the term ω_t, both in the latent semantic space. Elements in a vector $\mathbf{x}_d$ measure the extent to which the description s_d expresses latent factors, as well as elements in a vector $\mathbf{w}_t$ measure the extent to which the term ω_t appears in documents related to the corresponding factors. The inner product among both vectors yields the $TFIDF_{td}$ feature of the term ω_t into the description s_d, i.e., $Y_{td} = \mathbf{w}_t^T \mathbf{x}_d$. This can be written in a matrix form as follows:

$$\mathbf{Y} = \mathbf{W}^T \mathbf{X} \tag{2.5}$$

where $\mathbf{W} \in \mathbb{R}^{r \times m}$, $\mathbf{X} \in \mathbb{R}^{r \times n}$, $\mathbf{Y} \in \mathbb{R}^{m \times n}$, $\mathbf{W} = (\mathbf{w}_1 \quad \dots \quad \mathbf{w}_m)$, $\mathbf{X} = (\mathbf{x}_1 \quad \dots \quad \mathbf{x}_n)$, m is the number of terms into the descriptions, n is the number of services, and r is the number of latent factors such that $r < min(m, n)$. Therefore, behind LSI there is a matrix factorisation problem. This is solved by approximating a target term-document matrix $\mathbf{Y}$ as a product of two lower dimensional factor matrices, $\mathbf{W}$ and $\mathbf{X}$, where the common dimension r is smaller than m and n.

This problem is fixed in classical LSI by using Singular Value Decomposition (SVD see (Deerwester et al., 1990)). The strategy consists of choosing

$\mathbf{W}^T = \mathbf{U}_r\mathbf{D}_r$ and $\mathbf{X} = \mathbf{V}_r^T$, where $\mathbf{D}_r \in \mathbb{R}^{r\times r}$ is a diagonal matrix of the r largest singular eigenvalues of $\mathbf{Y}\mathbf{Y}^T$ and $\mathbf{Y}^T\mathbf{Y}$, $\mathbf{U}_r \in \mathbb{R}^{m\times r}$ and $\mathbf{V}_r \in \mathbb{R}^{n\times r}$ are orthonormal matrices, whose column vectors are eigenvectors of $\mathbf{Y}\mathbf{Y}^T$ and $\mathbf{Y}^T\mathbf{Y}$, respectively. Due to $\mathbf{X}$ is an orthonormal matrix, all its columns vectors in the latent semantic space are constrained to be orthogonal.

After factorising the term-description matrix $\mathbf{Y}$, the query vector $\mathbf{q} \in \mathbb{R}^m$ is projected onto the latent semantic space. Let $\mathbf{x} \in \mathbb{R}^r$ be the projected query vector. Given the following equation the aim is to find $\mathbf{x}$ such that:

$$\mathbf{q} = \mathbf{U}_r\mathbf{D}_r\mathbf{x}$$

multiplying $\mathbf{D}_r^{-1}\mathbf{U}_r^T$ in both sides of the equation yields the projection of the query vector as follows:

$$\mathbf{x} = \mathbf{D}_r^{-1}\mathbf{U}_r^T\mathbf{q} \tag{2.6}$$

Given the vector $\mathbf{x}$ and the factor matrix $\mathbf{X}$, the retrieval function f used in prior research on LSI applied to service retrieval is the cosine of the angle. This function is computed between $\mathbf{x}$ and each column-vector of $\mathbf{X}$ as follows:

$$f(Q, s_d) = cos(\mathbf{x}, \mathbf{x}_d) = \frac{\mathbf{x}^T\mathbf{x}_d}{\|\mathbf{x}\|\|\mathbf{x}_d\|} \tag{2.7}$$

where $d = 1, 2, \ldots, n$, and $\mathbf{x}_d$ is the d^{th} column of the matrix $\mathbf{X}$. So, each time a LSI-based system receives a query, after computing its vector representation, and projecting it on the semantic space, the system computes the cosine similarities between the projected query vector and each projected description vector. Finally, akin to VSM-based system, in a LSI-based system, the resulting scores are sorted, and the top-k services descriptions are selected.

2.5 PLSI-based service retrieval

In Probabilistic Latent Semantic Indexing (PLSI) model latent factors are called latent topics (Hofmann, 1999). Thus, let $\{\theta_i \in \mathbb{R}\}_{i=1}^r$ be a set of r latent topics

in a set of service descriptions. Each topic is represented by using a distribution of terms, or unigram language model. In the latter is assumed that the events, when different terms are used in a sequence, are stochastically independent. Therefore, it is assumed that the terms used to describe services may be sampled independently from a mixture of these r topics. So, to generate a term in a description s_d, one of the r topics is chosen regarding a topic selection probability distribution $\{p(i|s_d)\}_{i=1}^{r}$, where $p(i|s_d)$ is the probability of choosing the i^{th} topic θ_i given the document s_d, and then sample a term according the chosen term distribution. The probability of generating a term ω_t in a service description s_d is as follows:

$$p_{s_d}(\omega_t) = \sum_{i=1}^{r} p(i|S_d)p(\omega_t|\theta_i)$$

the log-likelihood of generating all the terms in a service description s_d is defined as follows:

$$log(p_{s_d}(\omega_t)) = \sum_{t=1}^{m} TF_{td} log\left(\sum_{i=1}^{r} p(i|S_d)p(\omega_t|\theta_i)\right)$$

In order to discover the term distribution of r latent topics, is necessary to fit the model to the collection of service descriptions, hence, the log-likelihood of such collection is defined as follows:

$$log(p(C|\Lambda)) = \sum_{d=1}^{n}\sum_{t=1}^{m} TF_{td} log\left(\sum_{i=1}^{r} p(i|s_d)p(\omega_t|\theta_i)\right)$$

where Λ is the set of parameters, i.e., $\Lambda = \{p(i|s_d)\} \cup \{\theta_i\}\ \forall_{i,d}$ such that $i = 1, 2, \ldots, r$ and $d = 1, 2, \ldots, n$. Through the hill climbing algorithm known as Expectation-Maximisation (EM), it is possible to compute either the maximum likelihood or the maximum *a posteriori* estimator. In order to estimate the maximum likelihood is necessary to solve the following optimisation problem: Given the collection of services descriptions C, the problem is to find the set of parameters Λ which maximise $p(C|\Lambda)$, namely:

$$\max_{\Lambda} p(C|\Lambda) \tag{2.8}$$

For this optimisation problem, the updating rules used in the EM algorithms are as follows:

$$p(z_{td} = i) = \frac{p^{\text{old}}(i|s_d)p^{\text{old}}(\omega_t|\theta_i)}{\sum_{j=1}^{r} p^{\text{old}}(j|s_d)p^{\text{old}}(\omega_t|\theta_j)} \tag{2.9}$$

$$p^{\text{new}}(i|s_d) = \frac{\sum_{t=1}^{m} TF_{td}p(z_{td} = i)}{\sum_{j=1}^{r}\sum_{t=1}^{m} TF_{td}p(z_{td} = j)} \tag{2.10}$$

$$p^{\text{new}}(\omega_t|\theta_i) = \frac{\sum_{d=1}^{n} TF_{td}p(z_{td} = i)}{\sum_{j=1}^{r}\sum_{d=1}^{n} TF_{td}p(z_{td} = j)} \tag{2.11}$$

where $z_{td} \in \mathbb{N}$ such that $z_{td} \leq r$. z_{td} is a hidden variable, it indicates which latent topic is used to generate the term ω_t in service description s_d.

For a description s_d, its topic distribution $\{p(i|s_d)\}_{i=1}^{r}$ is the new representation in the r-dimensional latent semantic space. Each dimension in this space is characterised by a term distribution θ_i. Thus, service descriptions and queries can be mapped to a semantic space and match them, for instance, through cosine similarity.

One advantage of PLSI over LSI is due to the possibility of incorporating extra knowledge by defining prior on parameters and using a maximum *a posteriori*.

Another advantage of PLSI over its classical version is that there is the possibility of adding more latent variables to PLSI in order to generate sophisticated semantic structures rather than the flat structure of r latent topics (e.g., a hierarchical structure).

Despite the advantage of PLSI over classical LSI, PLSI has two drawbacks, which we are mentioned below:

1. PLSI is not a generative model, hence, it is unclear how to assign probability to a service description which does not belong to the training dataset. In other words, $p(i|s_d)$ is computed regarding a specific service description s_d, thereby, those learned values of $p(i|s_d)$ are not able to be used for generating a new description different from s_d.
2. The set of parameter Λ for r latent topics is composed by r multinomial distributions of size m and n, thereby, there are $r(m+n)$ parameters, hence, the number of parameters grows linearly with the number of services. Therefore, there may be many local maximums for the likelihood function. As a result, the EM algorithm might find a non-optimal local maximum.

These shortcomings inspired the model Latent Dirichlet Allocation (Blei et al., 2003) which is described and discussed in next Section.

2.6 LDA-based service retrieval

The Latent Dirichlet Allocation (LDA) is a generative probabilistic model of a collection of discrete data (Blei et al., 2003) (i.e., service descriptions in the context of this research). This model is proposed to overcome the above mentioned drawbacks of PLSI. The strategy consists of defining $p(i|s_d)$ in a generative way by assuming that the topic distribution $p(i|s_d)$ has a Dirichlet prior. Once $p(i|s_d)$ is drawn from a Dirichlet distribution, it is used to generate all the terms in a service description s_d. With this strategy, the main problem of PLSI is addressed, namely, assigning a probability to a service description which does not belong to the training dataset. Besides, this strategy copes the second drawback of PLSI because the number of parameters is significantly reduced, hence, the problem of multiple local maximums is less severe in LDA than in PLSI.

The probability of generating a service description s_d is defined as follows:

$$p(s_d|\Lambda) = \int_{\Delta} Dir(\boldsymbol{\pi}|\alpha) \prod_{t=1}^{m} (\sum_{i=1}^{r} p(i|\boldsymbol{\pi}) p(\omega_t|\theta_i)) d\pi \tag{2.12}$$

where Δ is the $r-1$-dimensional simplex containing all possible values for $\boldsymbol{\pi}$ ($\boldsymbol{\pi} \in \mathbb{R}^r$), such that $\pi_i \geq 0$ and $\sum_{i=1}^{r} \pi_i = 1$. Let Λ the set of parameters such that $\Lambda = \{\alpha\} \cup \{\theta_i\}_{i=1}^{r}$. Each θ_i is a multinomial distribution over terms, whereas α is the parameter of the Dirichlet distribution.

The procedure to generate a service description s_d is as follows:

1. Choose a topic selection distribution $\boldsymbol{\pi}$ by sampling from the Dirichlet distribution. $\boldsymbol{\pi}$ is a multinomial distribution over all the r latent topics. If $\boldsymbol{\pi}$ gives a topic θ_i a high probability, as a consequence, θ_i is used very frequently to generate a term in the service description.
2. Generate a term by sampling a latent topic θ_i by using $p(i|\boldsymbol{\pi})$ and thus sampling a term from $p(\omega_t|\theta_i)$. This step is repeated m times to generate all terms in a service description s_d.

The topic distribution $\boldsymbol{\pi}$ serves as a new representation of service descriptions and queries in a r-dimensional latent semantic space.

Finally, we can note that the second step in the procedure is the same as in PLSI (see updating rules in Equations 2.10 and 2.11), however, the first step in PLSI (see updating rule in Equation 2.9) is different to LDA, the probability $p(i|\pi)$ does not depend on a specific service description to be generated. This explains why LDA is a generative model which is able to generate a new service description. Besides, we can note that in LDA there is only a set of $r(1+m)$ parameters for all the service descriptions, whereas in PLSI there are $r(m+n)$ parameters. As a consequence, in LDA there are more opportunities of finding an optimal maximum than in PLSI.

In next section, we discuss how all these above mentioned IR models have been applied for discovering services.

2.7 Discussion

The VSM has been applied in many approaches (see for example: (Wang and Stroulia, 2003; Platzer and Dustdar, 2005; Kokash et al., 2006; Lee et al., 2007; Crasso et al., 2008; Wu, 2012; Crasso et al., 2014)). In these works, a set of

WSDL documents composes the collection of service descriptions. Some of these approaches does not tackle the term mismatch problems (Platzer and Dustdar, 2005; Lee et al., 2007; Crasso et al., 2008; Wu, 2012; Crasso et al., 2014). Whereas, these problems have been addressed by expanding queries and WSDL documents with synonyms of their terms (Wang and Stroulia, 2003; Kokash et al., 2006). Synonyms are extracted from WordNet lexicon (Miller, 1995). Nevertheless, the query expansion based on the injection of synonyms significantly decreases precision because a term may have synonyms with different meanings depending of the context of the term in the query.

In other approaches, researchers applied LSI to cope term mismatch problems in the context of service discovery (Sajjanhar et al., 2004). However, factorising a matrix through SVD causes scalability issues in LSI. Therefore, other works handled this shortcoming instead of aiming to increase the effectiveness of LSI (Ma et al., 2008; Wu et al., 2008, 2009). However, scalability issues are out the scope of our research (see limitations of our research in Section 1.7, page 18).

LDA is another model based on latent factors in text documents. In this model the latent factors are topics, and their distribution is assumed to have Dirichlet prior. This model is applied to discover the latent topics from concepts contained in service descriptions written in OWL-S (Cassar et al., 2011, 2013). According to the results obtained in this research, LDA outperforms Probabilistic LSI (PLSI) (Cassar et al., 2011, 2013). Nonetheless, in the same study LDA has not been compared with other models used in prior research (e.g., LSI).

Another direction to deal with term mismatch problems is to use ontologies. Therefore, several works combine LSI and ontologies (Paliwal et al., 2007; Pan and Zhang, 2009; Paliwal et al., 2012). An ontology is used as a vocabulary to expand the query (Paliwal et al., 2012). In another hybrid approach K–means algorithm is used to divide the corpus in several clusters of documents (Pan and Zhang, 2009). Thereafter, given a query, SVD is applied on the most similar cluster (similar to (Ma et al., 2008)). However, this technique is complemented with a semantic–based matching, which is implemented on an ontology of services, by computing the similarity between service input and output parameters. At the end of the procedure, services are ranked according two both techniques.

The drawback of such ontology-based approach is that the human intervention is necessary, as ontologies must be built with the assistance of human experts of the domain. Therefore, the creation of ontologies is an expensive, time-consuming, tedious, and error-prone task (Gomez-Perez et al., 2003; Shamsfard and Barforoush, 2004). This is why we have decided not to design and build any ontology.

Most of the prior research on IR-based service discovery has been evaluated with different test suites. Some of them had common data sources, such as XMethods[6], which has been a source of WSDL documents for other researches (Kokash et al., 2006; Paliwal et al., 2007; Lee et al., 2007; Ma et al., 2008; Wu et al., 2009; Hao et al., 2010; Paliwal et al., 2012). Moreover, in other works, experiments are carried out on the collection of WSLD services descriptions used in (Hess et al., 2004) (see for instance (Paliwal et al., 2007; Crasso et al., 2008; Ma et al., 2008; Wu et al., 2008, 2009; Paliwal et al., 2012)). Nevertheless, this collection does not include neither queries nor relevance judgments due to this is designed for machine learning applications.

In other research works are carried out experiments over the same test collection. In (Kokash et al., 2006), researchers collected 40 Web services from XMethods, and they reused part of the set of WSDL documents collected for matching services research conducted in (Stroulia and Wang, 2005). In (Kokash et al., 2006), authors have used 447 services from the original collection composed by 814 services, excluding a group of 366 unclassified WSDL documents, and another one that was not parsed correctly. Whereas in (Lee et al., 2007), authors have been used the same dataset utilised in (Kokash et al., 2006). In (Sajjanhar et al., 2004) collected 47 services, 22 with description, and in the rest were assigned default descriptions based on the topic from IBM UDDI registry. In (Wu, 2012) used the dataset collected in the study conducted in (Klusch and Kapahnke, 2008). In (Platzer and Dustdar, 2005), researchers did not carry out an evaluation. Finally, in (Cassar et al., 2011, 2013), authors have used the collection named OWLS-TC v3.0[7]. It is composed by 1007 service descriptions written in OWL-S, 29 queries, and a relevant answer set for each query.

[6] XMethods, see www.xmethods.org, not available anymore

[7] OWL-S Service Retrieval Test Collection, see http://projects.semwebcentral.org/projects/owls-tc/, retrieved on July of 2014

Nowadays, to the best of our knowledge, it does not exist any works which have completely compared all prior research works. This might be because there is no standard test collection for IR-based service discovery. This literature survey raises the following questions which are addressed in this research:

- **Q1**: Which model has the best effectiveness between LDA and LSI based on SVD?
- **Q2**: Is it possible to increase the effectiveness of LSI with other matrix factorisation models?
- **Q3**: Which kind of model has the best effectiveness between LSI-based models or query expansion?

2.8 Summary

In this chapter we introduced the research trends in service discovery, i.e., 1) recommender systems-based service discovery, 2) semantic-matching-based service discovery, and 3) text description-based service retrieval.

In this research, we have focused in the third trend, where the main challenge is to address term mismatch problems between brief service descriptions and queries.

Term mismatch problems have been addressed by expanding queries and service descriptions with synonyms of their terms (Wang and Stroulia, 2003; Kokash et al., 2006). Synonyms are extracted from WordNet lexicon (Miller, 1995). However, the query expansion based on the injection of synonyms significantly decreases precision because a term may have synonyms with different meanings depending of the context of the term in the query.

In other approaches, term mismatch problems have addressed by applying LSI (Sajjanhar et al., 2004), PLSI and LDA (Cassar et al., 2011, 2013), or by combining LSI and ontologies (Paliwal et al., 2007; Pan and Zhang, 2009; Paliwal et al., 2012), notwithstanding that the creation of ontologies is an expensive, time-consuming, tedious, and error-prone task (Gomez-Perez et al., 2003; Shamsfard and Barforoush, 2004).

From this analysis of the literature survey, we have formulated the following open questions in service retrieval which we are addressed in this research: **Q1**: Which

model has the best effectiveness between LDA and LSI based on SVD? **Q2**: Is it possible to increase the effectiveness of LSI with other matrix factorisation models? **Q3**: Which kind of model has the best effectiveness between LSI-based models or query expansion?

In the next Chapter we describe two matrix factorisation models, which we have applied to LSI. In order to answer the question **Q2**, we experimentally compare these variations of LSI and the classical version based on SVD. Besides, in the next Chapter we also describe two models based on query expansion, which we apply for retrieving services. We empirically compare them against our family of LSI models and the models used in prior research on service retrieval. Through this experimental comparison we aim answer the question **Q3**.

Chapter 3
Proposed Information Retrieval models for indexing and retrieving services

Contents

Abstract In this Chapter we describe and explain the underlying mathematical models and algorithms of a family of IR models applied to retrieve services. To the best of our knowledge, none of them have been used in prior research on text description-based service retrieval. In order to study the effectiveness of Latent Semantic Indexing (LSI) with other matrix factorisation models, we apply Non-negative Matrix Factorisation (NMF) over LSI, and we propose a factorisation model based on Minimisation the Squared Error (MSE), which is applied to LSI. Besides, we apply other two models based on query expansion. In the first one, queries are expanded with synonyms from WordNet, but taking into account the parts of speech of the query. In the second model, queries are expanded with terms extracted from a co-occurrence thesaurus.

3.1 Introduction

In order to fulfil the aim proposed in this research, we apply IR models for discovering services. To the best of our knowledge, none of them have been used in prior research on text description-based service retrieval. Among these models, two of them are based on Latent Semantic Indexing (LSI) via two different factorisation models, i.e., Non-negative Matrix Factorisation (NMF), and factorisation based on Minimisation the Squared Error (MSE). Both models are proposed to be empirically compared with the classical LSI applied in the state-of-the-art on service retrieval. With the results of this empirical comparison, we aim to answer the research question **Q2**, hence, knowing whether the underlaying matrix factorisation model influences the effectiveness of the LSI model. The other two models are based on query expansion. The first one is based on natural language processing techniques and the extraction of synonyms from the lexicon called WordNet. In the second model, queries are expanded with terms extracted from a co-occurrence thesaurus. The last both models are proposed to be empirically compared with the two firsts in order to answer the research question **Q3**, thereby identify which kind of models has the best effectiveness between those based on LSI or query expansion.

The rest of the chapter is outlined as follows: First at all, in Section 3.2 we explain the preprocessing of service descriptions, which precedes the indexing process. In Sections 3.3 and 3.4 we describe Latent Semantic Indexing (LSI) models via minimising the square error and Non-negative Matrix Factorisation (NMF), respectively. In Sections 3.5 and 3.6 we detail the models based on the expansion of queries via WordNet and a co-occurrence thesaurus, respectively. Finally, in Section 3.7 we summarise the chapter.

3.2 Preprocessing of service descriptions

Preprocessing of a collection of service descriptions based on OWL-S documents involves the following steps:

1. Extract all terms from tags <profile:serviceName> and <profile:textDescription> (see Figure 1.2, page 8). In this study the first tag was assumed to be codified ac-

cording to camel case convention, i.e., the practice of writing identifier composed by several terms that start with a capital letter (e.g., nonNegativeMatrixFactorisation). Besides, it was assumed the terms in first tag to be separated by spaces or underscore character.
2. Remove punctuations and symbols.
3. Change to lowercase and lemmatise all terms. Stemming increases recall but decreases precision, hence, we adopted lemmatisation instead of stemming in order to get the base of dictionary form (a.k.a., lemma) of each term. We used the Northwestern University lemmatiser called MorphAdorner[1].
4. Remove stop words.
5. Compute the *Term Frequency* and *Inverse Document Frequency* (TFIDF) (Salton et al., 1975) in order to obtain the term-description matrix $\mathbf{Y} \in \mathbb{R}^{m \times n}$, where m and n are the number of terms and documents, respectively.

3.3 Latent Semantic Indexing via Minimising the Squared Error

Latent Semantic Indexing (LSI) aims at inferring a set of r latent (hidden) factors from patterns found in the $TFIDF$ features (see Section 2.4, page 28). The number of factors is less than the number of either terms or documents. These factors explain these $TFIDF$ features by characterising documents and terms. In case of documents, latent factors may measure dimensions, which might be uninterpretable but meaningful. For a given term, latent factors measure its occurrence in service descriptions related to the corresponding hidden factors.

Let $\mathbf{x}_d \in \mathbb{R}^r$ be the vector representation of a service description s_d, and $\mathbf{w}_t \in \mathbb{R}^r$ be the vector representation of the term ω_t, both in a r-dimensional latent semantic space. As we have mentioned in Section 2.4, elements of the vector $\mathbf{x}_d$ measure the extent to which the service description s_d expresses r latent factors, as well as elements of the vector $\mathbf{w}_t$ measure the extent to which the term ω_t appears in service descriptions related to the corresponding r factors. The in-

[1] MorphAdorner, http://devadorner.northwestern.edu/maserver/lemmatizer, retrieved on July of 2014

ner product between both vectors yields the $TFIDF_{td}$ feature of the term ω_t into the service description s_d: $Y_{td} = TFIDF_{td} = \mathbf{w}_t^T\mathbf{x}_d + E(\mathbf{w}_t, \mathbf{x}_d)$, here $E(\mathbf{w}_t, \mathbf{x}_d) = Y_{td} - \mathbf{w}_t^T\mathbf{x}_d$ is an error function, which measures the misfit between the $TFIDF_{td}$ feature Y_{td} and the inner product between both vectors $\mathbf{w}_t$ and $\mathbf{x}_d$.

Thus, computing these vectors ($\mathbf{w}_t$ and $\mathbf{x}_d$) can be defined as an optimisation problem as follows: Given $TFIDF_{td}$ feature Y_{td}, the aim is to estimate both vectors $\mathbf{w}_t$ and $\mathbf{x}_d$ such that minimise the absolute error function $\left|E(\mathbf{w}_t, \mathbf{x}_d)\right|$ or the squared error $\left(E(\mathbf{w}_t, \mathbf{x}_d)\right)^2$. We formalise this problem through the introduction of a cost function (a.k.a. loss function), which is the overall measure of the risk (or cost) incurred in making any of the available decisions. The decision stage consists of choosing a specific estimate $\mathbf{w}_t^T\mathbf{x}_d$ of the value of Y_{td} for each couple of vectors $\mathbf{w}_t$ and $\mathbf{x}_d$. So, our choice of cost function is the regularised squared error, hence, the optimal solution is the one which minimise the following cost function:

$$\begin{aligned} J(\mathbf{w}_1, \ldots, \mathbf{w}_m, \mathbf{x}_1, \ldots, \mathbf{x}_n) = \frac{1}{2}\sum_{t=1}^{m}\sum_{d=1}^{n}(Y_{td} - \mathbf{w}_t^T\mathbf{x}_d)^2 + \\ + \frac{\lambda}{2}\left(\sum_{t=1}^{m}\mathbf{w}_t^T\mathbf{w}_t + \sum_{d=1}^{n}\mathbf{x}_d^T\mathbf{x}_d\right) \end{aligned} \tag{3.1}$$

where the factor $1/2$ is used for later convenience, m is the number of terms used to describe services, and n is the number of services. Moreover, the cost function J is quadratic therefore its derivatives with respect to both vectors are linear. As a result, the minimisation of the cost function J has a unique solution. λ is a regularisation parameter, whose value is empirically chosen to give the right trade-off between making the squared error function $\left(E(\mathbf{w}_t, \mathbf{x}_d)\right)^2$ small, and avoiding that the coefficients $\sum_{t=1}^{m}\mathbf{w}_t^T\mathbf{w}_t$ and $\sum_{d=1}^{n}\mathbf{x}_d^T\mathbf{x}_d$ reach large magnitudes. The cost function in 3.1 can be written in a matrix form as follows:

$$J(\mathbf{X}, \mathbf{W}) = \frac{1}{2}\|\mathbf{Y} - \mathbf{W}^T\mathbf{X}\|_F^2 + \frac{\lambda}{2}(\|\mathbf{X}\|_F^2 + \|\mathbf{W}\|_F^2) \tag{3.2}$$

where $\mathbf{W} \in \mathbb{R}^{r \times m}$, $\mathbf{X} \in \mathbb{R}^{r \times n}$, $\mathbf{Y} \in \mathbb{R}^{m \times n}$, $\mathbf{W} = (\mathbf{w}_1 \quad \dots \quad \mathbf{w}_m)$, $\mathbf{X} = (\mathbf{x}_1 \quad \dots \quad \mathbf{x}_n)$, vectors $\mathbf{w}_t$ and $\mathbf{x}_d$ are the t^{th} and d^{th} columns in the matrices $\mathbf{W}$ and $\mathbf{X}$, respectively. Besides, as we have mentioned in Section 2.4, the number of latent factors r is less than number of terms m and services n. The term-description matrix $\mathbf{Y}$ is approximated to the product of the two lower dimensional factor matrices $\mathbf{W}$ and $\mathbf{X}$ (i.e., $\mathbf{Y} \approx \mathbf{W}^T\mathbf{X}$) when the cost function J (as defined in Equation 3.2) is minimised, namely.

$$\arg\min_{\mathbf{X},\mathbf{W}} J(\mathbf{X}, \mathbf{W}) \tag{3.3}$$

All the terms in J (see Equation 3.2) are squared, hence, the cost function has a minimum global, i.e., J is a convex cost function. This is useful because gradient descent may be used to find the minimum global instead of more costly heuristics such as simulated annealing or genetic algorithms. Therefore, to minimise the cost function, the gradient descent is calculated by setting the derivative of the cost function J (see Equation 3.2) with respect to $\mathbf{W}$ as follows:

$$\frac{\partial J(\mathbf{X}, \mathbf{W})}{\partial \mathbf{W}} = \mathbf{X}(\mathbf{W}^T\mathbf{X} - \mathbf{Y})^T + \lambda\mathbf{W} \tag{3.4}$$

As a result, the gradient descent is:

$$\mathbf{W} \leftarrow \mathbf{W} - \eta(\mathbf{X}(\mathbf{W}^T\mathbf{X} - \mathbf{Y})^T + \lambda\mathbf{W}) \tag{3.5}$$

where η is the learning rate. This updating rule is used to estimate $\mathbf{W}$. Moreover, taking the derivative of the cost function J (see Equation 3.2) with respect to $\mathbf{X}$ and setting it equal to zero is obtained $\mathbf{X}$ as follows:

$$\frac{\partial J(\mathbf{X}, \mathbf{W})}{\partial \mathbf{X}} = \mathbf{W}(\mathbf{W}^T\mathbf{X} - \mathbf{Y}) + \lambda\mathbf{X} = 0$$

carrying out the multiplication between matrices $\mathbf{W}$ and $(\mathbf{W}^T\mathbf{X} - \mathbf{Y})$ yields

$$\mathbf{W}\mathbf{W}^T\mathbf{X} - \mathbf{W}\mathbf{Y} + \lambda\mathbf{X} = 0$$

adding $\mathbf{W}\mathbf{Y}$ in both sides of the equality produces

$$(\mathbf{W}\mathbf{W}^T + \lambda\mathbf{I})\mathbf{X} = \mathbf{W}\mathbf{Y}$$

we obtain $\mathbf{X}$ by multiplying for in both sides of the equality by $(\mathbf{W}\mathbf{W}^T + \lambda\mathbf{I})^{-1}$ as follows

$$\mathbf{X} = (\mathbf{W}\mathbf{W}^T + \lambda\mathbf{I})^{-1}\mathbf{W}\mathbf{Y} \tag{3.6}$$

This Equation is used to compute $\mathbf{X}$, whereas $\mathbf{W}$ is updated with the rule given in Equation 3.5.

The Algorithm 1 estimates the factor matrices by minimising the cost function J in order to solve the matrix factorisation problem in concern, i.e., finding the factor matrices such that $\mathbf{Y} \approx \mathbf{W}^T\mathbf{X}$. In classical LSI the latent semantic representation of service descriptions $\mathbf{X}$ is orthonormal. In contrast, the factorisation model described in this Section does not require the latent semantic space to be orthogonal. Figure 3.1 illustrates the geometrical differences between both matrix factorisation models. We expect that by factorising matrixes by minimising the square error is more likely to capture the latent semantic factors which better describe services than through classical LSI, because vectors in the latent semantic space might not be constrained to be orthogonal.

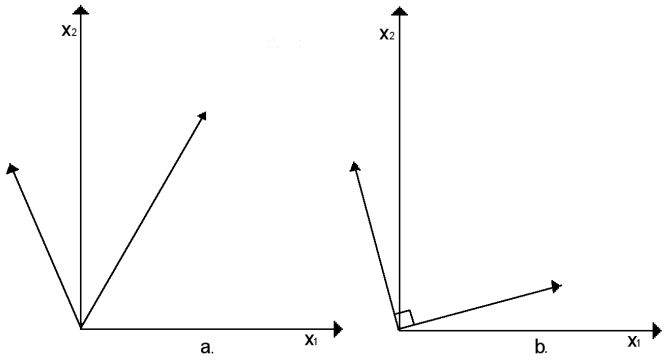

Fig. 3.1 **a.** Directions found by MSE. **b.** Directions found by SVD

Once the index is created, the latent factor representation of a query $\mathbf{q} \in \mathbb{R}^m$ is computed as follows:

Given:

- The term-description matrix $\mathbf{Y} \in \mathbb{R}^{m \times n}$, where m and n are the number of terms and services.
- The initial learning rate η_0.
- The regularisation parameter λ.
- The number of iterations L, for which the algorithm converges on a optimal solution.
- The number of latent factors r.

Goal: Estimate the factor matrices $\mathbf{W} \in \mathbb{R}^{r \times m}$ and $\mathbf{X} \in \mathbb{R}^{r \times n}$ which minimise the cost function J (in Equation 3.2), causing that the algorithm to converge on a reasonable set of factor matrices such that solve the matrix factorisation problem ($\mathbf{Y} \approx \mathbf{W}^T\mathbf{X}$).

1. Initialise $\mathbf{W}$ to small random values
2. For each value i such that $i \in \{l\}_{l=1}^{L}$ repeat the following
   a. Compute the latent factor representation for service descriptions ($\mathbf{X}$), by applying the analytic solution in Equation 3.6
   $$\mathbf{X} \leftarrow (\mathbf{W}\mathbf{W}^T + \lambda\mathbf{I})^{-1}\mathbf{W}\mathbf{Y}$$
   b. Update the learning rate. This decreasing rate depends on the number of iterations, the regularisation parameter, and the initial learning rate (Bottou, 2010). As a result, the gradient descends faster at the beginning. Thereafter, the learning rate is smaller with each iteration to avoid oscillations or divergence
   $$\eta \leftarrow \eta_0/(1 + \eta_0\lambda i)$$
   c. Update $\mathbf{W}$ according to the rule presented in Equation 3.5, with $\mathbf{X}$ fixed at the last known version
   $$\mathbf{W} \leftarrow \mathbf{W} - \eta(\mathbf{X}(\mathbf{W}^T\mathbf{X} - \mathbf{Y})^T + \lambda\mathbf{W})$$
3. **Return** the factor matrices $\mathbf{W}$ and $\mathbf{X}$

Algorithm 1: Matrix factorisation via minimising the squared error

$$\mathbf{x} = (\mathbf{W}\mathbf{W}^T + \lambda\mathbf{I})^{-1}\mathbf{W}\mathbf{q} \tag{3.7}$$

With this latent factor representation $\mathbf{x} \in \mathbb{R}^r$, the similarity between service descriptions and the query, all of them projected onto the latent semantic space,

is computed by using similarity cosine between $\mathbf{x}$ and each column vector of the factor matrix $\mathbf{X}$ (see Equation 2.7 in Section 2.4).

3.4 Latent Semantic Indexing via Non-negative Matrix Factorisation

Another method to factorise $\mathbf{Y}$ is by estimating the factor matrices ($\mathbf{X}$ and $\mathbf{W}$) as two non-negative matrices which minimise the cost function:

$$J(\mathbf{X}, \mathbf{W}) = \frac{1}{2}\|\mathbf{W}^T\mathbf{X} - \mathbf{Y}\|_F^2 \tag{3.8}$$

This function can be written in a convenience way as follows:

$$\begin{aligned}\frac{1}{2}\|\mathbf{W}^T\mathbf{X} - \mathbf{Y}\|_F^2 &= \frac{1}{2}Tr((\mathbf{W}^T\mathbf{X} - \mathbf{Y})(\mathbf{W}^T\mathbf{X} - \mathbf{Y})^T) \\ &= \frac{1}{2}Tr(\mathbf{W}^T\mathbf{X}\mathbf{X}^T\mathbf{W} - 2\mathbf{W}^T\mathbf{X}\mathbf{Y}^T + \mathbf{Y}\mathbf{Y}^T) \\ = \frac{1}{2}(Tr(\mathbf{W}^T\mathbf{X}\mathbf{X}^T\mathbf{W}) &- 2Tr(\mathbf{W}^T\mathbf{X}\mathbf{Y}^T) + Tr(\mathbf{Y}\mathbf{Y}^T))\end{aligned}$$

Thus:

$$J(\mathbf{X}, \mathbf{W}) = \frac{1}{2}(Tr(\mathbf{W}^T\mathbf{X}\mathbf{X}^T\mathbf{W}) - 2Tr(\mathbf{W}^T\mathbf{X}\mathbf{Y}^T) + Tr(\mathbf{Y}\mathbf{Y}^T)) \tag{3.9}$$

In Non-Negative Matrix Factorisation (NMF) the problem is to minimise J subject to $X_{ij} \geq 0$ and $W_{kl} \geq 0$, where $1 \leq i \leq r, 1 \leq j \leq m, 1 \leq k \leq r, 1 \leq l \leq n$. This constrained minimisation problem is solved with a Lagrange multiplier. Let $\mathbf{A} \in \mathbb{R}^{m \times m}$ and $\mathbf{B} \in \mathbb{R}^{n \times n}$ be the Lagrange multipliers for constraints $X_{ij} \geq 0$ and $W_{kl} \geq 0$, respectively. The Lagrange is as follows:

$$\Lambda(\mathbf{X}, \mathbf{W}, \mathbf{A}, \mathbf{B}) = J(\mathbf{X}, \mathbf{W}) + Tr(\mathbf{A}\mathbf{X}^T) + Tr(\mathbf{B}\mathbf{W}^T) \tag{3.10}$$

By setting the derivatives of Λ with respect to $\mathbf{W}$ and $\mathbf{X}$ is obtained:

$$\frac{\partial \Lambda(\mathbf{X}, \mathbf{W}, \mathbf{A}, \mathbf{B})}{\partial \mathbf{W}} = \mathbf{X}\mathbf{X}^T\mathbf{W} - \mathbf{X}\mathbf{Y}^T + \mathbf{A} \tag{3.11}$$

$$\frac{\partial \Lambda(\mathbf{X}, \mathbf{W}, \mathbf{A}, \mathbf{B})}{\partial \mathbf{X}} = \mathbf{W}\mathbf{W}^T\mathbf{X} - \mathbf{W}\mathbf{Y} + \mathbf{B} \tag{3.12}$$

By using the Kuhn-Tucker conditions $A_{ij}X_{ij} = 0$ and $B_{ij}W_{ij} = 0$ is obtained the following Equations for X_{ij} and W_{ij}:

$$(\mathbf{X}\mathbf{X}^T\mathbf{W})_{ij}W_{ij} - (\mathbf{X}\mathbf{Y}^T)_{ij}W_{ij} = 0 \tag{3.13}$$

$$(\mathbf{W}\mathbf{W}^T\mathbf{X})_{ij}X_{ij} - (\mathbf{W}\mathbf{Y})_{ij}X_{ij} = 0 \tag{3.14}$$

From both equations the following updating rules are yielded

$$W_{ij} \leftarrow W_{ij}\frac{(\mathbf{X}\mathbf{Y}^T)_{ij}}{(\mathbf{X}\mathbf{X}^T\mathbf{W})_{ij}} \tag{3.15}$$

$$X_{ij} \leftarrow X_{ij}\frac{(\mathbf{W}\mathbf{Y})_{ij}}{(\mathbf{W}\mathbf{W}^T\mathbf{X})_{ij}} \tag{3.16}$$

The cost function J is non-increasing with these updating rules, and the convergence is guaranteed (for a proof, see (Lee and Seung, 2001)).

The Algorithm 2 carrys out NMF on the term-description matrix $\mathbf{Y}$ to estimate the two non-negative factor matrices $\mathbf{X}$ and $\mathbf{W}$ with the above mentioned updating rules in Equations 3.16 and 3.15.

The Algorithm 3 estimates the projection of a query $\mathbf{q} \in \mathbb{R}^m$ on a r-dimensional latent semantic space, $\mathbf{x} \in \mathbb{R}^r$, by using the following updating rule:

$$x_i \leftarrow x_i\frac{(\mathbf{W}\mathbf{q})_i}{(\mathbf{W}\mathbf{W}^T\mathbf{x})_i} \tag{3.17}$$

With the query and service descriptions projected onto the same latent semantic space, the cosine similarity (see Equation 2.7 in Section 2.4) is applied to identify relevant results.

Given:

- The term-description matrix $\mathbf{Y} \in \mathbb{R}^{m \times n}$, where m and n are the number of terms and services, respectively.
- The number of iterations L, for which the algorithm converges on a optimal solution.
- The number of latent factors r.

Goal: Estimate the factor matrices $\mathbf{W} \in \mathbb{R}^{r \times m}$ and $\mathbf{X} \in \mathbb{R}^{r \times n}$ which minimise the cost function J (in Equation 3.9), causing the algorithm to converge on a reasonable set of factor matrices such that solve the matrix factorisation problem ($\mathbf{Y} \approx \mathbf{W}^T\mathbf{X}$).

1. Initialise $\mathbf{W}$ and $\mathbf{X}$ to small non-negative random values
2. For each value k such that $k \in \{l\}_{l=1}^{L}$ repeat the following
   a. Given $\mathbf{X}$ estimate $\mathbf{W}$ with the updating rule in Equation 3.15.

$$W_{ij} \leftarrow W_{ij} \frac{(\mathbf{X}\mathbf{Y}^T)_{ij}}{(\mathbf{X}\mathbf{X}^T\mathbf{W})_{ij}}, \forall_{i,j \in \mathbb{N}} \text{ such that } 1 \leq i \leq r \text{ and } 1 \leq j \leq m$$

   b. Given $\mathbf{W}$ estimate $\mathbf{X}$ with the updating rule in Equation 3.16

$$X_{ij} \leftarrow X_{ij} \frac{(\mathbf{W}\mathbf{Y})_{ij}}{(\mathbf{W}\mathbf{W}^T\mathbf{X})_{ij}}, \forall_{i,j \in \mathbb{N}} \text{ such that } 1 \leq i \leq r \text{ and } 1 \leq j \leq n$$

3. **Return** the factor matrices $\mathbf{W}$ and $\mathbf{X}$

Algorithm 2: Non-Negative Matrix factorisation algorithm

In the same way as the model presented in Section 3.3, we expect that by factorising matrixes through NMF is more likely to capture the latent semantic factors which better describe services than by mean of classical LSI, because vectors in the latent semantic space might not be constrained to be orthogonal, besides, with NMF is guaranteed that each vector in latent semantic space has only non-negative values in every direction (see Figure 3.2).

Given:

- The query $\mathbf{q} \in \mathbb{R}^m$, where m is the number of terms used to describe services.
- The number of iterations L, for which the algorithm converges on a optimal solution.
- The factor matrix $\mathbf{W} \in \mathbb{R}^{r \times m}$.
- The number of latent factors r.

Goal: Estimate the $\mathbf{x} \in \mathbb{R}^r$, which is the vector representation of a query $\mathbf{q}$ in a r-dimensional latent semantic space.

1. Initialise $\mathbf{x} \in \mathbb{R}^r$ to small non-negative random values
2. For each value k such that $k \in \{l\}_{l=1}^{L}$ repeat the following
   - Given the factor basis matrix $\mathbf{W}$ estimate $\mathbf{x}$ with the updating rule in Equation 3.17
   $$x_i \leftarrow x_i \frac{(\mathbf{W}\mathbf{q})_i}{(\mathbf{W}\mathbf{W}^T\mathbf{x})_i}, \forall_{i \in \mathbb{N}}, \text{ such that } 1 \leq i \leq r$$
3. **Return** the vector $\mathbf{x}$

Algorithm 3: Projection of a query in a latent semantic space

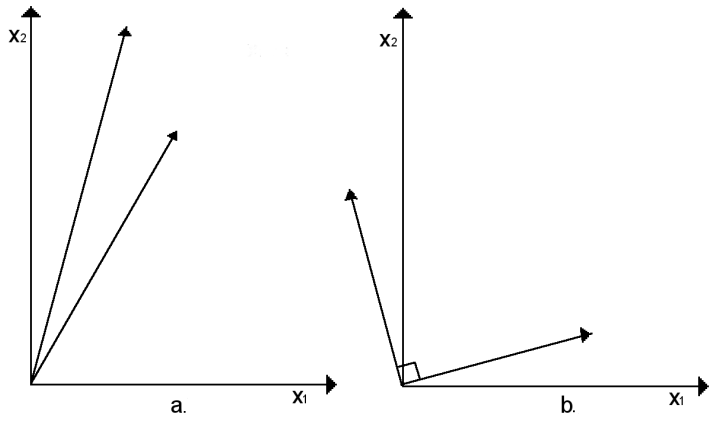

Fig. 3.2 **a.** Directions found by NMF. **b.** Directions found by SVD

3.5 Query Expansion via WordNet

The model presented in this section is based on VSM and query expansion rather than LSI. The query is expanded in order to cope synonymy problems. Notwith-

standing the problem with models based on query expansion is that this kind of models might significantly decrease precision (Wang and Stroulia, 2003; Kokash et al., 2006) (see Section 2.7). For instance, suppose a user submits a query like book a room, if the term *book* is considered a noun it may be expanded with WordNet with synonyms likewise: record, ledger, lever, Word of God, account book, Bible, al-Qur'an, etc. If all these synonyms are included into the query the system will retrieve services which are not related to reserve or book a room. However, if the same term is correctly identified as a verb, it may be expanded with WordNet with the following synonyms: hold and reserve. In this latter, the system will retrieve services for booking a room described with these synonyms, thus, the precision is increased.

Therefore, the problem in this context consists of tagging the parts of speech of the query in order to look for the synonyms in the thesaurus that will be injected into the query. In this study this problem has been tackled by using Apache OpenNLP[2] library to tag the parts of speech of a query. This library uses a probability model to predict the tag of a term based on its corresponding word type and its context in the query sentence. Thereafter, the synonyms associated with the term are sought in WordNet but considering it as an adverb, verb, noun or adjective.

Once the query is expanded, its vector representation is computed with the TF-IDF. Finally, services descriptions are ranked according their cosine similarity (see Equation 2.4 in Section 2.3) with the vector which represents the query.

3.6 Query Expansion via Co-Occurrence Thesaurus

Another alternative to WordNet consists of automatically generating a thesaurus by computing the Terms Similarity Matrix $\mathbf{\Theta} = \mathbf{Y}\mathbf{Y}^T$, where $\mathbf{\Theta} \in \mathbb{R}^{m \times m}$ and each element Θ_{ij} represents the similarity score between two different terms ω_i and ω_j, where $i \neq j$. This similarity score Θ_{ij} is the extent to which the term ω_i co-occurs with the term ω_j ($i \neq j$) in several service descriptions. Thereafter, the latent factors of each column vector of a term-description matrix are computed,

[2] Apache OpenNLP, http://opennlp.apache.org/, retrieved on July of 2014

by factorising it through the above mentioned methods in order to obtain factor matrices $\mathbf{W} \in \mathbb{R}^{r \times m}$ and $\mathbf{X} \in \mathbb{R}^{r \times m}$, such that $\mathbf{\Theta} = \mathbf{W}^T\mathbf{X}$. Let $V = \{\omega_i\}_{l=1}^{m}$ be a set of terms used to describe services (a.k.a. vocabulary), and let Q be a query such that $Q \subset V'$, where V' is the vocabulary composed by the terms used to issue queries. In this model, a term ω_i is used to expand a query whether it fulfils all the following conditions:

Given:

- The term-description matrix $\mathbf{Y} \in \mathbb{R}^{m \times n}$, where m and n are the number of terms and services.
- The query $Q \subset V'$, where V' is the vocabulary composed by the terms used to issue queries.
- The number of latent factors r.
- The threshold ρ.
- The vocabulary used to describe services V.

Goal: Expand the query Q with terms from a co-occurrence thesaurus. The expanded query shall be denoted as Q_e

1. Initialise the expanded query $Q_e \leftarrow Q$
2. Factorise the term similarity matrix $\mathbf{\Theta}$, where $\mathbf{\Theta} = \mathbf{Y}\mathbf{Y}^T$. The aim is to estimating the factor matrices $\mathbf{W} \in \mathbb{R}^{r \times m}$ and $\mathbf{X} \in \mathbb{R}^{r \times m}$ such that $\mathbf{\Theta} = \mathbf{W}^T\mathbf{X}$
3. For each term ω_j in the query Q which is also part of the vocabulary V (i.e., $\omega_j \in Q \cap V$) repeat the following
   - For each term ω_i in the vocabulary V repeat the following
     - If $cos(\mathbf{x}_i, \mathbf{x}_j) > \rho$ and $i \neq j$, then assign $Q_e \leftarrow Q_e \cup \{\omega_i\}$
4. **Return** the expanded query Q_e

Algorithm 4: Query expansion via co-occurrence thesaurus

- the term ω_i is not used in the query (i.e., $\omega_i \notin Q$),
- the term ω_i belongs to the thesaurus (i.e., $\omega_i \in V$), and
- the term ω_i is similar to another term ω_j included in the query. Besides, the term ω_j is part of the vocabulary used to describe services (i.e., $\omega_j \in Q \cap V$). The similarity between both terms depends on the extent of similarity between their vector representations in the factor matrix $\mathbf{X}$. Therefore, if all previous

conditions are accomplished, and the cosine similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$ (the i^{th} and j^{th} columns of $\mathbf{X}$) is bigger than a certain threshold ρ, then the term ω_j is injected in the query. The threshold ρ is estimated by means of experiments.

The Algorithm 4 carries out the expansion of queries. Once the query has been expanded, its vector representation is computed. Thus, given this vector, cosine similarity is applied to rank each vector which represents a service description.

3.7 Summary

In this chapter we have described our approach for preprocessing service descriptions in order to built an index for retrieving services.

Moreover, we have proposed two LSI-based models for retrieving services, whose underlying matrix factorisation models have not been applied in prior research on text description-based service retrieval. We have proposed both models to be compared with the classical LSI applied in the state-of-the-art on service retrieval. With this experimental comparison, our goal is to answer the open question **Q2** (see page 38) in order to know whether the matrix factorisation approach used to implement a LSI model affects its effectiveness.

In the first LSI-based model, we have defined the factorisation of a term-description matrix as an optimisation problem, whose aim is to minimise the squared error yielded from the projection of vectors, which represent service descriptions, onto the latent semantic space. In the second LSI-based model, we have applied NMF in order to factorise the term-description matrix.

Besides, we have proposed other two models based on query expansion. We aim to experimentally compare them with the above mentioned LSI-based models. The purpose of this experimental comparison is to answer the open question **Q3** (see page 38), hence, identify which kind of models have the best effectiveness between these two based on query expansion or those based on LSI.

In the first model based on query expansion, we apply natural language processing techniques to tag parts of speech in the query. In this way, each term in a query is identified as a verb, adjective, noun or adverb. According to the sense

of each term, its synonyms are retrieved from the lexicon named WordNet. These synonyms are injected into the query.

In the second model based on query expansion, we generate a co-occurrence thesaurus by computing a term similarity matrix from a term-description matrix. Each component of the term similarity is the extent to which two different terms co-occur in several service descriptions. Thereafter, the term similarity matrix is factorised in order to estimate the latent semantic factors of each term used to describe services. Thus, each term is represented in a vector space, therefore, the similarity between terms is determined by comparing their vector representations with cosine distance. Finally, the query is expanded with terms which are similar to those used to issue the query.

In classical LSI, service descriptions are orthonormal vectors into a latent semantic space. In contrast, factorisation models proposed in this section do not require the latent semantic space to be orthogonal. We expect that by factorising matrixes, either by minimising the square error or through NMF, it is more likely to capture the latent semantic factors which better describe services than through classical LSI, because vectors in the latent semantic space might not be constrained to be orthogonal, besides, with NMF is guaranteed that each vector in latent semantic space has only non-negative values in every direction.

In the next Chapter we present and analyse the results of the evaluation of all the models presented in this Chapter, and the most representative in the state-of-the-art on service retrieval.

Chapter 4
Implementation and evaluation

Contents

Abstract In this Chapter we present and analyse the results of systematic experiments carried out in our research. Firstly, the outcomes of our study suggest the expansion of queries via co-occurrence thesaurus outperforms the effectiveness of the other models studied in this research, hence, we have adopted this model to implement S^{3}niffer. Secondly, the outcomes reveal that the models based on Latent Semantic Indexing (LSI) outperform the effectiveness of Latent Dirichlet Allocation (LDA). Thirdly, despite the results reveal LSI via Non-negative Matrix Factorisation (LSI-NMF) outperforms the other LSI models, there not exist a statistical significative difference in the effectiveness between the underlying factorisation technique used in this study to implement LSI models. Finally, in this Chapter we describe the implementation details of S^{3}niffer.

4.1 Introduction

In this Chapter we report and discuss the outcomes of this research. We have conducted an experimental evaluation of each Information Retrieval (IR) model described in the previous chapter. Besides, we have evaluated those prominent models used in the state-of-the-art on service retrieval.

Our purpose with the empirical study discussed in this chapter is twofold. First of all, verifying the fulfilment of the goal of this research, secondly, to answer the research questions raised from the literature survey on service retrieval (see Chapter 2).

Moreover, among the IR models proposed in this research, we have implemented in S^{3}niffer the model with highest effectiveness. This selection is done according the results achieved in this study.

We have implemented the proof-of-concepts of all models in R[1] language. We evaluated each IR model with its respective proof-of-concept. Whereas we have implemented the final version of S^{3}niffer in Java[2] language. In this chapter we delve into the implementation details of S^{3}niffer.

The rest of this chapter is outlined as follows: In Section 4.2 we present the experimental setting of the evaluation of the family of IR models. In Section 4.3 we present the results of the research. In Section 4.4 we analyse the results achieved through experiments conducted in this research. In Section 4.5 we describe the implementation details of S^{3}niffer. Finally, we summarise the chapter in Section 4.6.

4.2 Experimental setting

During experiments, we have used the fourth version of the OWL-S service retrieval test collection called OWLS-TC4[3]. This collection provides the descriptions of 1083

[1] The R Project for Statistical Computing, http://www.r-project.org/, retrieved on February of 2015

[2] The Java programming language web site, www.oracle.com/technetwork/java/index.html, retrieved on February of 2015

[3] OWL-S Service Retrieval Test Collection, http://projects.semwebcentral.org/projects/owls-tc/, retrieved on July of 2014

Web services from 9 different domains (i.e., education, medical care, food, travel, communication, economy, weapon, geography, and simulation). All services are written in OWL-S 1.1, however, these are still compatible with OWL-S 1.0. Besides, this collection includes a set of 42 test queries associated with their relevance judgment provided by several users. A pooling strategy (as used in TREC[4]) is conducted to collect the relevance judgment set which is obtained from the top-100 results of participants of the S3 contest[5] in 2008.

Due to the judgment relevance has been graded in four different levels (i.e., value 3 for highly relevant, value 2 for relevant, value 1 for potentially relevant, and value 0 non-relevant), we have adopted the Normalised Discounted Cumulative Gain at 10 (NDCG@10) to measure the overall ranking effectiveness of each model rather than the Mean Average Precision (MAP).

OWLS-TC4 is the unique collection which exists to be applied service retrieval domain which has judgement relevance. Moreover, previous versions of this collection have been used for carrying out experiments in related recent research (Cassar et al., 2011, 2013).

On the other hand, we have tuned all the models, then the best settings for LSI via SVD (LSI-SVD), Minimising the Squared Error (LSI-MSE), and Non-negative Matrix Factorisation (LSI-NMF) are achieved when the number of latent factors are 147, 200, and 150, respectively (see Figure 4.1). The best setting for LDA is achieved with 180 latent topics (see Figure 4.1). Besides, we assessed LDA with several value for α. We found the best setting for α is the same suggested in (Griffiths and Steyvers, 2004). Moreover, to factorise matrixes by minimising the squared error, the best setting is obtained for the initial learning rate η_0 and the regularisation parameter λ, when their values are 0.2 and 0.001, respectively. Furthermore, the co-occurrence thesaurus is generated with the same three matrix factorisation models above mentioned. Several values of ρ are set for this approach (0.85, 0.90, 0.95, and 0.99). The best effectiveness of this models is obtained by generating the thesaurus and factorising the term similarity matrix via the technique to Minimise the Squared Error with 200 latent factors, and ρ equal to 0.95.

[4] Text Retrieval Conference, http://trec.nist.gov/, retrieved on February of 2015
[5] Semantic Service Selection, http://www-ags.dfki.uni-sb.de/ klusch/s3/, retrieved on February of 2015

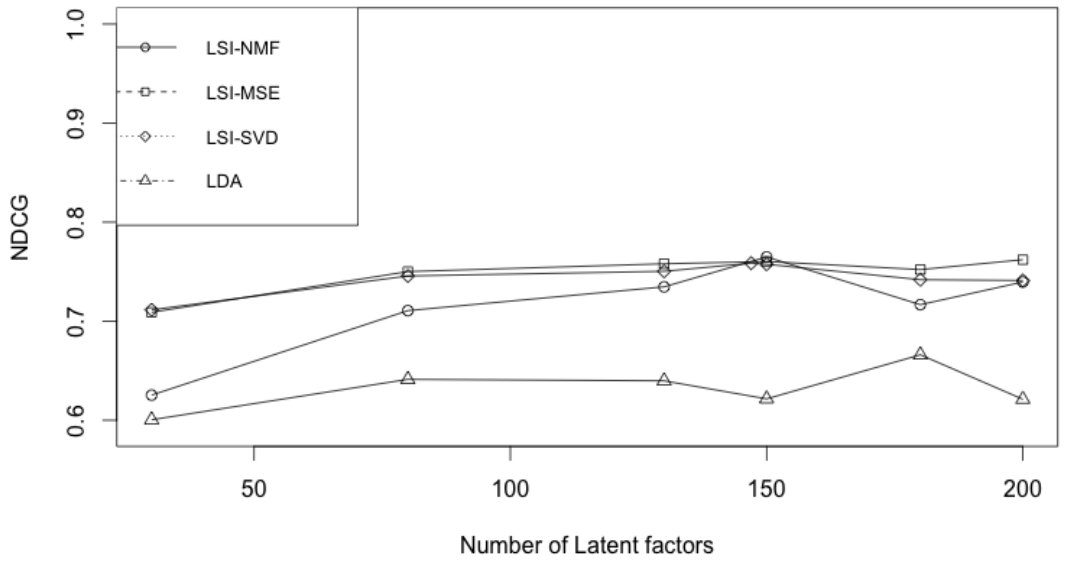

Fig. 4.1 Effectiveness (measured through NDCG@10) of LSI models and LDA given the number of latent factors (or topics)

With SVD the best performance is achieved with 220 latent factors and ρ equal to 0.90. Finally, with NMF the greatest effectiveness is obtained with 130 latent factors and ρ equal to 0.90.

4.3 The research results

Table 4.1 presents the results we obtained from the experiments. The three first rows show the retrieval effectiveness we obtained for those models applied in prior research on service retrieval, i.e., Vector Space Model (VSM), Latence Dirichlet Allocation (LDA), and Latent Semantic Indexing (LSI). Then the six last rows show results for our model extensions: Query Expansion via a Co-Occurrence Thesaurus (QECOT) automatically generated through SVD is called QECOT-SVD, QECOT generated using the method to Minimise the Squared Error (MSE), QECOT-MSE, QECOT generated through NMF, QECOT-NMF. Similarly, henceforth we denote

LSI-SVD the classical Latent Semantic Indexing (LSI) via SVD, LSI-MSE LSI via MSE, LSI-NMF LSI via NMF. Eventually, WordNet-based Query Expansion is shortened to WN-QE, Latent Direchlet Allocation to LDA, and Vector Space Model to VSM. The outcomes of the experiments suggest that QECOT-MSE outperforms all the above mentioned models in terms of effectiveness (see Table 4.1).

Table 4.1 Retrieval effectiveness.

Model	NDCG@10	Gain (%)
Models applied in prior research on IR-based service discovery		
VSM (baseline)	0.5435	N/A
LDA	0.6661	22.55
LSI-SVD	0.7586	39.57
Proposed family of models for text-based service retrieval		
LSI-MSE	0.7621	40.22
WN-QE	0.7645	40.66
LSI-NMF	0.7649	40.73
QECOT-NMF	0.7792	43.37
QECOT-SVD	0.7804	43.59
QECOT-MSE	**0.7897**	45.29

4.4 Analysis of the results

Table 4.2 Student's paired t-test on NDCG@10, with $p < 0.05$, to compare QECOT-MSE with other models applied in prior research on IR-based service discovery

Model QECOT-MSE	NDCG@10 **0.7897**	p-value	is statistically significant?
VSM	0.5435	3.47×10^{-7}	Yes
LSI-SVD	0.7586	0.08039	No
LDA	0.6661	7.613×10^{-5}	Yes

The results we got suggest that QECOT-MSE outperforms all the models studied in the paper. Indeed, Table 4.2 shows that the effectiveness of QECOT-MSE is better than LDA and VSM, which are the models applied for service retrieval

in (Wang and Stroulia, 2003; Platzer and Dustdar, 2005; Kokash et al., 2006; Lee et al., 2007; Crasso et al., 2008; Wu, 2012; Cassar et al., 2011, 2013), and the difference is statistically significative. Despite there is no difference which is statistically significative between the effectiveness of QECOT-MSE and LSI-SVD, the first model outperformed the second one in more queries. Indeed, in 5 queries both models had the same effectiveness, in 24 queries QECOT-MSE outperformed LSI-SVD, and only in 13 queries LSI-SVD has better effectiveness than QECOT-MSE (see Figure 4.2). Figure 4.2 depicts a comparison of the effectiveness of both models regarding each query used in the experiments. Points below the diagonal line correspond with queries where LSI-SVD outperformed QECOT-MSE (13 points). Whereas points above the line correspond with queries where QECOT-MSE outperformed LSI-SVD (24 points). Finally, points in the diagonal line correspond with queries where both models had the same effectiveness (5 points).

Another remarkable result suggest that LDA is outperformed by LSI-SVD (see Table 4.3), although the first models has been proposed to improve LSI in other IR domains. Perhaps the low effectiveness of LDA is due to the low frequency of the terms in the descriptions of the services. This fact answers the question **Q1** stated in Section 2.7 (see page 38).

LSI-NMF outperformed LSI-MSE, likewise the latter has a NDCG@10 greater than the one achieved with the classical LSI-SVD. This suggests that vectors in the latent semantic space might not be constrained to be orthogonal. Besides, with NMF is guaranteed that each vector in this space has only non-negative values in every direction. Notwithstanding the difference between NDCG@10 values is not statistically significant as it is shown in Table 4.3. This fact answers the question **Q2** stated in Section 2.7 (see page 38).

On the other hand, the evaluation reveals that LSI-NMF had a similar performance than WN-QE (see Table4.3). Maybe, due to this fact, LSI and query expansion have been the two main streams in the state-of-the-art on service retrieval. Nevertheless, QECOT-MSE has the best effectiveness, this suggests that query expansion might outperform LSI-models. This fact answers the question **Q3** stated in Section 2.7 (see page 38).

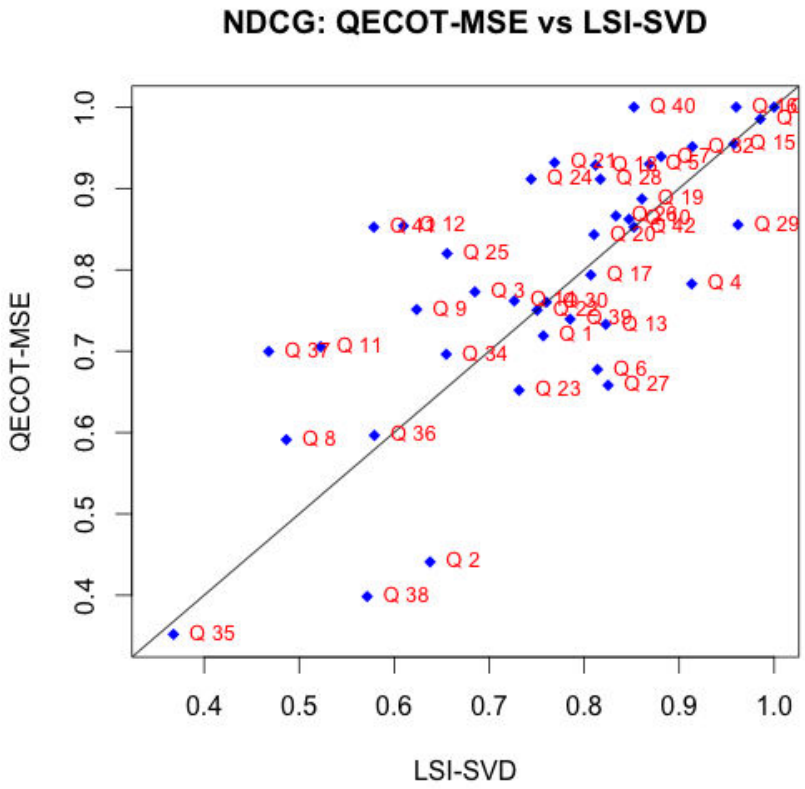

Fig. 4.2 Comparison between QECOT-MSE and LSI-SVD

Furthermore, between QECOT-MSE and WN-QE, the first has a better effectiveness. This might be due to QECOT-MSE automatically generates a thesaurus which is adapted to the knowledge of a specific domain. In this thesaurus the relationship among terms is determined by semantic latent factors, whereas WordNet is a general purpose thesaurus, which is not enable to be adapted to specific domains. For example, let us consider a query for orange services. With WordNet, this query may expand to orange services river tree orangeness. Nonetheless, if the original query refers to services provided by the French multinational telecommunications company, the query expanded with WordNet shall decrease the effectiveness of the retrieval process. On the other hand, the QECOT-MSE model is enable to learn from the collection of service descriptions, the latent factors which explain the

correlation among terms like orange and telecommunication, therefore this model might expand a query with terms that explain better a specific need.

Table 4.3 Student's paired t-tests on NDCG@10, with $p < 0.05$, to compare LSI-NMF with other LSI-based models, WN-QE, and LDA

Model	NDCG@10	p-value	is statistically
LSI-NMF	**0.7649**		significant?
Models applied in prior research on IR-based service discovery			
LSI-SVD	0.7586	0.6855	No
LDA	0.6661	8.37×10^{-4}	Yes
Proposed family of models for text-based service retrieval			
LSI-MSE	0.7621	0.8466	No
WN-QE	0.7645	0.9718	No

4.5 Implementation details of S^{3}niffer

S^{3}niffer is a service search system composed by four components as follows:

- Preprocessor: This component executes the procedure described in Section 3.2. This is written in R language and it is carried out off-line, before the creation of the index.
- Co-occurrence thesaurus creator: This component computes the factor matrix $\mathbf{X}$ from the term similarity matrix $\mathbf{\Theta}$ (see Section 3.6). The factorisation model implemented is Minimising the Squared Error (MSE), because with this method we achieved the best effectiveness (see Section 4.3). The factor matrix $\mathbf{X}$ is stored in a file to expand queries. This component is written in R language.
- Indexer: This component creates the index through the Vector Space Model. This component is written in Java with the Apache Lucene library[6].
- Service search engine: This component is indeed a Web service which has an operation called searchServices. This operation receives a text query and the number of candidate services (10 for default), and it returns a ranked list of candidate services. As the previous component, this one is also written in Java

[6] Apache Lucene library, http://lucene.apache.org/, retrieved on February of 2015

with the Apache Lucene library. However, before ranking services, this component expands the query with the stored matrix $\mathbf{X}$ through the Algorithm 4 explained in Section 3.6.

Taking into account the results we got from experiments, we have implemented the QECOT-MSE model in S^{3}niffer. We have adopted Apache Lucene library to implement the last two above mentioned components of S^{3}niffer. We made this decision because this library has been used in famous projects such as Krugle, SIREn, and LinkedIn (McCandless et al., 2010, pg. 383, 394, 409). Besides, Lucene supports the creation of optimised indexes, which are structured in several segments (McCandless et al., 2010, pg. 36). Hence, during the ranking process is not the whole index is not allocated in memory, instead of that, each segment is processed separately and the results are mixed later.

Another advantage of S^{3}niffer, which is derived from Lucene is that its query syntax supports:

- Wildcard characters (? and *) in the terms that belong to a query, e.g., to look for flash or flush, the query fl?sh may be used. But, if a wildcard for multiple characters is required, e.g., to seek surfing, surfer, or surf, then the query surf* may be used. Both wildcard characters might be used in the middle of a term. Nonetheless, any of them can not be used at the beginning.
- Fuzzy search query based on the Damerau-Levenshtein similarity. In Lucene, at most, the query shall match terms up to 2 edits. It is useful to find terms which are similar in spelling. It is indicated in the query by using the tilde character (~) at the end of a single term, e.g., the query aide~ matches terms such as eider, aid, aim, aida, or ail.

Both features supported in the Lucene query syntax are useful when a user is not sure about how to spell terms to query services.

4.6 Summary

In this Chapter we have presented a study conducted to assess the effectiveness of a family of IR models, which we have applied for retrieving services. To the best of

our knowledge, these models have not been applied in prior research on IR-based service discovery. Moreover, we have compared the family of IR models with those models used in the state-of-the-art on service retrieval.

With this evaluation we aim to answer the questions raised from the state-of-the-art on service retrieval, and fulfil the goal of this research, namely, to implement in S^{3}niffer a model which outperforms the effectiveness of those used in prior research on service retrieval.

In this chapter were have described the experimental setting used for testing all the studied IR models. Besides, in this chapter presented and discussed the results of the research. The findings of the research results are as follows:

1. The outcomes of the experiments suggest the expansion of queries via co-occurrence thesaurus outperforms the effectiveness of the other models studied in this work.
2. The results suggest that LDA has a lesser effectiveness than all the LSI-models evaluated in this work. However, for further research, this model might be used for expanding queries.
3. The results reveal there not exist a statistically significative difference in the effectiveness between the underlying factorisation technique used in this study to implement LSI models. Moreover, LSI-based models had a similar effectiveness than the injection of synonyms from WordNet taking into account the parts of speech of the query.

Finally, due to the results suggest the expansion of queries via co-occurrence thesaurus outperforms the other models studied in this research, we have detailed its implementation in S^{3}niffer. Thus, S^{3}niffer is a IR system for discovering services, whose components have been written in R and Java languages. All its components are in charge of procedures like preprocessing of service descriptions or service retrieval. The searching component is a Web service which may be invoked to retrieve a ranked list of candidate services.

Chapter 5
Conclusions

5.1 Summary and conclusions

The main contribution of this research is a service search system named S^{3}niffer, whose Information Retrieval (IR) model has been designed to outperform the effectiveness of those models used in prior research on service retrieval. The effectiveness of service retrieval systems is affected by term mismatch problems, which are worse in the context of this research, due to service descriptions are shorter than usual documents (e.g., books, Web pages, etc.).

In prior research on service retrieval have been applied the following IR models:

1. Vector space model (VSM) (Salton et al., 1975),
2. Latent Semantic Indexing (LSI) (Deerwester, 1988; Deerwester et al., 1990),
3. Probabilistic Latent Semantic Indexing (PLSI) (Hofmann, 1999), or
4. Latent Dirichlet Allocation (LDA) (Blei et al., 2003).

Term mismatch problems in VSM have been tackled by expanding queries and WSDL documents with synonyms of their terms (Wang and Stroulia, 2003; Kokash et al., 2006). Synonyms are extracted from WordNet lexicon (Miller, 1995). Nevertheless, the query expansion based on the injection of synonyms significantly decreases precision because a term may have synonyms with different meanings depending of the context of the term in the query.

In other works, term mismatch problems have been handled through LSI (Sajjanhar et al., 2004; Ma et al., 2008; Wu et al., 2008, 2009). Besides, PLSI and LDA have been applied to overcome term mismatch problems. According to the

results obtained in (Cassar et al., 2011, 2013), LDA outperforms Probabilistic LSI (PLSI) (Hofmann, 1999). Nonetheless, in (Cassar et al., 2011, 2013) did not compare LDA with other models used in prior research (e.g., LSI).

Moreover, ontologies have been built and applied in prior research on service retrieval to deal with term mismatch problems (Pan and Zhang, 2009; Paliwal et al., 2007, 2012). Nevertheless, we have decided not to design and build any ontology because of the creation of ontologies is an expensive, time-consuming, tedious, and error-prone task (Gomez-Perez et al., 2003; Shamsfard and Barforoush, 2004).

Thus, from the literature review on service retrieval has raised the following questions which have been addressed in this research:

- **Q1**: Which model has the best effectiveness between LDA and LSI based on Singular Value Decomposition (SVD)?
- **Q2**: Is it possible to increase the effectiveness of LSI with other matrix factorisation models?
- **Q3**: Which kind of model has the best effectiveness between LSI-based models or query expansion?

In order to answer the open question **Q1**, we have carried out experiments to compare the effectiveness of LDA and the classical LSI based on SVD. From the results of the research, we conclude that LDA is outperformed by classical LSI.

The another contribution of this research aim to answer the research question **Q2**. We have applied two LSI models based on matrix factorisation which are different to SVD, namely, LSI-models via Non-negative Matrix Factorisation (NMF) and Minimising the Squared Error (MSE). From the results of the research, we conclude there not exist a statistically significative difference in the effectiveness between the underlying factorisation technique used in this study to implement LSI models.

For the purpose of answering the last research question **Q3**, we have applied two models based on query expansion. The first one expands the query by injecting synonyms extracted from WordNet, by taking into account the parts of speech of the query. The second one expands the query with terms extracted from a co-occurrence thesaurus. From the results of the research, we conclude that LSI-based models have a similar effectiveness than the injection of synonyms from WordNet

taking into account the parts of speech of the query. Furthermore, we conclude that the expansion of queries via co-occurrence thesaurus outperforms the effectiveness of the other models studied in this research.

Therefore, we have implemented in S^{3}niffer the expansion of queries via co-occurrence thesaurus. We have implemented S^{3}niffer in R and Java with Apache Lucene library. As a consequence, S^{3}niffer supports two important features of Lucene query syntax, i.e., wildcard characters and fuzzy search query. Both features are useful when a user is not sure about how to spell terms to formulate a query.

Another remarkable conclusion of this research is that S^{3}niffer may be used by software engineers and software developers to search services in order to consume them in their applications.

5.2 Suggestions for further research

For future research we suggest:

- study the effectiveness of models based on the expansion of queries and service descriptions through a co-occurrence thesaurus, which is generated from a collection of documents that belong to an external source of knowledge, such as Wikipedia.
- design other matrix factorisation models based on the kernel trick, and apply them on LSI and query expansion models based on a co-occurrence thesaurus.
- study the effectiveness of the integration among term similarity and language models for expanding queries.

References

Birukou, A., Blanzieri, E., D'Andrea, V., Giorgini, P., and Kokash, N. (2007a). Improving Web Service Discovery with Usage Data. *IEEE Software*, 24(6):47–54.

Birukou, R., Birukou, R., Blanzieri, E., Blanzieri, E., Giorgini, P., Giorgini, P., Kokash, N., Kokash, N., and Modena, A. (2007b). IC-Service: A service-oriented approach to the development of recommendation systems. In *ACM Symposium on Applied Computing. Special Track on Web Technologies*, pages 1683–1688. ACM Press.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Boldt, J. (1995). The common object request broker: Architecture and specification. Specification formal/97-02-25, Object Management Group.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In Lechevallier, Y. and Saporta, G., editors, *Proc. of the 19th International Conference on Computational Statistics*, pages 177–187. Springer.

Brown, N. and Kindel, C. (1998). Distributed component object model protocol — dcom/1.0.

Burstein, M., Hobbs, J., Lassila, O., Mcdermott, D., Mcilraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., and Sycara, K. (2004). OWL-S: Semantic markup for web services. Technical report, World Wide Web Consortium.

Cassar, G., Barnaghi, P., and Moessner, K. (2011). A probabilistic latent factor approach to service ranking. In *2011 IEEE International Conference on Intelligent Computer Communication and Processing*, pages 103–109.

Cassar, G., Barnaghi, P., and Moessner, K. (2013). Probabilistic matchmaking methods for automated service discovery. *IEEE Transactions on Services Computing*, (99):1–1.

Chan, N., Gaaloul, W., and Tata, S. (2010a). Web Services Recommendation Based on User's Behavior. In *Proc. of 7th International Conference on e-Business Engineering*, pages 214–221.

Chan, N. N., Gaaloul, W., and Tata, S. (2010b). Collaborative Filtering Technique for Web Service Recommendation Based on User-Operation Combination. In Meersman, R., Dillon, T. S., and Herrero, P., editors, *On the Move to Meaningful Internet Systems: OTM 2010*, pages 222–239. Springer.

Chan, N. N., Gaaloul, W., and Tata, S. (2011). A web service recommender system using vector space model and latent semantic indexing. *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, 0:602–609.

Chan, N. N., Gaaloul, W., and Tata, S. (2012). A recommender system based on historical usage data for web service discovery. *Service Oriented Computing and Applications*, 6(1):51–63.

Chappell, D. and Jewell, T. (2002). *Java Web Services*. O'Reilly, german edition.

Chen, X., Zheng, Z., Liu, X., Huang, Z., and Sun, H. (2013). Personalized QoS-Aware Web Service Recommendation and Visualization. *IEEE Transactions on Services Computing*, 6(1):35–47.

Crasso, M., Mateos, C., Zunino, A., and Campo, M. (2014). Easysoc: Making web service outsourcing easier. *Information Sciences*, 259(0):452–473.

Crasso, M., Zunino, A., and Campo, M. (2008). Easy web service discovery: A query-by-example approach. *Science of Computer Programming*, 71(2):144–164.

Deerwester, S. (1988). Improving Information Retrieval with Latent Semantic Indexing. In Borgman, C. L. and Pai, E. Y. H., editors, *Proc. of the 51st ASIS Annual Meeting*, volume 25. American Society for Information Science.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

DeRemer, F. and Kron, H. (1975). Programming-in-the large versus programming-in-the-small. *ACM SIGPLAN Notices*, 10(6):114–121.

Gomez-Perez, A., Corcho-Garcia, O., and Fernandez-Lopez, M. (2003). *Ontological Engineering*. Springer-Verlag New York, Inc.

Griffiths, T. L. and Steyvers, M. (2004). Finding scientific topics. *Proc. of the National Academy of Sciences*, 101(1):5228–5235.

Hao, Y., Zhang, Y., and Cao, J. (2010). Web services discovery and rank: An information retrieval approach. *Future Generation Computer Systems*, 26(8):1053–1062.

Heineman, G. T. and Councill, W. T., editors (2001). *Component-based Software Engineering: Putting the Pieces Together*. Addison-Wesley Longman Publishing Co., Inc.

Hess, A., Johnston, E., and Kushmerick, N. (2004). ASSAM: A tool for semi-automatically annotating web services with semantic metadata. In *Proc. of the International Semantic Web Conference*. Springer.

Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proc. of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57.

Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20(4):422–446.

Jiang, Y., Liu, J., Tang, M., and Liu, X. F. (2011). An effective web service recommendation method based on personalized collaborative filtering. In *Proc. of the International Conference on Web Services*, pages 211–218. IEEE Computer Society.

Kent, A., Berry, M. M., Luehrs, and Perry, J. W. (1955). Machine literature searching viii, operational criteria for designing information retrieval systems. *American Documentation*, 6(2):93–101.

Kiefer, C. and Bernstein, A. (2008). The creation and evaluation of isparql strategies for matchmaking. In Bechhofer, S., Hauswirth, M., Hoffmann, J.,

and Koubarakis, M., editors, *The Semantic Web: Research and Applications*, volume 5021 of *Lecture Notes in Computer Science*, pages 463–477. Springer.

Klusch, M., Fries, B., and Sycara, K. (2006). Automated Semantic Web Service Discovery with OWLS-MX. In *Proc. of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 915–922.

Klusch, M., Fries, B., and Sycara, K. (2009a). OWLS-MX: A Hybrid Semantic Web Service Matchmaker for OWL-S Services. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(2):121–133.

Klusch, M. and Kapahnke, P. (2008). Semantic web service selection with sawsdl-mx. In *Proc. of the 2nd International Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web*. CEUR-WS.org.

Klusch, M., Kapahnke, P., and Zinnikus, I. (2009b). Hybrid Adaptive Web Service Selection with SAWSDL-MX and WSDL-Analyzer. *The Semantic Web: Research and Applications*, pages 550–564.

Klusch, M. and Kaufer, F. (2009). WSMO-MX: A Hybrid Semantic Web Service Matchmaker. *Web Intelligence and Agent Systems*, 7(1):23–42.

Kokash, N., Birukou, A., and D'Andrea, V. (2007). Web Service Discovery Based on Past User Experience. In Abramowicz, W., editor, *Business Information Systems*, volume 4439, pages 95–107. Springer Berlin Heidelberg.

Kokash, N., van den Heuvel, W.-J., and D'Andrea, V. (2006). Leveraging Web Services Discovery with Customizable Hybrid Matching. In *Proc. of the 4th International Conference on Service Oriented Computing*, pages 522–528.

Lee, D. D. and Seung, H. S. (2001). Algorithms for non-negative matrix factorization. In *Advances In Neural Information Processing Systems*, pages 556–562. MIT Press.

Lee, K.-H., Lee, M.-Y., Hwang, Y.-Y., and Lee, K.-C. (2007). A framework for xml web services retrieval with ranking. In *Proc. of the International Conference on Multimedia and Ubiquitous Engineering, 2007.*, pages 773–778. IEEE Computer Society.

Li, L. and Horrocks, I. (2003). A software framework for matchmaking based on semantic web technology. In *Proc. of the 12th International Conference on World Wide Web*, pages 331–339.

Ma, J., Zhang, Y., and He, J. (2008). Web Services Discovery Based on Latent Semantic Approach. In *Proc. of the International Conference on Web Services*, pages 740–747. IEEE Computer Society.

Manikrao, U. S. and Prabhakar, T. V. (2005). Dynamic selection of web services with recommendation system. *International Conference on Next Generation Web Services Practices*, pages 117–121.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

McCandless, M., Hatcher, E., and Gospodnetic, O. (2010). *Lucene in Action, Second Edition: Covers Apache Lucene 3.0*. Manning Publications Co.

Microsystems, S. (1999). Java remote method invocation. Specification, Sun Microsystems.

Miller, G. A. (1995). Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41.

Mouhoub, M. L., Grigori, D., and Manouvrier, M. (2014). A Framework for Searching Semantic Data and Services with SPARQL. In Franch, X., Ghose, A., Lewis, G., and Bhiri, S., editors, *Service-Oriented Computing*, pages 123–138. Springer.

Na-Lumpoon, P., Lei, M., Caicedo-Castro, I., Fauvet, M.-C., and Lbath, A. (2013). Context-Aware Service Discovering System for Nomad Users. In *7th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2013)*.

Na-Lumpoon, P., Lei, M., Kamnardsiri, T., Lbath, A., and Fauvet, M.-C. (2012). Illustrating some issues raised when designing context-aware personalized services for mobile users. In *Proceeding of the 6th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2012)*.

Ngan, L. D. and Kanagasabai, R. (2013). Semantic Web Service Discovery: State-of-the-art and Research Challenges. *Personal Ubiquitous Computing*, 17(8):1741–1752.

Paliwal, A. V., Adam, N. R., and Bornhövd, C. (2007). Web Service Discovery: Adding Semantics through Service Request Expansion and Latent Semantic Indexing. In *Proc. of the International Conference on Services Computing*, pages 106—113. IEEE Computer Society.

Paliwal, A. V., Shafiq, B., Vaidya, J., Xiong, H., and Adam, N. R. (2012). Semantics-based automated service discovery. *IEEE T. Services Computing*, 5(2):260–275.

Pan, S.-L. and Zhang, Y.-X. (2009). Ranked Web Service Matching for Service Description Using OWL-S. In *Proc. of the International Conference on Web Information Systems and Mining*, pages 427–431.

Paolucci, M., Kawamura, T., Payne, T. R., and Sycara, K. P. (2002). Semantic matching of web services capabilities. In *Proc. of the First International Semantic Web Conference on The Semantic Web*, ISWC '02, pages 333–347. Springer-Verlag.

Papazoglou, M. P. and Heuvel, W.-J. (2007). Service oriented architectures: Approaches, technologies and research issues. *The International Journal on Very Large Data Bases*, 16(3):389–415.

Platzer, C. and Dustdar, S. (2005). A vector space search engine for web services. In *Proc. of the 3rd European IEEE Conference on Web Services*, pages 14–16. IEEE Computer Society Press.

Ponte, J. M. and Croft, W. B. (1998). A language modeling approach to information retrieval. In *Proc. of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281. ACM.

Rijsbergen, C. J. V. (1979). *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition.

Sajjanhar, A., Hou, J., and Zhang, Y. (2004). Algorithm for web services matching. In Yu, J., Lin, X., Lu, H., and Zhang, Y., editors, *Proc. of the 6th Asia-Pacific Web Conference*, volume 3007, pages 665–670. Springer Berlin Heidelberg.

Sakai, T. (2007). On the reliability of information retrieval metrics based on graded relevance. *Information Processing and Management*, 43(2):531–548.

Salton, G., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Shamsfard, M. and Barforoush, A. A. (2004). Learning ontologies from natural language texts. *International Journal of Human–Computer Studies*, 60:17–63.

Sivashanmugam, K., Verma, K., and Sheth, A. P. (2004). Discovery of Web Services in a Federated Registry Environment. In *Proc. of the International Conference*

on Web Services, pages 270–278. IEEE Computer Society.

Snell, J., Tidwell, D., and Kulchenko, P. (2002). *Programming Web Services with SOAP*. O'Reilly & Associates, Inc., Sebastopol, CA, USA.

Stroulia, E. and Wang, Y. (2005). Structural and semantic matching for assessing web-service similarity. *International Journal of Cooperative Information Systems*, 14:407–437.

Vaughan-Nichols, S. (2002). Web services: beyond the hype. *Computer*, 35(2):18–21.

Wang, Y. and Stroulia, E. (2003). Semantic structure matching for assessing web service similarity. In *Proc. of the 1st International Conference on Service Oriented Computing*, pages 194–207. Springer-Verlag.

Wu, C. (2012). WSDL Term Tokenization Methods for IR-style Web Services Discovery. *Science of Computer Programming*, 77(3):355–374.

Wu, C., Chang, E., and Aitken, A. (2008). An empirical approach for semantic web services discovery. In *19th Australian Conference on Software Engineering*, pages 412–421.

Wu, C., Potdar, V., and Chang, E. (2009). Latent Semantic Analysis – The Dynamics of Semantics Web Services Discovery. In *Advances in Web Semantics I: Ontologies, Web Services and Applied Semantic Web*, pages 346–373. Springer-Verlag.

Zhai, C. and Lafferty, J. (2004). A study of smoothing methods for language models applied to information retrieval. *ACM Transactions Information Systems*, 22(2):179–214.

Zheng, Z. and Lyu, M. R. (2010). Collaborative Reliability Prediction of Service-oriented Systems. In *Proc. of the 32nd ACM/IEEE International Conference on Software Engineering*, pages 35–44.

Zheng, Z., Ma, H., Lyu, M. R., and King, I. (2009). WSRec: A Collaborative Filtering Based Web Service Recommender System. In *Proc. of the International Conference on Web Services*, pages 437–444. IEEE.