Trust and Reputation Management: a Probabilistic Approach

Mohamad Mehdi

A Thesis
in
The Computer Science Department

Presented in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy (Computer Science) at
Concordia University
Montréal, Québec, Canada

May 2015

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By:	**Mohamad Mehdi**
Entitled:	**Trust and Reputation Management: a Probabilistic Approach**

and submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

Dr. S. Mudur ____________	Chair
Dr. Noura Faci ____________	External Examiner
Dr. Olga Ormandjieva ____________	Examiner
Dr. Abdulwahab Hamou-Lhadj ____________	Examiner
Dr. Peter Grogono ____________	Examiner
Dr. Nizar Bouguila ____________	Supervisor
Dr. Jamal Bentahar ____________	Co-supervisor

Approved ____________
Chair of Department or Graduate Program Director

____________ 20 ______ ____________

Dr. Amir Asif, Ph.D., Dean

Faculty of Engineering and Computer Science

Abstract

Trust and Reputation Management in Web Services: A Probabilistic Approach

Mohamad Mehdi, Ph.D.

Concordia University, 2015

Software architectures of large-scale systems are perceptibly shifting towards employing open and distributed computing. Web services emerged as autonomous and self-contained business applications that are published, found, and used over the web. These web services thus exist in an environment in which they interact among each other to achieve their goals. Two challenging tasks that govern the agents interactions have gained the attention of a large research community; web service selection and composition. The explosion of the number of published web services contributed to the growth of large pools of similarly functional services. While this is vital for a competitive and healthy marketplace, it complicates the aforementioned tasks. Service consumers resort to non-functional characteristics of available service providers to decide which service to interact with. Therefore, to optimize both tasks and maximize the gain of all involved agents, it is essential to build the capability of modeling and predicting the quality of these agents.

In this thesis, we propose various trust and reputation models based on probabilistic approaches to address the web service selection and composition problems. These approaches consider the trustworthiness of a web service to be strongly tied to the outcomes of various quality of service metrics such as response time, throughput, and reliability. We represent these outcomes by a multinomial distribution whose parameters are learned using Bayesian inference which, given a likelihood function and a prior probability, derives the posterior probability. Since the likelihood, in this case, is a multinomial, a commonly used prior is the Dirichlet distribution. We propose, to overcome several limitations of the Dirichlet, by applying two alternative priors such as the generalized Dirichlet, and Beta-Liouville. Using these distributions, the learned parameters represent the probabilities of a web service to belong to each of the considered quality classes. These probabilities are consequently used to compute the trustworthiness of the evaluated web services and thus assisting consumers in the service selection process. Furthermore, after exploring the correlations among various quality metrics using real data sets, we introduce a hybrid trust model that captures these correlations using both Dirichlet and generalized Dirichlet distributions. Given their covariance structures, the former performs better when modeling negative correlations while the latter yields better modeling of positive correlations. To handle composite services, we propose various trust approaches using Bayesian networks and mixture models of three different distributions; the multinomial Dirichlet, the multinomial generalized Dirichlet, and the multinomial Beta-Liouville. Specifically, we employ a Bayesian network classifier with a Beta- Liouville prior to enable the classification of the QoS of composite services given the QoS of its constituents. In addition, we extend the previous models to function in online settings. Therefore, we present a generalized-Dirichlet power steady model that predicts compositional time series. We similarly extend the Bayesian networks model by using the Voting EM algorithm. This extension enables the estimation of the networks parameters after each interaction with a composite web service. Furthermore, we propose an algorithm to estimate the reputation of web services. We extend this algorithm by leveraging the capabilities of various clustering and outlier detection techniques to deal with malicious feedback and various strategic behavior commonly performed by web services. Alternatively, we suggest two data fusion methods for reputation feedback aggregation, namely, the covariance intersection and ellipsoidal intersection. These methods handle the dependency between the information that propagates through networks of interacting agents. They also avoid over confident estimates caused by redundant information. Finally, we present a reputation model for agent-based web services grouped into communities of homogeneous functionalities. We exploit various clustering and anomaly detection techniques to analyze and identify the quality trends provided by each service. This model enables the master of each community to allocate the requests it receives to the web service that best fulfill the quality requirements of the service consumers. We evaluate the effectiveness of the proposed approaches using both simulated and real data.

Acknowledgements

First and foremost, all praises to Allah for guiding and blessing me throughout this long journey.

I would like to express my sincere gratitude to my supervisors Prof. Nizar Bouguila and Prof. Jamal Bentahar for their continuous support and guidance that were vital to the completion of this thesis. Their insightful feedback taught me valuable lessons for life-long career in research. I extend my gratitude to all the members of the examining committee including Prof. Olga Ormandjieva, Prof. Peter Grogono, and Prof. AbdelWahab Hamou-Lhadj for their valuable feedback and suggestions that definitely ameliorated my thesis.

Special thanks to all my colleagues and friends, in particular, Elise Epaillard, Claude Fachkha, Elias Bou-Harb, and Ali S. Bakhtiari for the fruitful discussions and their encouragements especially during tough times.

I also would like to take this opportunity to extend my heartfelt gratitude to my parents who crafted my education path with their sweat and lots of sacrifice. Without their constant support and endless prayers I wouldnt have reached thus far. My brothers and sister, thank you for your incessant encouragements and love. Your faith in me was a never-ending source of energy and determination.

Finally, last but not the least, I express my profound and sincere gratitude to my wife for her patience and unwavering support and love. I am specially grateful for her tolerance towards my endless working hours and over-occupied mind. She was definitely the bedrock upon which the last five years of my life have been built.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

The emergence of service oriented architecture (SOA), the competitiveness of nowadays global markets, and the agility of business processes have contributed to the increase in the number of published web services. SOA provided the necessay infrastructure for distributed computing in which web services are built as Internet-based applications. Web services are business applications deployed as autonomous and interoperable agents that are published, found, and used on the web. As presented in [1–3], and as defined by W3C, each web service is associated with an agent that acts on its behalf and oversees its performance, commitments, and availability details. In this setting, an agent-based web service is a service entity that exists in a computational environment and acts both individually and cooperatively with other peers in order to accomplish its goals. Being loosely coupled and platform and language independent, web services ensure a flexible integration of diverse systems especially for business-to-business and business-to-consumer applications. Examples of such applications exist in various domains including e-commerce activities [4], semantic web, and peer-to-peer networks [5]. Key players in the high-tech space that participated in the existence and development of web services include IBM, Microsoft, Amazon, and Google.

The web services environment is basically defined by the following:

- **WSDL:** is an XML-based standard language used for describing web services. The main elements of a wsdl document as described by W3C are:
  - *types:* This element defines the data types used by the web services.
  - *message:* This element defines the data being handled by the corresponding web service.
  - *portType:* This is the core element of wsdl documents. It describes the operations and input and output messages of a web service.
  - *binding: This element defines the protocol and data format of each port type defines in the previous element.*
- **SOAP:** is an XML-based simple object access protocol used for accessing web services.
- **UDDI:** stands for Universal Description, Discovery and integration. It is employed as a web service directory that is described in WSDL and communicates via SOAP. UDDI is is the medium between service consumers and service providers.

Web services have been argued to have many advantages of which we select the following noteworthy benefits:

- Ease of information sharing and processing: as companies grow in size, various business processes become more cumbersome including data entry and lookup, and applications integration. Different applications may be written in different languages, run on different platforms, and store data in different formats. Web services are designed to overcome this challenge through a set of standardized protocols that govern the communcation and data sharing between different applications.
- Cutting costs: Customized tools and patches to optimize business processes can get very expensive. Web service are relatively cheaper to design and maintain.

Two major problems that are prominent in the web service research community motivate the works developed in this thesis, namely, the web service selection and composition problems.

Web Service Selection: Web services are deluging the web with similar functionalities, rendering the service selection a challenging task. We use the terms agent-based services, agents, and web services interchangeably throughout this thesis. Figure 1.1 illustrates the web service

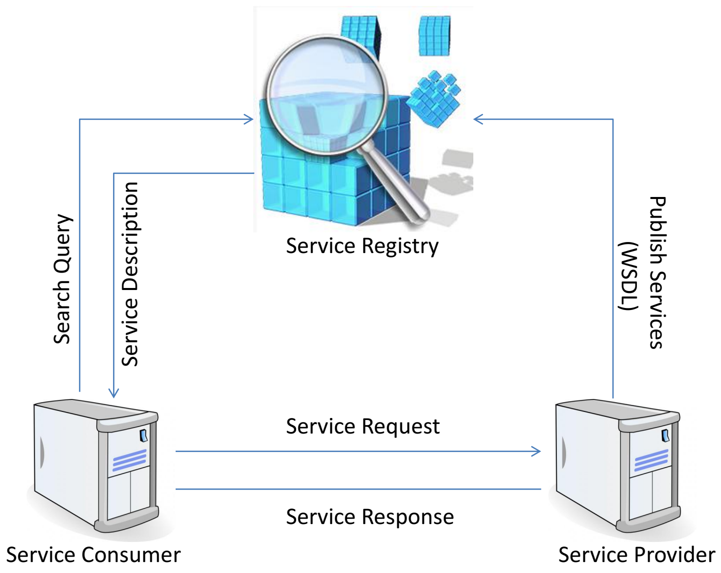


Figure 1.1: Web service selection

selection scenario that involves three main actors; the service consumer, service provider, and the registry in which a web service can be found (UDDI).

Given a search query, the UDDI returns the relevant web services. However, querying the UDDI is based on the functional requirements of the target web service only. The non-functional or quality characteristics required by the service consumer are utterly ignored. This increases the difficulty of choosing among of functionally similar web services the one that best meets all other requirements of a specific consumer.

Web Service Composition

The main motive behind the web services effort is to enable the interoperability between business applications by exploiting the Web standards, As such, the success of SOA was mainly influenced by the standardization of composition languages such as the Business Process Execution Language (BPEL). BPEL is based on the WSDL specifications and enables the definition of business processes as sets of integrated web services interactions. For a comprehensive description of BPEL and its characteristics, the reader can refer to [6]. Many business process management systems have been developed to enhance the automation the discovery, management and composition of business processes, including "SAPs NetWeaver, and IBMs WebSphere and BPM Suite" [7].

However, with the emergence of service oriented computing, the need for composing and integrating services from different providers becomes compelling. Indeed this emanates the same web service selection problem described above and the need to consider the quality characteristics of the constituent services. The difference between evaluating the quality of single and composite web services arises as a result of the dependencies between the composed services. Therefore, new composition systems or add-ons are needed to consider these dependencies when evaluating the quality of web services before selecting the ones to be included in compositions. Whether BPEL is used to describe the control logic that coordinates the web services in a process flow, or WSCI is employed to describe the exchanged messages between web services, the quality of service (QoS) characteristics are important to complement and optimize the composition process.

Trust and Reputation Based Solutions

Trust is a fundamental concern in our social life which is governed by the quality of our interactions. Due to the contingent nature of human behavior, this confidence is not stable, but rather a dynamic belief. A person you trust today might loose your trust tomorrow if s/he lets you down, has been caught lying, or betraying. The notion of human trust can be projected on the computational trust in the open world available on the web. This projection is rationalized due to the increase of various activities available on the web involving the interactions between various parties. Therefore, trust becomes a critical concern in this kind of loosely-controlled environment [8]. Trust has been defined in various ways in different contexts. In this thesis, we consider the definition given by the authors in [9]; a confidence one party has that the other party will act truthfully and reliably according to his/her claims.

Thus, in loosely-controlled open agent-based web service environment, trust becomes vital especially for selecting and composing services. Many applications require agents to coordinate with each other including e-commerce, supply chain management, semantic web and peer-to-peer networks [10]. Resources involved in the interactions among agents include time, money, and hardware usage. To make efficient use of these resources, it becomes extremely essential for an agent to be able to predict as accurately as possible the future behavior of its peers. These valuable resources play major roles in increasing productivity, enhancing the economic cycle, and saving the resources of individuals, businesses, and governments.

The majority of trust and reputation mechanisms has been employed in opens systems such as e-commerce, peer-to-peer-systems, and multi-agent systems. However, trust and reputation systems in web services has lately been an active research area with the continuous need for further investigations and studies. Therefore, our research questions are concerned with trust and reputation approaches related to web services even though we argue they can scale to other open systems.To compute the trustworthiness of web services, considering the web services QoS performance has become the norm. The trust and reputation models proposed in this thesis are also based on the QoS of the direct and indirect interactions among agents. The agent with the highest trust score will be then selected to provide the required functionalities.

1.2 Methodology

The trust and reputation models presented in this thesis are based on representing the quality of web services by various probabilistic distributions. The intuition behind this representation is that the outcome of an interaction with a web service is governed by the uncertainty concept. In other words, a service consumer is always skeptical about the quality a web service might provide. Modeling the quality as a statistical distribution is aligned with this fact and helps evaluating the uncertainties about the quality of web services.

Therefore, the basis of our models is to map the web services QoS outcomes into multinomial distributions. This is an intuitive mapping that appreciates the consumers need to have a simplistic way to view and assess the quality of one service and easily compare it to other web services. Estimating the parameters of the multinomial results in the probabilities of a web service to deliver a quality that belongs to one of the predefined quality classes. The probabilities provide a simple mechanism by which a service consumer would select, from a pool of functionally similar web services, the one that has the highest probability assigned to the highest quality class. To estimate these probabilities we apply, in Chapter 3, the Bayesian inference method which solves the poor estimates of the popular maximum likelihood estimation method in cases of low frequencies by adding a prior to the estimate the posterior. In our models, we present and compare three conjugate priors, Dirichlet, generalized Dirichlet, and Beta-Liouville. We argue for selecting the last two priors as they have more general covariance structures that overcome the limitation embedded in the strictly negative covariance of the Dirichlet distribution. In Chapter 3, we analyze the correlations among various QoS attributes and found that both positive and negative correlations exist. We leverage the correlation information to propose a trust model that provides a trust score based on the outcomes of multiple QoS attributes. It also computes the correlations among each pair of attributes using a hybrid model that takes advantage of both Dirichlet and generalized Dirichlet. These correlations are subsequently employed to enhance the estimation of the uncertainties in the delivered trust scores.

The previous models consider web services that function independently of others to provide a simple business function. However, larger business processes require the collaboration among multiple services which enter what is referred to as a service composition. Therefore, there is a need to devise a model that is capable of evaluating the trustworthiness of composite services and the influence of each of the constituent services in their overall quality. Chapter 4 is dedicated to trust models that are developed for these purposes. These models consist of representing the quality of composite services by Bayesian networks or mixtures of the probability distributions discussed in Chapter 3.

The previous learning approach belongs to the batch learning family. In other words, it requires a historical data on which the model is trained. However, such data is not always possible and the learning has to occur in near real-time. For single web services, we proposed a generalized power steady model that takes as input a time series of the quality classes proportions (Chapter 3). As for composite web services, we extended the Bayesian networks approach to function in online settings using two methods. The first applies the Voting-EM algorithm to learn the parameters of the networks when the underlying data is complete. The second method deals with missing data by employing the expectation maximization algorithm and a moving window that incorporates larger weights for newer data and smaller ones for older data within the same window (Chapter 4).

The aforementioned trust models are based on direct interactions among web services. The computed trust scores can be then viewed as the personalized opinion of each web service in the peers it interacts with. Nevertheless, the personal opinion is not always satisfactory to solve the web service selection problem especially when service consumers have no direct experience with certain services. This raises the need for reputation models that compute the global opinion about the web services quality based on the trust scores of multiple consumers. The three models proposed in Chapter 5 are developed to serve this purpose. The first model is based on two data fusion algorithms, namely covariance intersection and ellipsoidal intersection. Both methods are motivated by the need to combine two estimates while handling correlated errors (the former) and mutual information (the latter). The second model extends the correlation-based trust approach from Chapter 3 by combining trust scores that are perceived credible by the service consumer who requested these scores. The credibility of the trust scores is computed using a combination of a heuristic method and a cluster-based outlier detection algorithm. This reputation model deals with unfair referrals sent to badmouth or unreasonably praise a service. It also identifies various dynamic behaviors a web service may encounter; milking, building, and oscillating behaviors. The first occurs when a service performs well over a long period of time then suddenly drops its quality. The second behavior, building, is the opposite phenomenon. A web service's quality is boosted after a long period of low performance. The oscillating behavior is an alternation between both milking and building behaviors. The third model aims to demonstrate the potential of applying data mining techniques to evaluate the trustworthiness of communities of web services. These communities group web services that provide similar functionalities in order to better serve consumers by handling simultaneous requests and implementing a fault-tolerant architecture.

1.3 Thesis Contributions

This thesis aims to present various probabilistic approaches to solve the web service selection and composition tasks via QoS-aware trust and reputation. These approaches are built upon the modeling of the web services QoS metrics as various composite multi-simensional distributions, namely, multinomial Dirichlet, multinomial generalized Dirichlet, and multinomial-Beta Liouville. The main contributions of this thesis can be summarized as follows:

☆ **QoS Modeling Using Multi-Dimensional Composite Distributions:**
We propose to compute the QoS ratings based on the outcomes of multiple quality metrics. Subsequently, we model the counts of the QoS ratings, after a number of interactions, by various multi-dimensional statistical distributions including the multinomial-Dirichlet, multinomial-genralized Dirichlet and multinomial-Beta Liouville. These distributions offer, in comparison to the Beta-binomial used in current models, a more comprehensive QoS representation of evaluated web services. We also leverage the correlations among QoS metrics to devise a hybrid trust approach based on Dirichlet and generalized Dirichlet distributions. This approach aims to improve the accuracy of trust estimates and their corresponding uncertainties.

☆ **Online Trust Estimation Via Time-Series Forecasting Model:**
To assist service consumers in real-time, we introduce a time-series forecasting approach, based on a generalized-Dirichlet state space model. This approach follows the popular divide-and-conquer method by transforming a generalized-Dirichlet distribution into multiple Dirichlet distributions in a lower dimensional space.

☆ **Bayesian Networks and Mixture Models for Trustworthy Composite Services:**
Business processes, in the majority of cases, provide more than a single functionality, which necessitate the collaboration among various web services to form a composite service. To evaluate the performance of composite services and assist the composition of new web services, we propose multiple Bayesian networks classifiers and mixture models. These are capable of learning the dependencies between the composed web services and the responsibilities (contributions) of each web service in the overall quality of the composite service. We also extend the Bayesian networks model to estimate the future quality of composite web services in an online setting using the Voting EM algorithm. In other words, the estimated conditional probabilities of the QoS of composite web services are updated after every interaction. This addresses the dynamic behavior of web services that might offer a low quality interaction at time t and a better one at time $t+1$.

☆ **Data Fusion Methods for Reputation Feedback Aggregation:**
One of the main issues in considering the indirect interactions among web services in the trust computation is the aggregation of the reputation feedback. Kalman filter is an attractive method to combine multiple measurements (in this case, reputation feedback) and predict the state of a

dynamic system. However, Kalman filter is limited by the independence assumption between the estimation and measurement errors at time t and $t+1$, respectively. This assumption leads to under/over-estimated covariance matrices of the predictions. To overcome this limitation, we employ two data fusion methods that avoid this assumption and lead to more consistent estimates; covariance intersection and ellipsoidal intersection.

✰ **Reputation Management Using Clustering and Outlier Detection Techniques:** It has been recorded that the quality of web services fluctuate both intentionally and unintentionally. In this thesis, the interest is in the intentional fluctuations due to different malicious behavior. We propose a reputation management framework that captures different types of web service malicious behaviors including milking and building reputation. It also penalize web services that exhibit such behaviors by diminishing their reputation scores. This framework is based on clustering and outlier detection techniques.

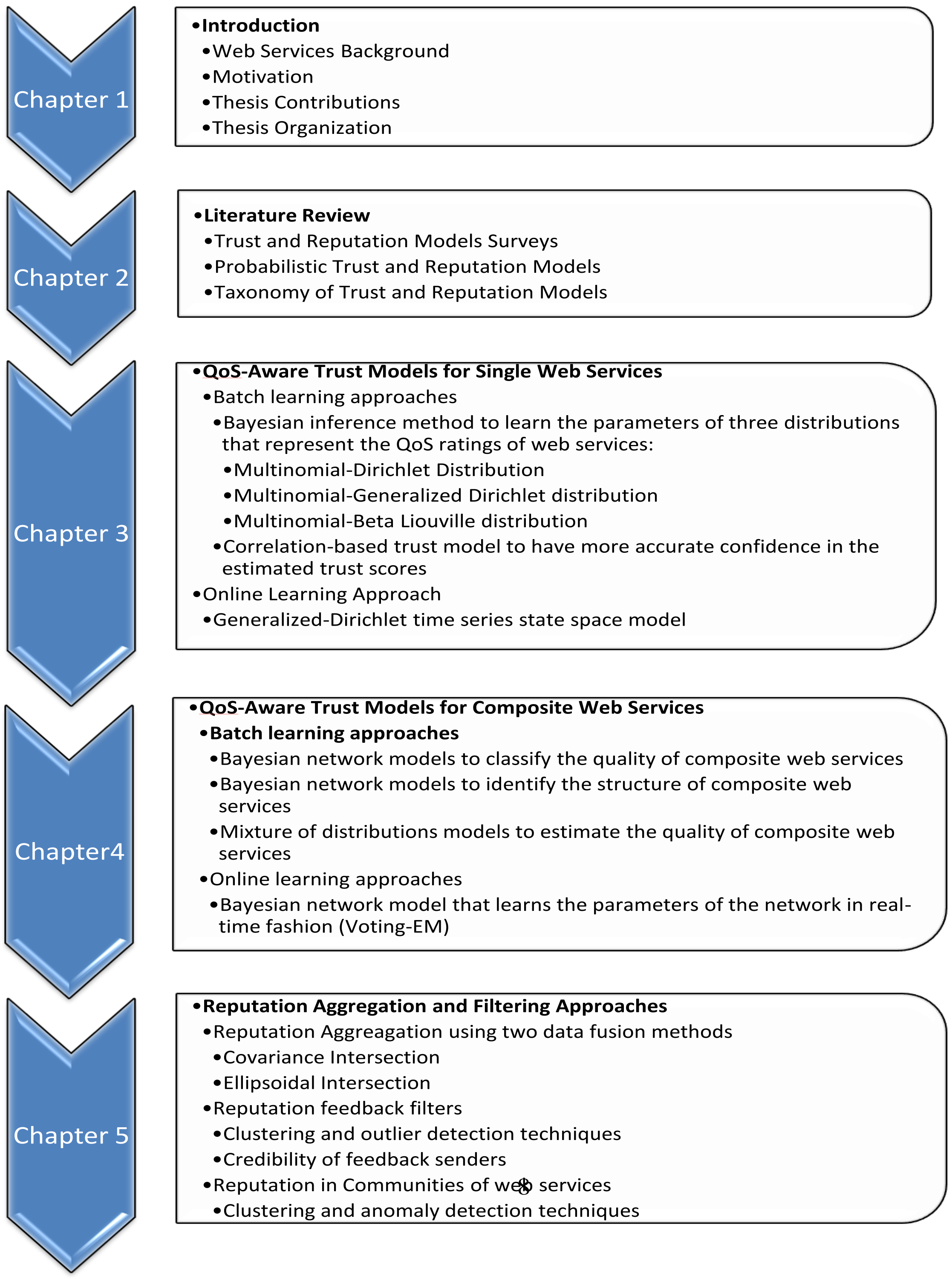
1.4 Thesis Organization

This thesis is organized into an introductory chapter, four chapters that describe the proposed models, and a closing chapter that summarizes the contributions and sets the stage for potential future works. This organization is illustrated in Figure 1.2.

❏ **Chapter 1:** overviews the web service selection and composition problems highlighting the notion of trust to solve these problems. The motivation of the proposed approaches is also clearly defined. Afterwards, the existing literature of trust and reputation systems is reviewed and the main contributions of the thesis are described.

❏ **Chapter 2:** presents a trust approach that is based on modeling the outcomes of various QoS metrics using three composite distributions; multinomial-Dirichlet, multinomial-generalized Dirichlet, and multinomial-Beta Liouville. This approach is further extended by considering the correlations among the considered QoS metrics. This extension aims to improve the estimated trust scores and provide an accurate uncertainty measure in these scores.

Chapter 1. Introduction



- **Introduction**
  - Web Services Background
  - Motivation
  - Thesis Contributions
  - Thesis Organization

- **Literature Review**
  - Trust and Reputation Models Surveys
  - Probabilistic Trust and Reputation Models
  - Taxonomy of Trust and Reputation Models

- **QoS-Aware Trust Models for Single Web Services**
  - Batch learning approaches
    - Bayesian inference method to learn the parameters of three distributions that represent the QoS ratings of web services:
      - Multinomial-Dirichlet Distribution
      - Multinomial-Generalized Dirichlet distribution
      - Multinomial-Beta Liouville distribution
    - Correlation-based trust model to have more accurate confidence in the estimated trust scores
- Online Learning Approach
  - Generalized-Dirichlet time series state space model

- **QoS-Aware Trust Models for Composite Web Services**
  - **Batch learning approaches**
    - Bayesian network models to classify the quality of composite web services
    - Bayesian network models to identify the structure of composite web services
    - Mixture of distributions models to estimate the quality of composite web services
  - Online learning approaches
    - Bayesian network model that learns the parameters of the network in real-time fashion (Voting-EM)

- **Reputation Aggregation and Filtering Approaches**
  - Reputation Aggreagation using two data fusion methods
    - Covariance Intersection
    - Ellipsoidal Intersection
  - Reputation feedback filters
    - Clustering and outlier detection techniques
    - Credibility of feedback senders
  - Reputation in Communities of web services
    - Clustering and anomaly detection techniques

Figure 1.2: Chapters Organization

Figure 1.2: Thesis Organization

❑ **Chapter 3:** proposes various Bayesian network models to estimate the trustworthiness of composite web services. Each of these models employs one of the QoS models described in Chapter 1. An alternative approach, based on generative mixture of multinomial Dirichlet distributions is also introduced to learn the distributions of the quality outcomes of the constituents web services given the quality outcomes of the composite web service.

❑ **Chapter 4:** extends the proposed trust models to function in online settings. Specifically, a generalized Dirichlet power steady model is proposed to model the time series that consist of the proportions of the quality classes of the evaluated web service over a period of time. Moreover, the Bayesian network models from Chapter 3 are extended to learn the networks parameters after each data point.

❑ **Chapter 5:** describes three models for reputation aggregation taking into account the existence of dishonest feedback and malicious behaviors. These models are based on different outlier detection algorithms, data fusion methods, and data mining techniques.

❑ **Chapter 6:** summarizes the thesis by highlighting its main contributions. It also discusses some concluding remarks and the limitations of the proposed models. Directions for future works are also outlined.

Chapter 2

Trust and Reputation: a Literature Review

2.1 Existing Surveys

Computational trust has gained the attention of a wide range of research in the area of reputation and rating systems [11]. The notion of trust has been defined differently by researchers influenced by their diverse background areas and applications. On one hand, the most common definition of trust can be stated as being an entity's opinion regarding another entity's behavior. The importance of this opinion appears in governing decision making processes such as deciding on which entity (agent/service provider) to be selected. Reputation, on the other hand, is the community's opinion about the standing of an entity [12]. This section far from being systematic, aims to discuss the current literature's most representative models. It also intends to fit our research within the existing academic studies. The next few paragraphs are dedicated to examining the existing literature reviews of trust and reputation systems [13], [14], [12], [5], [15], [16].

Jøsang et al. presented an in-depth literature review of trust and reputation systems in various web communities rather than in specific ones [13]. After disambiguating the notion of trust, this survey identified various reasons that motivate research in this area. The authors also discussed and provided multiple dimensions to differentiate between trust and reputation system. For instance, they used Grandison and Sloman's classification (2000) to divide trust into five classes; provision, access, delegation, identity, and context. The reputation systems were classified as either centralized or decentralized. This classification depicts how ratings and reputation scores are distributed among agents in reputation systems. The reputation computation adds another dimension that further divides reputation systems into simple summation, Bayesian, discrete trust, belief, fuzzy and flow systems. [13] also lists a number of problems that current trust and reputation systems face along with solutions to each of these problems. The prominent problem is uncovering and dealing with biased positive and negative ratings that unfairly affect the trust and reputation values of an entity.

Artz et al. reviewed scholarly work related to trust in computer science and semantic web [14]. The authors of this survey classified trust into four main areas including policy-based trust, reputation-based trust, general models of trust, and trust in information resources. In our research, we treat the reputation-based trust to evaluate the future behavior of an agent-based web service based on the quality of its past performance. They further divided the reputation-based trust into four subcategories. Namely, decentralization and referral trust, trust metrics in a web of trust, trust in P2P networks and grids, application specific reputation.

Wang and Vassileva investigated trust and reputation systems for the purpose of drawing potential research directions for similar systems in web services [12]. The main contribution of this work is a three-dimensional classification of existing trust and reputation systems. The first dimension captures the reputation management aspect of such systems and distinguish between "centralized" and "decentralized" systems. The second dimension depicts the reputation modeling target. On one hand, trust and reputation systems such as eBay can be viewed as "person/agent" if the target is a person or an agent. On the other hand, they can be viewed as "resource" systems such as Epinions if the reputation is associated with products or services. The third and last dimension encapsulates the basis or source of the reputation of a person or service. For instance, if this source is the opinions of specific group of people, the system is referred to as "personalized". However, if the reputation is the result of the opinions of all entities, then the system is classified as "global". According to the authors in [12], the majority of trust and reputation systems for web services are centralized, resources-based and personalized. Indeed, these systems make use of a central registry that is responsible for collecting and storing the QoS data. This opens new research directions for investigating decentralized, agent-based and global trust and reputation systems for web services.

In [5], the authors classified the different approaches to trust in multi-agent systems as individual-level trust and system-level trust. The individual-level trust enables agents to reason about their level of trust in other agents using different models including socio-cognitive, reputation, and evolutionary and learning models. The system-level trust mechanisms such as trustworthy interaction, reputation, and distributed security mechanisms, ensure that an agent can trust other agents' actions. Our study builds upon and extends the trust learning models using machine learning techniques.

Sabater and Sierra classified trust and reputation models with special attention on computational models. This classification's highest level consider the following dimensions [15]. The conceptual model, information sources, visibility types, model's granularity, agent behavior assumptions, type of exchanged information, and trust/reputation reliability measure. We will elaborate on these dimensions in the trust and reputation systems categorization section.

Pinyol and Sabater-Mir reviewed in [16] the classification schemes of trust and reputation systems in three previous works; [15], [17] and the European project eRep *(2006)*. The authors also contributed with an additional four-dimensional classification: trust, cognitive, procedural and generality dimensions.

2.2 Probability-Based Trust And Reputation Systems

This section overviews studies in which trust is computed using statistical models. Hang and Singh modeled the quality of a service using a Binomial distribution [18]. The authors of this study proposed two service selection approaches based on Bayesian networks and Beta-mixture model. These approaches capture the responsibilities of the constituent services in the overall quality of a composite service. They used various composition functions including switch, sum, product, min, and max. The proposed approaches can only handle one quality metric at a time. It is rare that agents would be interested in only one quality metric when willing to interact with other agents. Although it is possible to treat each quality metric separately, this would be time consuming. Also, the Binomial distribution represents the quality of a service as good or bad which fails to depict its degree of goodness or badness.

Wang and Vassileva proposed a Bayesian network approach representing and combining different faces of trust In [19]. The focus of this work is peer-to-peer networks. for instance, the Bayesian network root represent the overall quality of a file provider. The root's children and network's leaves represent the trust in various capabilities of the file provider. These may include the download speed, the file quality and the file type. It would be interesting to scale this to handle other types of large scale distributed systems.

Rettinger et al. aimed for learning context-specific trust using statistical relational learning [20]. The authors used the collaborative filtering to improve the performance of trust learning. The proposed model in this work does not consider composite services. According to [20], a large number of the current models of trust are based on a holistic view of the behavior of the trustee. The contextual information of the subject entities and their environment is not deeply addressed yet. Trust is based on various quality metrics which constitute quantifiable measures of web services reliability, throughput, availability, latency, and response time. Most of existing trust models either deal with each quality metric separately or fail to capture the responsibility of each metric in the overall performance of an agent. An agent, for example, might be reliable, but disappoints its truster because of its low availability.

Teacy et al. used probability theory to compute trust based on past interactions between agents [21]. Using such approach, each agent stores the outcomes of its interactions with other agents. These outcomes have the values 1 in case of a successful interaction (contract fulfilled) and 0 otherwise. A beta distribution is then used to represent the probability of having a good outcome based on previous ones. Two main measures are then computed; the level of trust in an agent and a confidence value associated with it. Moreover, this system considers the reputation of an agent in cases where the confidence values are low. It also employs an exogenous solution to filter inaccurate reputation.

In [22], Li and Wang proposed a subjective probability based deductive approach to compute trust for composite web services. The composition of services is modeled according to six types of invocations between them; sequential, parallel, probabilistic, circular, synchronous, and asynchronous. A direct invocation from one service to another causes trust dependency which is modeled by a conditional probability. An algorithm, SELECTIVE, has also been proposed to compute the subjective trustworthiness of a composite service. The travel example was used to represent a composite service for which the ratings were taken from Epinions. Neither the reputation of service providers nor the inaccurate ratings were considered in this work.

2.3 Other Trust and Reputation Systems

Zacharia et al. proposed in [23] two collaborative filtering methods to compute users' reputation ratings; SPORAS and HISTOS for loosely and highly connected online communities, respectively. SPORAS combined direct and witness ratings to compute the reputation of an entity. It dealt with new entities by assigning them low reputations which are then built up after interactions with other entities. To update the reputation of an entity, SPORAS considered the last rating only. When SPORAS is considered as a global reputation system, HISTOS is a personalized system that accounts for pairwise ratings. These ratings are propagated through the edges of a directed graph connecting entities that had interacted with each others. A breadth first search and a recursive step are used to to compute a personalized reputation score of an entity. Both systems were tested using four simulations presenting different scenarios including a new entity joining the system and a collusion by two entities.

A QoS-based service selection with trust and reputation reinforcement was proposed in [24]. The prediction of a web service future quality is accomplished using a real-valued time series forecasting technique. The basis of the approach developed in this study is the quality conformance values of a service dispensed for various quality attributes. This is computed as the ratio of the difference between the observed quality and the one claimed by the service provider to the latter. The conformance valued are also broadcasted among users as reputation reports about each web service. The dishonest values were filtered out ”by combining a trust-distrust propagation approach with a data-mining method”.

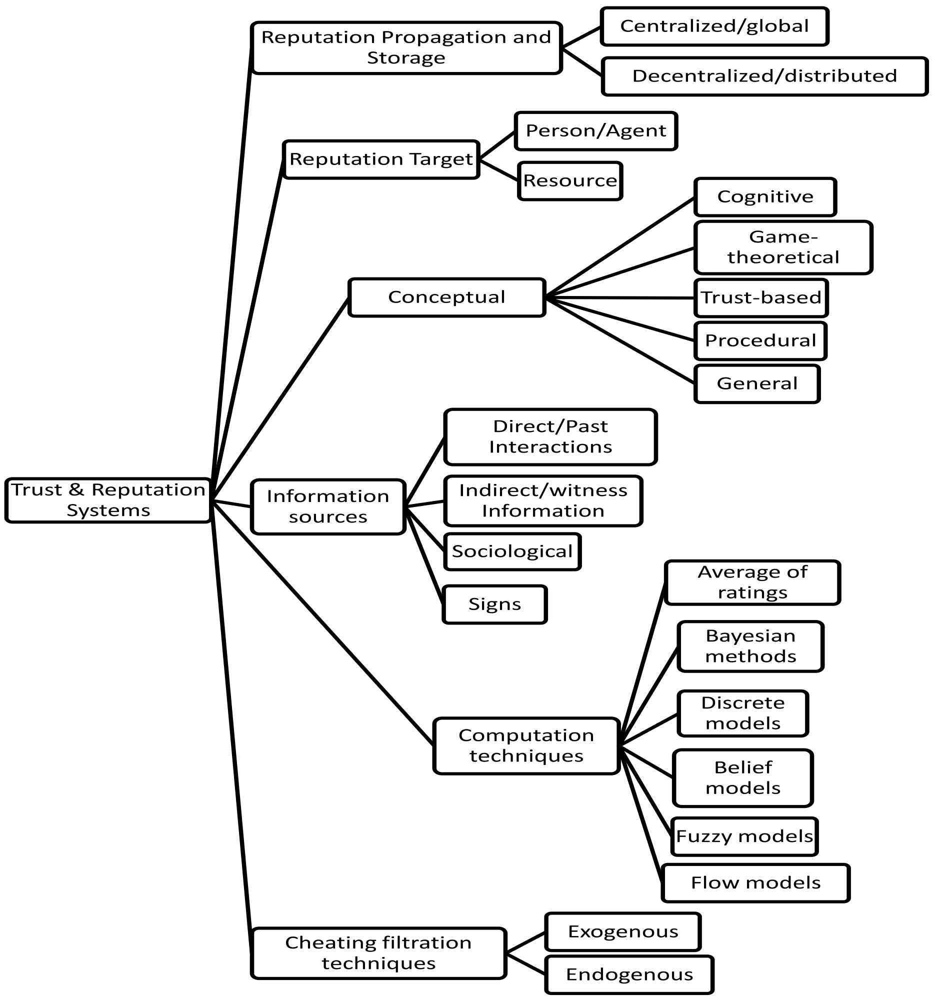


Figure 2.1: Trust and Reputation Systems Taxonomy

2.4 Trust and reputation systems categorization

This section amalgamates the trust and reputation systems dimensions extracted from the various studies reviewed in the literature review section. Figure 2.1 summarizes these dimensions in a hierarchical taxonomy. The following sections provide descriptions for each of the categories illustrated in this taxonomy.

2.4.1 Reputation propagation and storage

The current trust and reputation systems can be divided by the means the reputations of the system's entities are propagated and stored. For instance, we differentiate between *Centralized* and *Decentralized* systems [12], [13]. These two groups match the two visibility types reported in [15], namely *Global* and *Personal*, respectively. They also overlap with the *Centralized* and *Distributed* reputation network architectures reported in [13].

Centralized Centralized systems follow the repository architectural style. In other words, these systems employ a central repository that is responsible for computing and storing the ratings of their entities. The ratings are then globally visible and accessible by all the entities presently in the system.

Decentralized Decentralized systems are designed according to the distributed architecture. In other words, there is no central repositories for all entities ratings. Instead, each entity stores a model of the ratings of each other entities it interacts with. Afterwards, an entity can request the ratings of other entities with which it had no direct interaction.

2.4.2 Reputation target

This dimension focuses on describing the reputation target that can be classified into two main classes [12]; *Person/Agent* and *Resource*.

Person/Agent The most common reputation system that reports the ratings associated to a person is Ebay. When the reputation target is a person or an agent acting on behalf of a person, the system is referred to as Person/Agent.

Resource Systems in which the reputation target a product or a service are considered Resource systems. A popular example of a Resource system is Epinions.

2.4.3 Conceptual modeling

Various studies have considered the conceptual dimension to classify trust and reputation systems. Sabater and Sierra divided the conceptual modeling category into two classes *Cognitive* and *Game-theoretical* [15].

Cognitive Cognitive approaches are based on the entities' mental states that drive to trust other entities or assign them reputation scores. The mental state of an entity depend on the degree of beliefs related to other entities.

Game-theoretical Game-theoretical approaches apply a more practical method to compute trust and reputation scores when compared to cognitive ones. This method relies on the probability an entity X has assigned to entity Y that the latter successfully fulfills certain functions that the former's goal depends on. Indeed this probability is based on past interactions between the two entities X and Y.

2.4.4 Information sources

This is the dimension that can capture any trust and reputation system under at least one of its classes. The majority of these systems compute their entities' trust values based on *Direct past interactions* between them. However, considering *Indirect/Witness information* about an entity has lately gained the attention of many researchers [21], [25], [26], [27], etc... [15] appended two new classes to the existing information sources: *Sociological* and *Signs*.

Direct past interactions In almost all trust and reputation systems, entities keep a model of other entities they interact with. This model consists of the status or quality of the direct past interactions between the two entities. This is a natural mapping of the foundation of human trust to the concept of computational trust.

Indirect/Witness information A common real life scenario might be the following: an entity is willing to interact with another entity with which it had no previous experience. In this case it is hard for the former entity to decide if the latter is trustworthy or not. Therefore, the indirect or witness information would be vital to assist the truster entity in such scenarios. This information consist of the reputation of the trustee entity computed by other entities who had direct interactions with it.

Sociological This is a fairly new information source for trust and reputation systems. This can be described as a projection of human social relations that have great impact on reputation values. For instance, a group of people who have collaborative relationships might report biased reputations about each others. This can also be the case in case of collaborative agents or web services. Another example can be about two web services providing similar functions. These might then report unfair low reputations one about the other. A social analysis of agents and web services is a new challenge that assists the computation of trust and reputation.

Signs This class is called *prejudice* in [15]. We substituted prejudice by signs to amputate the negativity of the former term. Signs can be attributed as identifiers of a group of people. For instance, signs of harmless drugs and biological products can be the approval of the Food and Drug Administration (FDA).

2.4.5 Computation techniques

In addition to the different information sources, trust and reputation can be computed using various methods. These include the summation or *Average of ratings*, *Bayesian*, *Discrete*, *Belief*, *Fuzzy* and *Flow* methods [13].

Summation/Average of ratings The authors in [13] accounted for three forms of computing reputation values. The first and simplest way is by calculating the difference between the sum of positive and negative ratings. This model is implemented in the eBay reputation system. The second form would be by computing the average of all ratings given to an entity. Epinions is an example of a reputation system that uses this form of reputation computing. The third and last form add weights to the ratings before computing the average.

Bayesian methods Bayesian systems are based on the statistical updating method that combine the a priori and new values to compute the updated a posteriori score. Most of the Bayesian-based reputation systems represent the reputation score by a Beta distribution parameters (α, β) [18]. α and β represent the amounts of positive and negative ratings respectively. Other models also considered multi-valued scores which then can be represented by a Dirichlet distribution parameters $\alpha_1, \ldots, \alpha_N$. N is the number of categories a score can belong to. For instance, a system with the three categories score model *High*, *Medium*, *Low* will introduce a Dirichlet distribution with the parameters $\alpha_1, \alpha_2, \alpha_3$. α_1, α_2, and α_3 represent the amounts of *High*, *Medium*, and *Low* ratings respectively. Then, the trust values can be computed as the expected values of the Beta or Dirichlet distributions given respectively by $\frac{\alpha}{\alpha+\beta}$ and $\frac{\alpha}{\sum_{i=1}^{N} \alpha_i}$.

Discrete models Discrete trust models such as in [25] are those that “rate performance in the form of discrete verbal statements, than continuous measure” [13].

Belief models Belief models refer to the system that apply belief theory to compute trust or reputation scores. This theory is based on the belief measures of the behavior of an entity in past interactions.

Fuzzy models The intuition behind fuzzy models is the approximate reasoning rather than the exact or fixed. Following this intuition, a reputation or trust score is represented by an approximate value that depicts the degree to which a system is trustworthy or not.

Flow models [13] defined flowed models by the systems that “compute trust or reputation by transitive iteration through looped or arbitrarily long chains”.

2.4.6 Cheating filtration

Most of trust and reputations systems are based on ratings of entities provided by other entities. These ratings might be influenced by social relations or rivalries among these entities. Therefore, one can surely assume that a number of the provided ratings are not just. Many cheating filtration mechanisms have been proposed by the community. Jøsang et al. classified these mechanisms as *Endogenous* and *Exogenous* [13]:

Endogenous Endogenous methods use the ratings statistical properties to analyze the ratings values and discount unfair ones. The authors in [28], [29], and [30] proposed statistical filtering techniques that fit under this category.

Exogenous Exogenous methods cope with unfair ratings by using the rater's global reputation. This technique is applicable under the assumption that raters with low reputations tend to provide unfair ratings. Examples of exogenous methods are available in [31] and [27].

2.5 Discussion

The trust and reputation systems literature seems to be facing the following research gaps:

1. **Indirect/Witness information:**
   In addition to past direct interactions among web services, witness information is of comparable importance especially in two main scenarios:
   (a) A service consumer needs to evaluate the trustworthiness of a new web service (no past interactions are available). This is a common scenario in open and distributed systems.
   (b) A service consumer needs to evaluate the trustworthiness of a web service with which only few interactions exist. In other words, the former's model trust model of the latter is based on the outcomes of few past interactions.
2. **Confidence in trust scores:**
   A trust score that dates a year back is not as reliable as another score that was recently updated. Moreover, a reputation score that has been assigned by an entity with high reputation is more accurate than a score assigned by another one with low reputation. This raises the need to assign a confidence value to each trust and reputation score. For instance, a web

service with a trust score 0.5 and confidence value 0.8 is better than another web service with trust score 0.8 and confidence value 0.3.

3. **Unfair ratings:**
Rivalries and alliances are homely to open systems whose entities might provide similar goods or services. Therefore, the propagated ratings between the systems' entities include inequitable ones. Possible scenarios that might incur unfair ratings include:

   (a) A web service WS1 requests a referral from WS2 about its biggest competitior WS3.

   (b) Two web services WS1 and WS2 have formed and alliance to provide a group of complementary functions. WS2 global reputation is not high due to certain flaws in its behavior. When WS3 requests a referral from WS1 about WS2, the former sends a high but unjust reputation score.

4. **Service providers trustworthiness:**
An additional information resource that helps directing any trust-based decision is the trustworthiness of service providers. For instance, a web service WS1 is willing to interact with a new web service WS2. WS1 has never interacted with WS2 and neither did other web services currently available in the system. Suppose WS1 has already interacted with WS3, a web service supplied by the same service provider of WS2. Then, the trustworthiness of the service provider of both WS2 and WS3 can be considered an initial indicator of the trustworthiness of WS2.

Chapter 3

QoS-Based Probabilistic Trust Models

The service selection problem has gained the attention of researchers from the computational trust community, especially with abundance of functionally similar services. Following our definition of trust, designing a model that evaluates the trustworthiness of web services presents a challenging yet motivating problem. Accoriding to [32], this is a defying task given the lack of intelligence in the communication protocol among web services.

This chapter is devoted to the description of a QoS-based trust model that is based on the statistical modeling of the QoS ratings of web services interactions. We propose to classify the quality of web services into multiple classes (more than 2) based on different sets of QoS metrics according to the consumers' preferences. This approach allows consumers to maintain a trust model for each service provider they interact with. As such, the trust model assists consumers in selecting, among a plethora of similar services, the most trustworthy one. We thus associate the trust in a service to its performance denoted by QoS ratings instigated by the amalgamation of various QoS metrics. Since the quality of a service is contingent, which renders its trustworthiness uncertain, we adopt a probabilistic approach for the prediction of the quality of a service based on the evaluation of past experiences (ratings) of each of its consumers. We represent the QoS ratings of services using different statistical distributions, namely, multinomial-Dirichlet (*MDD*) , multinomial-generalized Dirichlet (*MGDD*), and multinomial-Beta-Liouville (*MBLD*). These representations enable the estimation of the probabilities of each web service to belong to different quality classes. For this purpose, we use the Bayesian inference method to estimate the parameters of the aforementioned distributions, which presents a multidimensional probabilistic embodiment of the quality of the corresponding web services.

In [14], the authors classified trust into four main areas including policy-based trust, reputation-based trust, general models of trust, and trust in information resources. Following their definitions of each type of trust, our approach fits the reputation-based trust definition which consists of evaluating the future behavior of web services based on the quality of its past performance. The reputation-based trust may also include referrals from peers which is discussed in chapter 5.

The main contributions presented in this chapter can be summarized as follows:

1. We propose to compute QoS ratings based on multiple quality metrics such as the following performance and dependability quality metrics: response time, throughput, availability and reliability. Subsequently, we model the QoS ratings of web services by various multi-dimensional statistical distributions which offer more descriptive representation than the Beta Binomial.

2. We estimate the future quality of web services by learning the parameters of their QoS distributions using the Bayesian inference method based on their past behavior. Using this method, the ratings of a service and the prior knowledge about its behavior are combined to compute the posterior probabilities.

3. In addition to the multidimensional representation of the QoS of a web service, we propose a QoS-based trust function to compute one-dimensional trust scores. These scores can be used to compare between web services from various systems with disparate number of QoS ratings.

3.1 QoS Modeling Via Composite Distributions

The following sections overview a subset of the QoS metrics that may be considered in the proposed trust model. They also discuss the details and advatanges of modeling the QoS ratings using the different distributions; *MDD*, *MGDB*, and *MBLD*.

3.1.1 QoS Metrics Overview

The QoS of agent-based web services has gained the attention of wide spectrum of research especially in the contexts of service discovery, selection and composition. The computational trust notion in agents was particularly treated in the pursuit of solving the preceding tasks. Given a pool of agents that fulfill the functional requirements of a consumer, the latter chooses the one that is estimated to provide the best quality. As such, the trustworthiness of a service is positively correlated to the aggregation of the outcomes of its QoS metrics. However, aggregating the measures obtained from monitoring the different QoS metrics is not trivial. This is due to (1) not all consumers are interested in monitoring and evaluating the same metrics. (2) given the values of a set of QoS metrics, there is no standardized approach to aggregate them.The list below defines some examples of QoS metrics [33]:

- Response time (*RT*): This is a measure of the time spent between sending a request and receiving the last byte of the response. RT is inversely proportional to the quality of a service. In other words, a lower RT value implies a better service.
- Throughput (*T*): The throughput of a service is the number of requests it can handle per time unit. This measure is directly proportional to the QoS.
- Availability (*A*): The availability measure denotes the probability of the service to be up and ready to answer its users requests. Availability is directly proportional to the overall quality of a service.
- Reliability (*R*): Reliability may refer to the ratio of the number of valid responses to the total number of responses an agent-based service provides.
- Latency (*LA*): This refers to the time a server needs to process a request.
- Cost (*C*): The cost of a service measures the usage of resources during its execution time and might refer to the service fees. This metric is inversely proportional to the general quality of a service. A higher cost will compromise the quality of the corresponding service.

After each interaction with a service provider, the above metrics are measured and the service provider is then assigned a QoS score given by:

$$QoS\_Score = \frac{RT\ W_{RT}}{Max(RT)} + \frac{T\ W_T}{Max(T)} + AW_A + RW_R + CW_C$$

,

where W_M is the weight of the quality metric M and $\sum_{M\in\{RT,T,A,R\}} W_M = 1$. $Max(RT)$ and $Max(T)$ are used to normalize the values of RT and T, respectively. Afterwards, a QoS rating (class) is attributed to the current interaction with the service provider. The following is one of many possible ways to partition the QoS ratings:

$$QoS\_Rating = \begin{cases} 1 & \text{if } 0.7 < QoS\_Score < 1 \\ 2 & \text{if } 0.5 < QoS\_Score < 0.7 \\ 3 & \text{if } 0.3 < QoS\_Score < 0.5 \\ 4 & \text{otherwise} \end{cases}$$

3.1.2 MDD, MGDD and MBLD for QoS Modeling

To compute the trustworthiness of a web service, we propose to model its quality over a period of time using a *MDD*. This distribution helps positioning the quality of a service on a spectrum that captures its level of goodness or badness. Given this distribution, we learn the probabilities of the QoS to belong to various quality classes (note that we use the terms quality class and quality rating interchangeably). To assist the service selection task, we will compute the trustworthiness of a web service based on its past direct interactions with other web services. We will defer the use of indirect observations based on peers recommendations to Chapter 4. The qualities of these past interactions are represented as vectors of quality rankings counts. Each of these vectors has a size equal to the number of quality classes. More formally, let $\mathcal{X} = [\vec{X}_1, \vec{X}_2, ..., \vec{X}_N]$ represents a set of N K-dimensional vectors of counts where $\vec{X}_i = [X_{i1}, X_{i2}, ..., X_{iK}]$ represents the i^{th} set of quality rankings and K is the number of classes the quality of a service can belong to. X_{ik} is the number of times the quality of a web service was ranked as k. We assume that $\vec{X}_i$ is generated from a multinomial distribution with parameters $\vec{\theta} = (\theta_1, \ldots, \theta_{K+1})$ given by:

$$p(\vec{X}_i|\vec{\theta}) = \frac{|\vec{X}_i|!}{\prod_{k=1}^{K} X_{ik}!} \prod_{k=1}^{K} \theta_k^{X_{ik}} \tag{3.1}$$

where $|\vec{X}_i| = \sum_{k=1}^{K} X_{ik}$. A common method to estimate the parameters of the multinomial is the Maximum Likelihood Estimation (MLE) which gives the following estimates:

$$\hat{\theta}_k = \frac{X_{ik}}{|\vec{X}_i|} \tag{3.2}$$

For example, suppose that an agent A (truster) wants to estimate the trustworthiness of another agent B (trustee) and we have four quality classes. Therefore, the past experiences of agent A with agent B will be represented by 4-dimensional vectors such as the following: $\vec{X} = [8, 6, 3, 1]$. Then, the MLE estimates of the trustworthiness of agent B will be according to equation (3.2) the probabilities vector $\vec{\theta} = [\frac{8}{18}, \frac{6}{18}, \frac{3}{18}, \frac{1}{18}]$. Each element of this vector represents the probability of agent B providing one of the four qualities. Although the MLE method tends to give more or less natural estimates, it gives poor estimates in the case of low or rare frequencies. For instance, suppose agent A had eight interactions with agent B all of which were classified as first-class quality. That is, the quality vector is $\vec{X} = [8, 0, 0, 0]$. Then, the parameters vector $\vec{\theta}$ will be $[1, 0, 0, 0]$ which gives a poor estimate by setting the distribution to zero.

To deal with the above problem, we can apply a Bayesian inference-based method to improve these estimates. This approach consists of selecting a prior distribution for the $vec\theta$ parameters. It is a common practice to assign a Dirichlet prior to the parameter vector of a multinomial distribution since it is a conjugate prior [34]. In other words, both the prior and the posterior are Dirichlet. The Dirichlet distribution is the multivariate extension of the Beta distribution. Therefore, this model states that the quality of an agent is drawn from a multinomial that represents the counts of the quality being in each of the classes. The prior is a Dirichlet which parameters work as pseudo-counts. The resulting distribution is known as *MDD* [35] given by:

$$p(\vec{X}_i|\theta) = \frac{\Gamma(\sum_{k=1}^{K+1} \alpha_k)\Gamma(\sum_{k=1}^{K+1} X_{ik} + 1)}{\Gamma(\sum_{k=1}^{K+1} X_{ik} + \sum_{k=1}^{K+1} \alpha_k)} \prod_{k=1}^{K+1} \frac{\Gamma(X_{ik} + \alpha_k)}{\Gamma(\alpha_k)\Gamma(X_{ik} + 1)} \tag{3.3}$$

The posterior is given by:

$$\begin{aligned} p(\vec{\theta}|\vec{X}_i, \alpha_1, ..., \alpha_{K+1}) &= \frac{p(\vec{X}_i, \vec{\theta}|\alpha_1, ..., \alpha_{K+1})}{p(\vec{X}_i|\alpha_1, ..., \alpha_{K+1})} \\ &= \frac{\Gamma(\sum_{k=1}^{K+1} X_{ik} + \sum_{k=1}^{K+1} \alpha_k)}{\prod_{k=1}^{K+1} \alpha_k + X_{ik}} \prod_{k=1}^{K+1} \theta_k^{\alpha_k + X_{ik} - 1} \end{aligned}$$

which is a Dirichlet with parameters $(\alpha_1 + X_{ik}, ..., \alpha_{K+1} + X_{iK+1})$, where $(\alpha_1, ..., \alpha_{K+1})$ are the parameters of a Dirichlet. Using this prior information, $\hat{\theta}_k$ becomes:

$$\hat{\theta}_k = \frac{X_{ik} + \alpha_k}{\sum_{k=1}^{K+1} \alpha_k + \sum_{k=1}^{K+1} X_{ik}} \tag{3.4}$$

MDD is the multidimensional case of the Beta binomial distribution widely used in many applications including trust estimation [18]. Moreover, *MDD* has one extra degree of freedom when compared to the multinomial distribution, since its parameters are not constrained to sum up to one, which makes it more practical [36]. Further details about the *MDD* properties and the development of its moments can be found in [37].

Suppose we have the same vector, $\vec{X} = [8, 0, 0, 0]$, which we have previously handled poorly using MLE. By introducing a Dirichlet prior with the parameters $\vec{\alpha} = [1, 1, 1, 1]$, the smoothed estimates evaluated using equation (3.4) are:

$$\begin{aligned} \vec{\theta} &= [\frac{8+1}{4+8}, \frac{0+1}{4+8}, \frac{0+1}{4+8}, \frac{0+1}{4+8}] \\ &= [\frac{9}{12} = 0.75, \frac{1}{12} \approx 0.083, \frac{1}{12} \approx 0.083, \frac{1}{12} \approx 0.083] \end{aligned}$$

The hyperparameters $\vec{\alpha}$ can be thought of as being hidden quantities added in order to represent our confidence about the estimates. They are also beneficial to moderate the extreme estimates given by the MLE of the multinomial distribution. In spite of the flexibility of the Dirichlet distribution and the fact that it is conjugate to the multinomial, it still suffers from various restrictions. For instance, the Dirichlet has a very restrictive negative covariance matrix defined by:

$$Cov(\theta_i, \theta_j) = -\frac{\alpha_i \alpha_j}{(\sum_{i=1}^{K+1} \alpha_i)^2 (\sum_{i=1}^{K+1} \alpha_i + 1)} \tag{3.5}$$

Indeed, this covariance presumes that any two random variables in $\vec{\theta}$ are negatively correlated which is not always the case [38]. Another restriction of the Dirichlet distribution is that variables with the same mean must have the same variance which is proven in [39]. These restrictions can be avoided if we use the generalized Dirichlet distribution (GDD) as a prior instead of the Dirichlet. The GDD with parameter vector $\vec{\alpha} = (\alpha_1, \beta_2, ..., \alpha_K, \beta_K)$ is defined by [40]:

$$p(\theta_1, ..., \theta_d) = \prod_{k=1}^{K} \frac{\Gamma(\alpha_k + \beta_k)}{\Gamma(\alpha_k) + \Gamma(\beta_k)} \theta_k^{\alpha_k - 1} (1 - \sum_{j=1}^{k} \theta_j)^{\gamma_k}, \tag{3.6}$$

given $\sum_{k=1}^{K} \theta_k < 1$ and $0 < \theta_k < 1$ for $k = 1, ..., K$, where $\alpha_k > 0$, $\beta_k > 0$, $\gamma_k = \beta_k - \alpha_{k+1} - \beta_{k+1}$ for $k = 1, ..., K - 1$, and $\gamma_K = \beta_K - 1$. The GDD has $K - 1$ more parameters than the Dirichlet which gives it K degrees of freedom when used to construct a prior. It also has a more general covariance and doesn't restrict variables with the same mean to have the same variance [35]. Additionally, it is conjugate to the multinomial distribution. The mean, variance, and covariance of the GDD are available in [40]. Thus, the marginal distribution of $\vec{X}_i$ obtained from integrating the joint distribution of $\vec{X}$ and $\vec{\theta}$ over the latter is:

$$\begin{aligned} p(\vec{X}_i|\vec{\alpha}) &= \int_{\vec{\theta}} p(\vec{X}_i, \vec{\theta}|\vec{\alpha}) d\vec{\theta} \\ &= \frac{\Gamma((\sum_{k=1}^{K+1} X_{ik}) + 1)}{\prod_{k=1}^{K+1} \Gamma(X_{ik} + 1)} \prod_{k=1}^{K} \frac{\Gamma(\alpha_k + \beta_k)}{\Gamma(\alpha_k) + \Gamma(\beta_k)} \prod_{k=1}^{K} \frac{\Gamma(\alpha'_k) + \Gamma(\beta'_k)}{\Gamma(\alpha'_k + \beta'_k)} \end{aligned}$$

The above density is called *MGDD*. Therefore, using the generalized Dirichlet as a prior to the multinomial results with the following estimates:

$$\hat{\theta}_k = \frac{\alpha_k + X_{ik}}{\alpha_k + \beta_k + n_{ik}} \prod_{l=1}^{k-1} \frac{\beta_l + n_{il+1}}{\alpha_l + \beta_l + n_{il}}, \tag{3.7}$$

where $n_{ik} = X_{ik} + X_{ik+1} + \cdots + X_{iK+1}$.

To illustrate the estimation using *MGDD*, suppose we have the same vector $\vec{X} = [8,0,0,0]$ handled above with MLE and *MDD*. By introducing a generalized Dirichlet prior in dimension $K = 3$ with the parameters $\vec{\alpha} = [1,1,1,1,1,1]$, the new estimates computed using equation (3.7) become:

$$\begin{aligned}\vec{\theta} &= [\frac{1+8}{2+8}, \frac{1+0}{2}\frac{1}{2+8}, \frac{1+0}{2}(\frac{1}{10}\frac{1}{2}), 1-(\frac{9}{10}+\frac{1}{20}+\frac{1}{40})] \\ &= [\frac{9}{10} = 0.1, \frac{1}{20} = 0.05, \frac{1}{40} = 0.025, \frac{1}{40} = 0.025]\end{aligned}$$

Another possible choice of prior to the multinomial belongs to the Liouville family of distributions of second kind. In dimension K, the Liouville distribution, with positive parameters $(\alpha_1, \ldots, \alpha_K)$ and generating density $f(\cdot)$ with parameters ξ, is defined by:

$$p(\vec{\theta}|\alpha_1, \ldots, \alpha_K, \xi) = f(u|\xi)\frac{\Gamma(\sum_{k=1}^{K}\alpha_k)}{u^{\sum_{k=1}^{K}\alpha_k - 1}}\prod_{k=1}^{K}\frac{\theta_k^{\alpha_k - 1}}{\Gamma(\alpha_k)}$$

where $u = \sum_{k=1}^{K}\theta_K < 1$ and $\theta_k > 0$, $k = 1, \ldots, K$. One common and convenient choice of a generating density for u is the Beta distribution with parameters α and β which is defined as:

$$f(u|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)+\Gamma(\beta)}u^{\alpha-1}(1-\alpha)^{\beta-1}$$

The shapes in the Beta distribution are variable enough to allow for an approximation of almost any arbitrary distribution [41]. The resulting distribution is called Beta-Liouville (BLD) which has a covariance structure that can be either positive or negative, unlike the strictly negative covariance matrix of Dirichlet distribution. Like both Dirichlet and generalized Dirichlet distributions, the BLD is also conjugate to the multinomial distribution. The marginal distribution of $\vec{X}_i$ obtained from integrating the joint distribution of $\vec{X}$ and $\vec{\theta}$ over the latter is:

$$\begin{aligned}p(\vec{X}_i|\vec{\alpha}) &= \int_{\vec{\theta}} p(\vec{X}_i, \vec{\theta}|\vec{\alpha})d\vec{\theta} \\ &= \frac{\Gamma((\sum_{k=1}^{K+1}X_{ik})+1)}{\prod_{k=1}^{K+1}\Gamma(X_{ik}+1)}\frac{\Gamma(\sum_{k}^{K+1}\alpha_k)\Gamma(\alpha+\beta)\Gamma(\alpha')\Gamma(\beta')\prod_{k=1}^{K}\Gamma(\alpha'_k)}{\Gamma(\sum_{k}^{K+1}\alpha'_k)\Gamma(\alpha'+\beta')\Gamma(\alpha)\Gamma(\beta)\prod_{k=1}^{K}\Gamma(\alpha_k)}\end{aligned}$$

where $\alpha'_k = \alpha_k + X_{ik}$, $\alpha' = \alpha + \sum_{k=1}^{K} X_{ik}$ and $\beta' = \beta + X_{iK+1}$. The above density is called *MBLD*. When using a BLD prior to the multinomial distribution, we obtain:

$$\hat{\theta}_k = \frac{\alpha + \sum_{k=1}^{K} X_{ik}}{\alpha + \sum_{k=1}^{K} X_{ik} + \beta + X_{iK+1}} \frac{\alpha_k + X_{ik}}{\sum_{k=1}^{K} (\alpha_k + X_{ik})} \tag{3.8}$$

The new estimates using equation (3.8) for the same example discussed above become:

$$\vec{\theta} = [\frac{81}{110} \approx 0.74, \frac{9}{110} \approx 0.08, \frac{9}{110} \approx 0.08, \frac{1}{110} \approx 0.01]$$

3.2 Correlations-Based Trust Model

3.2.1 Correlation Analysis of Two Real Datasets

This section aims to shed the light on the correlations that exist among various QoS metrics. In order to support the motivation for our work, we collected the meta-data of 9392 web services available from *programmableweb* through their API [1]. The attributes captured in this data include various meta-data information as well as the values of multiple QoS parameters of their web services database. We use the terms parameters and metrics interchangeably throughout this thesis.

ProgrammableWeb: We extracted 39 attributes of meta-data and various QoS metrics of 9392 web services. These attributes include the *ID*, *Name*, *Description*, *Type*, *Download Counts*, and *Number of Comments* for each of these web services. Table 3.1 lists all the extracted attributes:

ID	Name	Package	Author
Description	Type	Downloads	UseCount
DateModified	Rating	RemoteFeed	NumComments
Tags	Category	Protocols	ServiceEndpoint
Version	WSDL	DataFormats	ApiGroups
Example	ClientInstall	Authentication	SSL
Readonly	VendorApiKits	CommunityApiKits	Blog
Forum	Support	AccountReq	Commercial
Provider	ManagedBy	NonCommercial	DataLicensing
Fees	Limits	Company	

Table 3.1: Attributes extracted from the ProgrammableWeb API

[1]http://www.apihub.com/programmable-web/api/programmable-web-api

We select from this data the values of two critical parameters: the security measure embedded within the agent-based service and the fees for interacting with it. We aim to identify the relationship that exists between these two parameters. Since the data is incomplete, we wrote a script to consider the services that have values for both metrics and convert these values to binary. As such, a "none" or "free" fees values are converted to 0, and all values that contain the dollar sign are converted to 1. The true or false security values are mapped to their corresponding binary values (1 and 0, respectively). A snapshot of the raw values of these two metrics is given in Table 3.2 that shows that some data is missing in the original dataset. Only rows that had both information *SSL* and *Fees* were considered in our analysis.

SSL	**Fees**
	http://www.360businesstool.com/priser/
No	1-10000 credits/$.05sms 10001-25000
Yes	USA: $4.99-8.99$ /mo -or- $5.00-20.00$ for 400-2500 texts
	Contact 3TIER for current rates
No	No
Yes	0.055/sms 0.02/min to land lines
	Free
Yes	Free

Table 3.2: Sample of *SSL* and *Fees* raw values

Given the processed data, we calculate the correlation between both attributes by first computing its covariance matrix. This matrix is usually denoted by Σ with each entry, Σ_{ij}, representing the covariance between the QoS metrics i and j:

$$\Sigma_{ij} = E[(Q_i - \mu_i)(Q_j - \mu_j)], \tag{3.9}$$

where $\mu_i = E[Q_i]$ is the expectation of the value of the i_{th} quality metric, Q_i. The covariance matrix of *Fees* and *SSL* parameters is:

$$\Sigma_{SSL,Fees} = \begin{pmatrix} 0.2461 & 0.0590 \\ 0.0590 & 0.2196 \end{pmatrix}.$$

Afterwards, we compute the corresponding matrix of correlation coefficients, ρ given by:

$$\rho_{ij} = \frac{\Sigma_{ij}}{\sqrt{\Sigma_{ii}\Sigma_{jj}}}. \tag{3.10}$$

The resulting correlation coefficient between the *SSL* and *Fees* parameters is $\rho_{SSL,Fees} = 2.538$. This result confirms that the security and fees metrics are positively correlated. Indeed, agents offering security measures to their service calls will possibly charge other agents a service call fee. All the correlation coefficients reported here are calculated with 95% confidence intervals.

QWS: This dataset reports the following performance metrics of 2507 real agent-based web services available on the web: response time, availability, throughput, successability, reliability, compliance (to the WSDL specification provided by W3C standards), best practices (according to the web service interoperability, WS-I, basic profile), latency, and documentation. A sample of this dataset is given by Table 3.3. We applied Equations 3.9 and 3.10 to compute the correlation matrix of these metrics which is displayed in Table 3.4.

Response Time (ms)	**Availability (%)**	**Throughput (invokes/sec)**	**Successability (%)**	**Reliability (%)**	**Compliance (%)**	**Best Practices (%)**	**Latency (ms)**	**Documentation (%)**	**Service Name**
302.75	89	7.1	90	73	78	80	187.75	32	MAPPMatching
482	85	16	95	73	100	84	1	2	Compound2
3321.4	89	1.4	96	73	78	80	2.6	96	USDAData
126.17	98	12	100	67	78	82	22.77	89	GBNIRHolidayDates

Table 3.3: A sample from the QWS dataset

	Response Time	Availability	Throughput	Successability	Reliability	Compliance	Best Practices	Latency	Documentation
Response Time	1.0000								
Availability	-0.0664	1.0000							
Throughput	-0.2530	0.2007	1.0000						
Successability	-0.0773	0.9892	0.2007	1.0000					
Reliability	0.0471	0.1289	0.2556	0.1211	1.0000				
Compliance	-0.0828	0.2436	0.0603	0.2609	-0.0300	1.0000			
Best Practices	0.0327	0.0571	0.1684	0.0554	0.6895	0.0336	1.0000		
Latency	0.3907	-0.0988	-0.1450	-0.1107	-0.0239	-0.0773	-0.0079	1.0000	
Documentation	-0.0402	-0.0058	-0.0311	0.0044	0.0606	-0.0803	-0.0366	-0.0403	1.0000

Table 3.4: Correlation of QWS dataset QoS metrics

We then select the most significant correlation coefficients; the ones with an absolute value greater than 2.5. Subsequently, we display the normalized values of the corresponding QoS metrics for each selected correlation coefficient which is plotted as a regression line in Figure 3.1. The strongest positive correlation is between *Availability* and *Successability* with a value of 0.9892 after which comes the correlation between *Reliability* and *Best Practices* with a value of 0.6895. The strongest negative correlation is observed between the *Response Time* and *Throughput* with a value of -0.2530. This result is consistent with the technical nature of these two metrics; the smaller the *Response Time* of an agent-based service, the higher the number of requests it could possibly process.

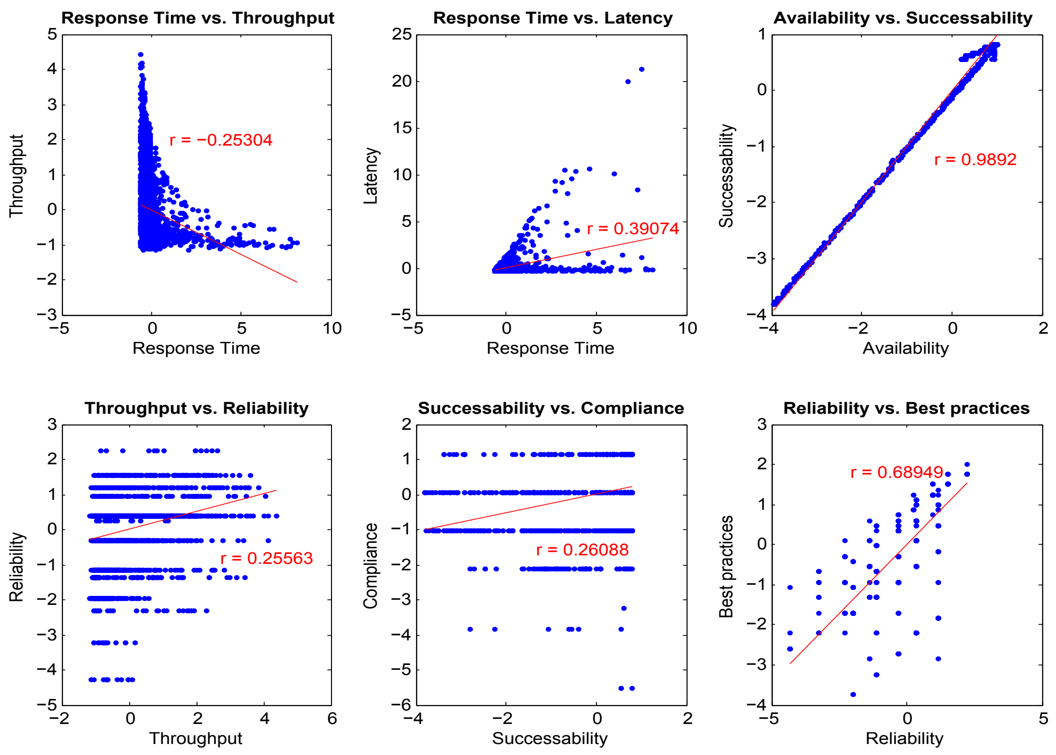


Figure 3.1: Correlation between six pairs of QoS metrics from the QWS data set

3.2.2 QoS-Aware Trust Model

The trust model we propose consists of two main modules. The first one allows the agents to select, for each web service they intend to interact with, the QoS metrics they choose to scrutinize. The majority of existing trust models designate a binary variable to represent the outcomes of QoS metrics (1 for successful outcome, 0 otherwise). This representation lacks the explanation of when the outcomes of the quality metrics are considered successful and when they are not. One possible solution to this issue is to specify a threshold under or above which a quality metric is perceived successful. This threshold may vary depending on the type of data handled by the agent and the requirements set by the designer of the trust system. For instance, a data critical agent-based service should be very reliable which requires a higher threshold for its reliability metric. Metrics such as RE, with values directly proportional to the quality of the service, will be assigned to 1 if they are above the defined threshold and 0 otherwise. The QoS metrics with values indirectly proportional to the overall quality of the service (such as RT) will be assigned to 1 if they are below the defined threshold and 0 otherwise. For example, the RT threshold can be set to 300 milliseconds (ms). Then, any response under 300 ms is deemed successful. However, if the TH threshold is set to 10 and the time unit is a second, then handling 10 requests or more per second is considered successful and unsuccessful otherwise. This seems a practical solution for the problem in hand. However, the question becomes what values do we assign to these thresholds? As a matter of fact, there is no magical values to be appointed to the thresholds of these metrics. Nonetheless, it is reasonable to assume that for the metrics proportional to the quality of the service, a higher threshold implies higher quality standards and expectation. A lower threshold in the case of indirectly proportional metrics has the same implications. An agent looking to interact with an agent-based web service that deals with sensitive data will necessarily set a high threshold for the security aspect of the agent. Similarly, an agent that places high emphasis on productivity will certainly be interested in a respective low and high thresholds for the response time and throughput metrics. Therefore, these thresholds will be set by the truster agents (agents looking to interact with other agents) to meet their non-functional requirements.

The performance monitoring of agent-based web services is not the focus of this thesis. We presented the module above as a preparatory phase to set the stage for the main contributions incorporated in the second module of the proposed approach. In this module, each truster agent will be enabled to estimate the expected trustworthiness of other agents given the QoS outcomes of their past interactions. This assists the former in maximizing their gain by selecting and interacting with the agent that is assigned the highest trustworthiness score. The details of the trust estimation approach based on the outcomes of the QoS metrics of agent-based services are described in the subsequent section.

3.2.3 Trust with multi-dimensional QoS

The second module of our approach implements a QoS-aware trust model that leverages the correlation information among various QoS metrics. This model, based on the probability theory, estimates the trustworthiness of web services by exploiting two statistical distributions, namely, Dirichlet and generalized Dirichlet. These distributions represent the outcomes of multiple correlated QoS metrics. The former distribution is employed when the QoS metrics are positively correlated while the latter handles negatively correlated metrics.

Suppose the following scenario: An agent wants to interact with an other agent-based web service that returns a list of information about multiple stocks. The former requires the latter to be available, reliable and fast. Therefore, the corresponding three QoS metrics to be considered are AV, RE and RT. For clarity of notation, we use av, re and rt in the rest of this chapter. After each interaction with an agent, the truster agent keeps track of the outcomes of each of the three QoS metrics. Each of the latter is assigned the values 1 or 0 to represent a successful or unsuccessful outcome, respectively, as discussed above. After N interactions, the vector $\vec{X} = (X_{av}, X_{re}, X_{RT})$ represents the number of interactions for which each of the QoS metrics had successful outcomes. Since $\vec{X}$ is a vector of counts, it is common to assume that it is generated by a multinomial distribution. The Dirichlet distribution is a widely used conjugate prior to the multinomial in various applications including QoS modeling [42]. Thus, to estimate the probabilities of each QoS metric to have an outcome equal to 1, $\hat{p}(X) = (E(P_{av}), E(P_{re}), E(P_{RT}))$, we apply a Dirichlet prior with the parameters $(\alpha_{av}, \alpha_{re}, \alpha_{RT})$. The density obtained after introducing this prior is called the multinomial Dirichlet distribution (MDD), the multi-dimensional case of the Beta binomial distribution. The posterior is then a Dirichlet with parameters $(\alpha_{av} + X_{av}, \alpha_{re} + X_{re}, \alpha_{RT} + X_{RT})$. The derived estimates of av, for example, are given by:

$$E(P_{av}) = \frac{\alpha_{av} + X_{av}}{\sum_{l=1}^{d+1} \alpha_l \sum_{l=1}^{d+1} X_l}, \tag{3.11}$$

where d is the dimension in which the Dirichlet is defined. Since $\sum_{l=1}^{d+1} P_l = 1$, d in this case is equal to 2 as P_3 can be inferred from the first P_1 and P_2. Despite the flexibility of the Dirichlet distribution, its negative covariance matrix limits its applicability:

$$Cov(P_i, P_j) = -\frac{\alpha_i \alpha_j}{(\sum_{i=1}^{d+1} \alpha_i)^2 (\sum_{i=1}^{d+1} \alpha_i + 1)}. \tag{3.12}$$

Indeed, this covariance is presuming that any two random variables in P_i and P_j are negatively correlated which is not always the case [38]. We have also shown that the QoS metrics can be either positively or negatively correlated. Moreover, the fact that the variables with the same mean must have the same variance [39] adds another restriction to the use of Dirichlet that makes certain models unrealistic [35]. To transcend these disadvantages, we use the generalized Dirichlet distribution (GDD) to replace the Dirichlet distribution. Several studies have shown the flexibility of the generalized Dirichlet and its advantages in various applications such as text modeling [43], and image processing [44]. In dimension d, the GDD with parameter vector $\vec{\alpha} = (\alpha_1, \beta_2, ..., \alpha_d, \beta_d)$ is defined by [40] as:

$$p(P_1, ..., P_d) = \prod_{l=1}^{d} \frac{\Gamma(\alpha_l + \beta_l)}{\Gamma(\alpha_l) + \Gamma(\beta_l)} P_l^{\alpha_l - 1} (1 - \sum_{j=1}^{l} P_j)^{\gamma_l}, \tag{3.13}$$

for $\sum_{l=1}^{d} P_l < 1$ and $0 < P_l < 1$ for $l = 1, ..., d$, where $\alpha_l > 0$, $\beta_l > 0$, $\gamma_l = \beta_l - \alpha_{l+1} - \beta_{l+1}$ for $l = 1, ..., d-1$, and $\gamma_d = \beta_d - 1$. Also, note that $\Gamma(x) = \int_0^\infty t^{x-1} \mathrm{e}^{-t} \mathrm{d}t$. The moments of the GDD are given by:

$$E(P_l) = \frac{\alpha_l}{\alpha_l + \beta_l} \prod_{k=1}^{l-1} \frac{\beta_k}{\alpha_k + \beta_k}, \tag{3.14}$$

$$V(P_l) = E(P_l) \Big(\frac{\alpha_l + 1}{\alpha_l + \beta_l + 1} \prod_{k=1}^{l-1} \frac{\beta_k + 1}{\alpha_k + \beta_k + 1} - E(P_l) \Big). \tag{3.15}$$

GDD has $d - 1$ more parameters than the Dirichlet which gives it d degrees of freedom when used to construct a prior. It also has a more general covariance and does not restrict variables with the same mean to have the same variance [35]. Additionally, it is conjugate to the multinomial distribution. The mean, variance, and covariance of the GDD are available in [40]. The marginal distribution of $\vec{X}$ obtained from the employment of a GDD prior is given by:

$$\begin{aligned}
p(\vec{X}|\vec{\alpha}) &= \int_{\vec{P}} p(\vec{X}, \vec{P}|\vec{\alpha}) d\vec{P} \\
&= \frac{\Gamma((\sum_{l=1}^{d+1} X_l) + 1)}{\prod_{l=1}^{d+1} \Gamma(X_l + 1)} \prod_{l=1}^{d} \frac{\Gamma(\alpha_l + \beta_l)}{\Gamma(\alpha_l) + \Gamma(\beta_l)} \\
&\times \prod_{l=1}^{d} \frac{\Gamma(\alpha'_l) + \Gamma(\beta'_l)}{\Gamma(\alpha'_l + \beta'_l)}
\end{aligned} \tag{3.16}$$

where $\alpha'_k = \alpha_l + X_l$ and $\beta'_l = \beta_l + X_{l+1} + \cdots + X_{d+1}$ for $l = 1, \ldots, d$. The above density is called the multinomial generalized Dirichlet distribution (MGDD). Therefore, using the generalized Dirichlet as a prior to the multinomial yields the following moments:

$$E(P_l) = \frac{\alpha_l + X_l}{\alpha_l + \beta_l + n_l} \prod_{k=1}^{l-1} \frac{\beta_k + n_{k+1}}{\alpha_k + \beta_k + n_k}, \tag{3.17}$$

where $n_l = X_l + X_{l+1} + \cdots + X_{d+1}$, and

$$\begin{aligned}
V(P_l) = E(P_l)\Big(&\frac{X_l + \alpha_l + 1}{\alpha_l + \beta_l + 1 + n_l} \times \\
&\prod_{k=1}^{l-1} \frac{\beta_k + 1 + n_{k+1}}{\alpha_k + \beta_k + 1 + n_k} - E(P_l)\Big).
\end{aligned} \tag{3.18}$$

The probabilities that each of the QoS metrics outcome is equal to 1 given by $\hat{p}(X)$ are used to estimate the trustworthiness of the evaluated service provider. In addition to these probabilities, the service consumer assigns each of the QoS metrics a weight that represents its value in the overall quality required by the consumer. The trustworthiness of a service provider sP as estimated by a service consumer sC is denoted by $Tr(sP, sC)$ and given by Equation 3.19. The uncertainty in the computed trust is equally important to communicate the confidence of the service consumer in the estimated trust score. This is supplied by the variance of the trust score, $Var(Tr(sP, sC))$, defined in Equation 3.20.

$$Tr(sP, sC) = \sum_{l=1}^{d+1} E(P_l) W_l, \tag{3.19}$$

where W is the vector of consumers predefined weights of the significance of each of the $d + 1$ QoS metrics in the quality required by sC from sP.

$$Var(Tr(sP, sC)) = W Cov(\vec{P}) W^T, \tag{3.20}$$

where $Cov(\vec{P})$, is a covariance matrix that represents the variances or uncertainties in $\vec{P}$ (diagonal terms) and the correlations between each pair of QoS metrics (off-diagonal terms). The variances in $\vec{P}$ are calculated following the same process used for $\vec{P}$ by marginalizing over the variances given by Equation 3.18. The off-diagonal terms necessitate the computation of the covariances between each pair of QoS metrics which are given by the covariance of the MGDD over the joint probabilities of each pair of metrics. To simplify the notation, let t be the position of the combined outcomes which, in case of pairwise combinations of binary values, $t = 1, \ldots, 4$. Thus, for each of the 11, 10, 01, and 00 outcomes, t is set to 1, 2, 3, and 4, respectively. Back to the example with the three QoS metrics av, re, and rt, we define the following: $s_t^{av,re}$ is the number of QoS outcomes for which av and re had the combined outcome at position t. $E_t^{av,re}$ is the estimate of the outcomes of av and re at position t. $Var_t^{av,re}$ is the variance in the estimate $E_t^{av,re}$ and $C_{tu}^{av,re}$ is the covariance between the estimates $E_t^{av,re}$ and $E_u^{av,re}$, where $u = 1, \ldots, 4$ represents a position of the combined outcomes 11, 10, 01, and 00, s.t. $u \neq t$. Similar notations are used for the other pairs, (av, rt) and (re, rt). The means, variances, and covariances of the MGDD over the pairs of QoS metrics follow Equations 3.21, 3.22, and 3.23 that are defined with the av, rt pair:

$$E_t^{av,rt} = \frac{\alpha_t^{av,rt} + s_t^{av,rt}}{\alpha_t^{av,rt} + \beta_t^{av,rt} + n_t^{av,rt} + 1} \times \prod_{k=1}^{t-1} \frac{\beta_k^{av,rt} + n_{k+1}^{av,rt}}{\alpha_k^{av,rt} + \beta_k^{av,rt} + n_k^{av,rt}} \tag{3.21}$$

$$V_t^{av,rt} = E_t^{av,rt}\Big(\frac{\alpha_t^{av,rt} + s_t^{av,rt} + 1}{\alpha_t^{av,rt} + \beta_t^{av,rt} + n_t^{av,rt} + 1} \times \prod_{k=1}^{t-1} \frac{\beta_k^{av,rt} + n_{k+1}^{av,rt} + 1}{\alpha_k^{av,rt} + \beta_k^{av,rt} + n_k^{av,rt} + 1} - E_t^{av,rt}\Big) \tag{3.22}$$

$$C_{tu}^{av,rt} = E_u^{av,rt}\Big(\frac{\alpha_t^{av,rt} + s_t^{av,rt}}{\alpha_t^{av,rt} + \beta_t^{av,rt} + n_t^{av,rt} + 1} \times \prod_{k=1}^{t-1} \frac{\beta_k^{av,rt} + n_{k+1}^{av,rt} + 1}{\alpha_k^{av,rt} + \beta_k^{av,rt} + n_k^{av,rt} + 1} - E_t^{av,rt}\Big) \tag{3.23}$$

3.2.4 Practical Example

To exemplify, suppose we have 10 outcomes of the three QoS metrics given in Table 3.5. The vector of counts that represent the total number of each metric being successfully fulfilled is $\{6,\ 6,\ 7\}$. The probability of each QoS metric outcome being equal to 1 is then calculated using Equation

3.17 and is given by $\hat{p} = E(P_{av}) = 0.3333,\ E(P_{re}) = 0.3111,\ E(P_{rt}) = 0.3556$. The pairwise outcomes counts are displayed in Table 3.6. The joint probabilities of the pairwise outcomes are computed by Equation 3.21 and given in Table 3.7.

Availability (a)	1	1	0	1	1	0	1	0	1	0
Reliability (r)	1	0	0	1	0	1	1	1	1	0
Response time (rt)	1	1	1	0	0	1	1	1	0	1

Table 3.5: 10 outcomes of three QoS metrics

QoS metrics pair	11	10	01	00
(a, r)	4	2	2	2
(a, rt)	3	3	4	0
(r, rt)	4	2	3	1

Table 3.6: Counts of pair-wise outcomes combinations

	P_{11}	P_{10}	P_{01}	P_{00}
(a, r)	0.4167	0.2188	0.1823	0.1823
(a, rt)	0.3333	0.2963	0.3086	0.0617
(r, rt)	0.4167	0.2188	0.2431	0.1215

Table 3.7: Mean of pair-wise outcomes combinations

The uncertainties in the above estimates and the correlations between the pairs of QoS metrics are given by the following covariance matrix.

$$Cov(p(X)) = \begin{pmatrix} V_a & C_{a,rt} & C_{a,rt} \\ C_{a,rt} & V_r & C_{r,rt} \\ C_{a,rt} & C_{r,rt} & V_{rt} \end{pmatrix} \tag{3.24}$$

The diagonal terms are obtained from the variances calculated using Equation 3.18 with the vector of counts $\{6,\ 6,\ 7\}$; $V_a = 0.0101,\ V_r = 0.0093,\ V_{rt} = 0.0147$. To evaluate the off-diagonal terms, we compute the covariances of the counts displayed in Table 3.6. For example, to compute $C_{a,rt}$, we employ Equation 3.23 to calculate the covariances of the estimates of positive outcomes of the metrics a and r; $C^{a,rt}_{t=1,u=3} \equiv C^{a,rt}_{1101}$, $C^{a,rt}_{t=2,u=1} \equiv C^{a,rt}_{1011}$, and $C^{a,rt}_{t=2,u=3} \equiv C^{a,rt}_{1001}$. We also compute the variance in positive outcomes, $V_{t=1}^{a,rt} \equiv V_{11}^{a,rt}$ using Equation 3.22. $C_{a,rt}$ is then given by:

$$C_{a,rt} = C_{13}^{a,rt} + C_{21}^{a,rt} + C_{23}^{a,rt} + V_{11}^{a,rt} \tag{3.25}$$

The resulting covariance matrix is given by:

$$Cov(p(X)) = \begin{pmatrix} 0.0101 & 0.0034 & -0.0044 \\ 0.0034 & 0.0093 & 0.0006 \\ -0.0044 & 0.0006 & 0.0147 \end{pmatrix} \tag{3.26}$$

Therefore, given the observations from Table 3.5, the service consumer will report $\hat{p}(X)$, and $Cov(p(X)$. The former depicts the probabilities that each of the QoS metrics of the evaluated service provider is fulfilled successfully. The latter provides the uncertainties in these probabilities and the correlations between the observed QoS metrics. The service consumer will then use $\hat{p}(X)$ and $Cov(p(X)$ to compute the trustworthiness of the service provider using Equation 3.19. Suppose the consumer associates the following weights with each of the observe QoS metrics: $W = \{0.25,\ 0.35,\ 0.45\}$, the resulting trust value will be: $Trust = 0.25 \times 0.3333 + 0.35 \times 0.3111 + 0.45 \times 0.3556 = 0.3522$. The uncertainty in this trust estimate is computed using Equation 3.20, which yields to $var(Trust) = 0.0072$.

It is noteworthy to mention that the proposed model scales to multi-valued QoS metrics. Assuming we have k-valued QoS metrics, we will have 2^K pairwise outcomes instead of just 4 (2^2) in the binary case. The reason behind considering binary values is the intuitive mechanism that classifies each metric into two classes; if a consumer believes, after interacting with a web service, that a given metric was fulfilled, this metric is assigned to class "1". Otherwise, if the consumer was not satisfied with the outcome of a specific metric, it is then assigned to class "0". In the case of multi-valued metrics, one can define k ranges of values according to which each metric will be classified into 1 to k classes. However, the challenge becomes to justify the boundaries of each of the ranges. Another issue with considering multi-valued metrics is the need for more interactions with a web service so the vector of counts of pairwise outcomes does not mainly consist of 0s.

3.3 Online Trust: Generalized Dirichlet Power Steady Model

The interactions with web services occur in a dynamic and real-time environment. The trust models described so far function in a batch learning framework. In other words, a collection of data samples is needed to perform the learning. In this case, the ratings of past interactions with a web service are needed before the model can estimate its trustworthiness. In more realistic scenarios, a large number of past interactions are not always available to pass their ratings to the batch learning algorithm. A service consumer would rather need to evaluate the trustworthiness of a web service after few interactions and ultimately after each interaction. This allows a dynamic update of the trust models of web services leading more accurate estimations and, subsequently, an optimized selection process. To fulfill this requirement, we propose a time series approach that is based on a generalized Dirichlet power steady model. This approach aims to model the proportions of the ratings of web services and forecast their future ones.

This section presents a power steady model (PSM) based on GDD observations to model and predict compositional time series. The models unobserved states evolve according to the GDD conjugate prior distribution which is transformed to a set of Beta distributions each of which is denoted by a 2-dimensional re-parametrized Dirichlet. We demonstrate that subdividing the modeling of a time series into multiple smaller problems lead to better prediction results. We evaluate this model with the web service selection application. Specifically, we analyze the proportions of the quality classes that are assigned to the outcomes of the web services interactions. Our model is compared with another PSM that assumes Dirichlet observations. The experiments show promising results in terms of standardized residuals and mean squared errors of predictions.

3.3.1 Compositional Time Series

Time series of continuous proportions that are also known as compositional data, have been analyzed and modeled using various approaches [45, 46]. This kind of series presents itself in various domains such as economics (e.g., yearly provincial gross domestic product), chemistry (e.g., chemical compositions of rocks) and political sciences (e.g., vote and seat shares). Generally, time series of compositional data are multivariate and denoted by time-dependent vectors of proportions that sum to one. To model such data, one might resort to standard techniques such as the multivariate autoregressive integrated moving average (ARIMA) [47] and Kalman filters. However, due to the positive nature of the components of compositional data and their sum to one constraint, these techniques are deemed not applicable [45].

Various approaches have been proposed to deal with the positivity and dependence of compositional data's components. For instance, Aitchison proposed in [45] the mapping of the data from the positive simplex

$$\mathbb{S}^d = \{(s_1, \ldots, s_d), s.t. \sum_{i=1}^{d} s_i < 1\},$$

to the d-dimensional real space $\mathbb{R}^d$. Specifically, the author suggested the additive and multiplicative logistic transformations. Let $\vec{s}$ be a vector defined on the positive simplex $\mathbb{S}^d$ and $\vec{y}$ is another vector defined on the real space $\mathbb{R}^d$. Therefore, the aforementioned transformations are defined as:

Additive log-ratio transformation:

$$y_i = \log \frac{s_i}{s_{d+1}} \,. \tag{3.27}$$

The inverse: additive logistic transformation:

$$s_i = \begin{cases} \frac{\exp(y_i)}{1+\sum_{j=1}^{d} \exp(y_i)} & \text{for } i = 1, \ldots, d \\ \frac{1}{1+\sum_{j=1}^{d} \exp(y_i)} & \text{if } i = d+1 \end{cases} \tag{3.28}$$

Multiplicative log-ratio transformation:

$$y_i = \log \frac{s_i}{1 - \sum_{j=1}^{i} s_{d+1}},$$

The inverse: multiplicative logistic transformation:

$$s_i = \begin{cases} \frac{\exp(y_i)}{\prod_{j=1}^{d} 1+\exp(y_i)} & \text{for } i = 1, \ldots, d \\ \frac{1}{\prod_{j=1}^{d} 1+\exp(y_i)} & \text{if } i = d+1 \end{cases} \tag{3.29}$$

where s_{d+1} is considered as the fill-up value. Inspired by Aitchison proposals, [46] employed the multivariate ARIMA to model compositional time series transformed using the above additive logistic transformation. The authors demonstrated the practicality of this transformation via a public opinion polls application. However, they argued that the limitation of such approach is dealing with zero values of s_i which yield $y_i = \pm\infty$. In the same line of research, the authors in [48] used the same transformation with multivariate dynamic linear models. To circumvent the zero-infinity issue, looking for a replacement for the additive transformation might be the answer. For instance, in [49], the authors proposed an alternative approach that employs a hyperspherical transformation. This was intended to overcome the positivity and unit-sum constraints of compositional data. It also promised to solve the problems that arise when cases have zero-valued components. This approach is motivated by the problem of maintaining the unity-sum when forecasting is based on modeling each component of the time series by the available data instances. To overcome this problem, the compositional time series will first be mapped through a non-linear dimensionality reduction approach onto a hypersphere. As such, a time series of dimension d will be reduced to $d - 1$. After transforming the original data via a square root transformation, $y_i = \sqrt{s_i}$,the illustration of the hypersphere mapping is given by:

Hyperspherical transformation:

$$y_i = \cos\theta_i \prod_{j=i}^{d} \sin\theta_{j+1} \ , \tag{3.30}$$

where θ_i is the hyperspherical coordinate of y_i that is computed recursively as follows:

$$\theta_i = \arccos\Big(\frac{y_i}{\prod_{j=2}^{d} sin\theta_{j+1}}\Big)$$

Furthermore, the authors in [50] suggested a more general transformation, the Box-Cox, that given $\vec{s}$, $\vec{y}$ is obtained as follows:

Box-Cox transformation:

$$y_i = \begin{cases} \log \frac{(\frac{s_i}{s_{d+1}})^{\lambda_i - 1}}{\lambda_i} & \text{if } \lambda_i \neq 0 \\ \log \frac{s_i}{s_{d+1}} & \text{if } \lambda_i = 0 \ , \end{cases} \tag{3.31}$$

where $\lambda_i \in \mathbb{R}$ is the unknown Box-Cox parameter. Afterward, the authors proposed a regression model that is framed in a dynamic Bayesian structure to model compositional time series. As illustrated above, the additive logistic transformation is a special case of the Box-Cox transformation.

Forecasting is a major part of the body of time series literature. The various models developed for time series forecasting during 25 years up to 2006 are reviewed in [51]. The classes of reviewed models include the exponential smoothing methods, ARIMA, state space and structural models, Kalman filters, and autoregressive conditional heterscedastic (ARCH) models. Compositional time series, in particular, were also subject to various forecasting approaches influenced by those applied with any time series such as the ones previously mentioned. Particularly, state space models have been exploited for analyzing, modeling, and forecasting time series. These models consist of observation and state processes that may be non-linear and non-Gaussian. The main usage of such models is to deduce the properties of the states given the knowledge from the observations. They are also employed for forecasting and parameter estimation. It is noteworthy to mention that all autoregressive moving average (ARMA) and ARIMA may be written as state space models. In the case of linear processes, Kalman filters are used to solve the corresponding state space models. In [52], the authors developed a Dirichlet state space model (*DSPM*) to forecast compositional time series. Their model also includes an estimation approach of these series' trends, covariates, and interventions. They compared three models, the power steady, trend, and covariate models, using the Bayesian information criterion (BIC) metric. They used a motor vehicle production data set that consists of the number of vehicle production in Japan, United States, and the rest of the world during the years 1947 to 1987.

Motivation: As mentioned earlier, a wide range of real applications in varied domains involve compositional time series. The majority of these applications handle series that consist of yearly, quarterly, or monthly proportions. However, with the plethora of online data, some compositional time series may arise on a daily or even hourly basis. For example, the geographic distribution of the users of social media websites may be measured on an hourly basis for various business related functions. Therefore, given the large amount of available data, the need to understand this data, and the benefits in turning it into actionable insights, building a modeling and forecasting model for compositional time series becomes of unprecedented significance.

Contributions: We build upon and extend the literature of compositional time series forecasting by the following contributions:

- We propose to model and forecast compositional time series based on a novel PSM in which the observations are assumed to follow a generalized Dirichlet (GD) distribution.

- We transform the GD distribution of d dimensions to d Beta distributions which, in turn, are transformed to d unidimensional Dirichlet distributions in their exponential form. This approach partitions the modeling and forecasting of (d + 1)-dimensional time series into d smaller problems with fewer parameters to learn.
- We evaluate our model with a new application, web service selection, in comparison to outdated ones used in the literature.
- We compare our models forecasting performance to that of the Dirichlet-based power steady model (DPSM) proposed in [52]. We show the outperformance of our model via standardized residuals and mean squared error of the predictions.

3.3.2 Generalized Dirichlet Formulation

Let $\vec{X} = (X_1, \dots, X_{d+1})$ denote a vector of proportions that follows a d-dimensional GD distribution with the parameters vector $\vec{\alpha} = (\alpha_1, \beta_1, \dots, \alpha_d, \beta_d)$. Then, the probability distribution function of $\vec{X}$ is given by Equation 3.13. Since $\vec{X}$ follows a generalized Dirichlet and is completely neutral, it can be transformed to d independent Beta distributions [53, 54]. Let $\vec{Y} = (Y_1, \dots, Y_d)$ be the result of the following transformation:

$$Y_j = \begin{cases} X_j \ , & \text{if } j = 1 \ , \\ \frac{X_j}{1 - X_1 - \dots - X_{j-1}} \ , & \text{if } 2 \leq j \leq d \ . \end{cases}$$

The parameters vector $\vec{\alpha}$ can be estimated by considering that each of the Y_j has a Beta distribution with parameters α_j and β_j. Therefore, the joint probability distribution of $\vec{Y}$ can be written as follows:

$$p(\vec{Y}|\vec{\alpha}) = \prod_{l=1}^{d} B(\alpha_l, \beta_l)^{-1} y_l^{\alpha_l - 1} (1 - y_l)^{\beta_l - 1} \ , \tag{3.32}$$

where $B(\alpha_l, \beta_l) = \frac{\Gamma(\alpha_l)\Gamma(\beta_l)}{\Gamma(\alpha_l + \beta_l)}$. The Beta distribution belongs to the exponential family in which each density is given by the following:

$$p(\vec{Y}|\theta) = H(\vec{Y}) exp\Big(\sum_{s=1}^{S} \eta_s(\theta) T_l(\vec{Y}) + \Phi(\theta)\Big) \ , \tag{3.33}$$

where $\eta_s(\theta)$ are called the natural parameters, $T_l(\vec{Y})$ are the sufficient statistics, $H(\vec{Y})$ is a base measure, and $\Phi(\theta)$ is referred to as the log-partition function. Equation (3.32) can thus be written as an exponential density:

$$\begin{aligned} p(\vec{Y}|\vec{\alpha}) &= \exp\Big[\sum_{l=1}^{d} \log(B(\alpha_l, \beta_l)^{-1}) + (\alpha_l - 1)\log Y_l + (\beta_l - 1)\log(1 - Y_l)\Big] \qquad (3.34) \\ &= \prod_{l=1}^{d} \frac{1}{Y_l(1 - Y_l)} \exp\Big[\sum_{l=1}^{d} \log(\frac{\Gamma(\alpha_l + \beta_l)}{\Gamma(\alpha_l)\Gamma(\beta_l)} + \alpha_l \log Y_l + \sum_{l=d+1}^{2d} \beta_l \log(1 - Y_l)\Big] \ . \end{aligned}$$

Let $S = 2d$, then we have:

$$H(\vec{Y}) = \prod_{l=1}^{d} \frac{1}{Y_l(1 - Y_l)} \ , \tag{3.35}$$

$$T(Y_l) = \begin{cases} \log Y_l & \text{for } l = 1, \ldots, d \ , \\ \log(1 - Y_l) & \text{for } l = d+1, \ldots, 2d \ , \end{cases} \tag{3.36}$$

$$\eta_l(\theta) = \begin{cases} \alpha_l & \text{for } l = 1, \ldots, d \ , \\ \beta_l & \text{for } l = d+1, \ldots, 2d \ , \end{cases} \tag{3.37}$$

$$\Phi(\theta) = \sum_{l=1}^{d} \log\Big(\frac{\Gamma(\alpha_l + \beta_l)}{\Gamma(\alpha_l)\Gamma(\beta_l)}\Big) \ . \tag{3.38}$$

In the case of exponential density functions, a conjugate prior on θ is of the following form [55]:

$$\pi(\theta) \propto \exp\Big(\sum_{s=1}^{S} \rho_s \eta_s(\theta) + \kappa \Phi(\theta)\Big) \ , \tag{3.39}$$

where $(\rho_1, \ldots, \rho_S)$ and κ are the prior's hyperparameters. Therefore, the conjugate prior family to d-dimensional GD distributions transformed to d independent Beta written in their exponential form is given by:

$$\pi(\theta) \propto \exp\Big[\sum_{l=1}^{d} \rho_l \alpha_l + \sum_{l=d+1}^{2d} \rho_l \beta_l + \kappa \sum_{l=1}^{d} \log\Big(\frac{\Gamma(\alpha_l + \beta_l)}{\Gamma(\alpha_l)\Gamma(\beta_l)}\Big)\Big] \ . \tag{3.40}$$

The d Beta distributions that generate $\vec{Y}$, are also simplified unidimensional Dirichlet distributions. In $K+1$ dimensions, the Dirichlet density of a vector of proportions, $\vec{Y} = (Y_1, \ldots, Y_{K+1})$, is given by:

$$p(\vec{Y}|\vec{\alpha}) = \frac{\prod_{j=1}^{K+1} \Gamma(\alpha_j)}{\Gamma(\sum_{j=1}^{K+1} \alpha_j)} \prod_{j=1}^{K+1} Y_j^{\alpha_j - 1} \ , \tag{3.41}$$

where $\vec{\alpha} = (\alpha_1, \ldots, \alpha_{K+1})$ is the parameters vector, $\sum_{j=1}^{K+1} Y_j = 1$ and $0 < Y_j < 1$. This distribution can also be depicted by $Y \sim Dir(\vec{\alpha})$. In the exponential form, the density (3.41) becomes:

$$p(\vec{X}|\vec{\theta}) = \exp\Big[\log\Big(\Gamma\Big(\sum_{l=1}^{K+1} \alpha_l\Big) - \sum_{l=1}^{K+1} \log(\Gamma(\alpha_l)\Big) + \sum_{l=1}^{K+1} \alpha_l \log(X_l) - \sum_{l=1}^{K+1} \log(X_l)\Big] \ . \tag{3.42}$$

In [52], the authors re-parametrized Equation (3.42) to separate the effects of its mean, $\vec{\theta} = \left(\frac{\alpha_1}{\sum_{j=1}^{K+1} \alpha_j}, \ldots, \frac{\alpha_{K+1}}{\sum_{j=1}^{K+1} \alpha_j}\right)$, and spread $\tau = \sum_{j=1}^{K+1} \alpha_j$. The result of this re-parametrization yielded the following:

$$p(\vec{Z}|\vec{\theta}, \tau) = \exp\Big[\tau \vec{Z}^T \vec{\theta} + \tau \frac{\sum_{j=1}^{K+1} W_j}{K+1} - \log\Big(\frac{\prod_{j=1}^{K+1} \Gamma(\theta_j \tau)}{\Gamma(\sum_{j=1}^{K+1} \theta_j \tau)}\Big)\Big] \ , \tag{3.43}$$

where $\vec{W} = \log(\vec{X})$ and $\vec{Z} = \vec{W} - \frac{\sum_{l=1}^{K+1} W_l}{K+1}$. In case $K = 2$, the Beta distribution of each Y_j can be written in the exponential form of a unidimensional Dirichlet as follows:

$$\begin{aligned} p(Y_j|\vec{\theta}) = \exp\Big[&\log\Big(\frac{\Gamma(\alpha_1 + \alpha_2)}{\Gamma(\alpha_1)\Gamma(\alpha_2)}\Big) + \alpha_2 \log(1 - Y_j) \\ &+ \alpha_1 \log(Y_j) - \log(Y_j) - \log(1 - Y_j)\Big] \ . \end{aligned} \tag{3.44}$$

Following the same re-parametrization, Equation (3.44) becomes:

$$p(\vec{Z}'|\vec{\theta}', \tau') = \exp\Big[\tau'(\vec{Z}'^T \vec{\theta}' + W_j') - \log\Big(\frac{\prod_{l=1}^{K} \Gamma(\theta_l' \tau')}{\Gamma(\sum_{l=1}^{K} \theta_l' \tau')}\Big)\Big], \tag{3.45}$$

where $\vec{\theta}' = (\theta_1' = \frac{\alpha_1}{\tau'}, \theta_2' = \frac{\alpha_2}{\tau'})$, $\tau' = \alpha_1 + \alpha_2$, $W_j' = \frac{\log(Y_j) + \log(1 - Y_j)}{2}$, and $\vec{Z}_j' = (\log(Y_j) - W_j', \log(1 - Y_j) - W_j')$. Given these parameters, we represent the distribution of $\vec{Y}$ by $\vec{Y} \sim DirBeta(\tau' \vec{\theta}')$. A conjugate prior family to the Dirichlet distributions in their exponential form is:

$$p(\vec{\theta}'|\sigma, \kappa, \tau') \propto \exp\Big[\sigma\Big[\tau' \vec{\kappa}^T \vec{\theta}' - \log\Big(\frac{\Gamma(\tau'\theta_1')\Gamma(\tau'\theta_2')}{\Gamma(\tau'\theta_1' + \tau'\theta_2')}\Big)\Big]\Big] \ . \tag{3.46}$$

3.4 State Space Models

Dynamic linear models (DLM) can be represented in what is called a state space form. This representation consists in identifying the change of an observed variable (aka observation vector) in terms of another unobserved variable (aka state vector). The authors in [56] proposed a steady model for DLM that is only defined for normal distributions and is equivalent to an ARIMA(0,1,1) model. However, [57] generalized this model by redefining it across non-Gaussian distributions. More specifically, it was generalized to cases where the conditional probability of the observations given the states follows an exponential family distribution. The generalized model, also known as the PSM, is defined by:

$$\begin{cases} (x_t|y^t) \sim PR(\omega) \ , \\ p(x_{t+1}|y^t) \propto p(x_t|y^t)^k \ , \end{cases} \tag{3.47}$$

where $0 < k < 1$ and $PR(\omega)$ is the conjugate prior for the exponential family distribution of $p(y_t|x_t)$. This model was first developed to be applied to univariate observations. However, [58] generalized the PSM of a time series to handle multivariate processes. Specifically, a symmetric multivariate PSM in which the process evolution is defined based on the density of the parameter vector is proposed. This generalization was also introduced as part of a Bayesian forecasting framework. However, this model undergoes some limitations when the observations follow a Dirichlet distribution [52], which are mostly due to the fact that the PSM estimates both the dispersion and location of the distribution at the same time. This problem can be solved by using the re-parametrized form of the Dirichlet distribution (Equation (3.42)) which allows the separation of the dispersion τ and the location θ.

3.5 Generalized Dirichlet Power Steady Model (GDPSM)

This section describes the time series model that is based on the power steady model and reparametrized generalized Dirichlet (GDPSM). Suppose a time series of multivariate observations that consists of continuous proportions, $(X_t : t = 1, \dots, T)$, where $X_t = (X_{t1}, \dots, X_{td+1})$. Suppose that each of these observations follows a GDD (Equation (3.13)), and that a state space model is based on these observations. Instead of dealing with the GD distribution directly, we transform the observation data into sub-spaces in which they follow d Beta distributions as described earlier. We represent each Beta distribution as a unidimensional Dirichlet distribution.

3.5.1 Transformations

Below is a summary of the proposed model. Given a time series of proportions denoted by X : $\{X_t = (X_{t1}, \ldots, X_{td+1})\}$, where $t = 1, \ldots, T$, we first assume that each vector in this time series follows a GD distribution $X_t \sim GD(\alpha_{t1}, \ldots, \alpha_{td}, \beta_{t1}, \ldots, \beta_{td})$. Afterward, we apply a geometric transform on each of these vectors [54, 59].

Geometric transformation:

$$W_{tl} = \begin{cases} X_{tl} \ , \ \text{for } l = 1 \ , \\ (\frac{X_{tl}}{1 - X_{t1} - \cdots - X_{tl-1}}) \ , \ \text{for } l = 2, \ldots, d \ . \end{cases} \tag{3.48}$$

Using this transform, each W_{tl} has a Beta distribution with parameters $(\alpha_{tl}, \beta_{tl})$ that defines the GD distribution which is followed by X_t. Since Beta distributions are special cases of Dirichlet distributions, we finally model the time series by $T \times d$ unidimensional Dirichlet distributions, $X_{tl} = Dir(\alpha_{tl_1}, \alpha_{tl_2})$. Each of the $T \times d$ distributions is then re-parametrized as per Equation (3.44) and (3.45). Subsequently, we build $T \times d$ state space models, each of which is based on an unobserved state $\vec{\theta}'$, to model each of the observations $(W_{11}, \ldots, W_{1d}, \ldots, W_{T1}, \ldots, W_{Td})$. These observations are denoted by:

$$(W_{tj}|\vec{\theta}'_t, \tau'_t) \sim DirBeta(\tau'_t\vec{\theta}'_t) \ , \tag{3.49}$$

where $1 \leq j \leq d$ and $\vec{\theta}'_t$ follows the PSM given by Equation (3.47). In other words, the conditional probability of $\vec{\theta}'_{t+1}$ given the observations $W_{1j}, \ldots, W_{tj}$, is defined as:

$$p(\vec{\theta}'_{t+1}|\vec{W}_t) \propto p(\vec{\theta}'_t|\vec{W}_t)^{\gamma} \quad \text{where } 0 < \gamma < 1 \ . \tag{3.50}$$

Equation (3.50) reveals an interesting property of the $(\vec{\theta}'_{t+1}|\vec{W}_t)$ and $(\vec{\theta}'_t|\vec{W}_t)$ distributions; their modes are equal, but the dispersion of the former is greater.

3.5.2 Time Series Model

The GD time series model is defined by two steps similar to those of a Gaussian Kalman filter: a prediction and an update step. In the prediction step, $p(\vec{\theta}'_{t+1}|\vec{W}_t)$ is computed using Equation (3.50). Both sides of this equation follow the conjugate prior given by Equation (3.46), each with different parameters. Formally, this is given by:

$$p(\vec{\theta}'_{t+1}|\vec{W}_t) \sim \exp\left[\sigma_{t+1|t}\left[\tau'_t\vec{\kappa}^T_{t+1|t}\vec{\theta}'_{t+1} - \log\left(\frac{\Gamma(\tau'_t\theta'_{t+1,1})\Gamma(\tau'_t\theta'_{t+1,2})}{\Gamma(\tau'_t\theta'_{t+1,1} + \tau'_t\theta'_{t+1,2})}\right)\right]\right] , \tag{3.51}$$

$$p(\vec{\theta}'_t|\vec{W}_t) \sim \exp\left[\sigma_{t|t}\left[\tau'_t\vec{\kappa}^T_{t|t}\vec{\theta}_t' - \log\left(\frac{\Gamma(\tau'_t\theta'_{t1})\Gamma(\tau'_t\theta'_{t2})}{\Gamma(\tau'_t\theta'_{t1} + \tau'_t\theta'_{t2})}\right)\right]\right] . \tag{3.52}$$

The prediction step consists of Equation (3.54), which is a known fact and (3.55), which is derived by taking the $\log$ of Equation (3.50), and the two equations above:

$$\begin{aligned}\sigma_{t+1|t}&\left[\tau'_t\vec{\kappa}^T_{t+1|t}\vec{\theta}' - \log\left(\frac{\Gamma(\tau'_t\theta'_1)\Gamma(\tau'_t\theta'_2)}{\Gamma(\tau'_t\theta'_1 + \tau'_t\theta'_2)}\right)\right] \\ &= \gamma\sigma_{t|t}\left[\tau'_t\vec{\kappa}^T_{t|t}\vec{\theta}' - \log\left(\frac{\Gamma(\tau'_t\theta'_1)\Gamma(\tau'_t\theta'_2)}{\Gamma(\tau'_t\theta'_1 + \tau'_t\theta'_2)}\right)\right] ,\end{aligned} \tag{3.53}$$

knowing that:

$$\vec{\kappa}_{t+1|t} = \vec{\kappa}_{t|t} , \tag{3.54}$$

therefore,

$$\sigma_{t+1|t} = \gamma\sigma_{t|t} . \tag{3.55}$$

γ is a model parameter, such that $0 < \gamma < 1$. As for the update step, we need to compute $p(\vec{\theta}'_{t+1}|\vec{W}_{t+1})$ which, according to Bayes' theorem, can be written as:

$$p(\vec{\theta}'_{t+1}|\vec{W}_{t+1}) = p(\vec{W}_{t+1}|\vec{\theta}'_{t+1}) \times p(\vec{\theta}'_{t+1}) , \tag{3.56}$$

where $p(\vec{W}_{t+1}|\vec{\theta}'_{t+1})$ is the data likelihood that follows, in this case, the GD reformulated as $DirBeta(\tau'_t\vec{\theta}'_t)$ and given by Equation (3.45). $p(\vec{\theta}'_{t+1})$ is the prior and is given by Equation (3.46). Therefore, applying the $\log$ to both sides of Equation (3.56) yields the following:

$$\sigma_{t+1|t+1} = \sigma_{t+1|t} + 1 , \tag{3.57}$$

$$\kappa_{t+1|t+1} = \left(1 - \frac{1}{\sigma_{t+1|t+1}}\right)\kappa_{t+1|t} + \frac{1}{\sigma_{t+1|t+1}}z_{t+1} . \tag{3.58}$$

3.5.3 Model Evaluation

We evaluate our model by the standardized residuals, the mean squared error (MSE) of the predictions, and the correlations between the residuals at lag 0. We compare our results with those of *DPSM*. The standardized residuals are computed as follows [52]:

$$R_t = \frac{\vec{Z}_t - E[\vec{Z}_t|D_{t-1}])}{var[\vec{Z}_t|D_{t-1}]}, \tag{3.59}$$

where D_{t-1} denotes all the observations available at time $(t-1)$. $E[\vec{Z}_t|D_{t-1}]$ and $var[\vec{Z}_t|D_{t-1}]$ are the respective posterior mean and variance of the prediction density, formally given by:

$$p(\vec{Z}_{t+1}|\vec{W}_t) = \int p(\vec{Z}_{t+1}|\vec{\theta_{t+1}})p(\vec{\theta}_{t+1}|\vec{W}_t)d\vec{\theta}_{t+1} \ . \tag{3.60}$$

Since there is no direct solution for this density, we use the approximation proposed in [60] and used in [52]. Given the density $p(\vec{Z}_{t+1}|\vec{W}_t)$, its mean is approximated as follows:

$$E[p(\vec{Z}_{t+1}|\vec{W}_t)] = E[p(\vec{Z}_{t+1}|\vec{\Theta})] \ , \tag{3.61}$$

where $\vec{\Theta} = (\sigma, \kappa, \tau')$. According to [52, 61], if a variable follows the Dirichlet distribution in Equation (3.45) with the conjugate prior given by Equation (3.46), then the following equality holds:

$$E[p(\vec{Z}_{t+1}|\vec{\Theta})] = E[p(\vec{Z}_{t+1}|\sigma, \kappa, \tau')] = \kappa \ . \tag{3.62}$$

Therefore, the posterior mean of the density in Equation (3.60) is equal to κ. The posterior variance also lacks an exact solution and is solved in [60] by approximating each term of the following:

$$var[p(\vec{Z}_{t+1}|D_{t-1})] = E[p(\vec{Z}_{t+1}|D_{t-1})^2] - (E[p(\vec{Z}_{t+1}|D_{t-1})])^2. \tag{3.63}$$

Furthermore, we compute the correlations at lag 0 between the residuals of each pair of dimensions in the analyzed time series. These correlations are additional indicators of the model quality; the weaker the correlations the better the model. Stronger correlations imply that further modeling is necessary to better fit the time series [52].

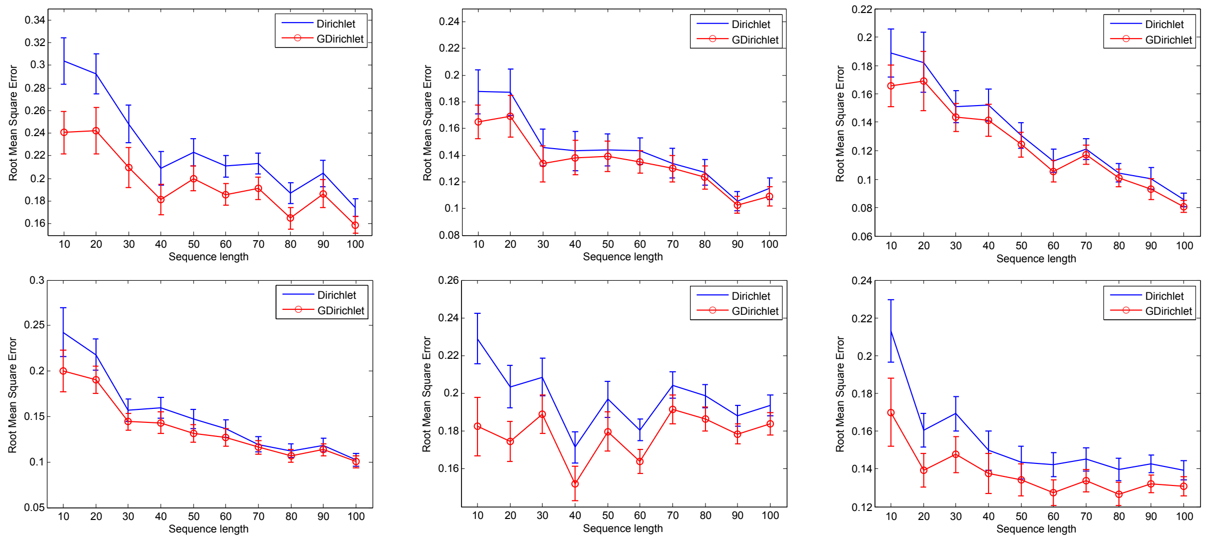


Figure 3.2: MSE of estimating positive correlations

3.6 Experiments

In this section, we assess the potency of the proposed approach with the generalized Dirichlet in leading to accurate estimates of the trustworthiness of agent-based web services. We compare our results with those of the model proposed in [62] which employs the Dirichlet distribution.

3.6.1 Correlation Estimation

In this experiment, we compare the accuracy of estimating the correlations among the QoS metrics using the Dirichlet and generalized Dirichlet. The experiment setup is summarized as follows:

- We consider a five-dimensional QoS (five metrics, $\{M_1, \ldots, M_5\}$) for which we sample a long sequence of 100000 outcomes from a multivariate discretized probability distribution with specified correlations. The correlation matrix calculated from the sampled sequence is:
- We select 10 sequences of varied length starting at a random positions of the long sequence.

	M_1	M_2	M_3	M_4	M_5
M_1	1	-0.2798	0.2695	0.3258	-0.5596
M_2	-0.2798	1	0.3314	0.2199	0.2426
M_3	0.2695	0.3314	1	0.4799	-0.2942
M_4	0.3258	0.2199	0.4799	1	-0.3503
M_5	-0.5596	0.2426	-0.2942	-0.3503	1

Table 3.8: Correlation matrix of 100000 outcomes

- For each sequence we compute $\vec{P}$, $Cov(\vec{P})$, and the correlation between the two metrics using the approach proposed earlier with both Dirichlet and generalized Dirichlet.
- The mean squared errors in the correlation estimations are averaged over 100 runs.

We compare the correlations estimated using the approach described earlier with both Dirichlet and generalized Dirichlet. Figures 3.2 and 3.3 reveal that the former distribution estimates more accurate negative correlations while the latter provides higher accuracy when estimating positive correlations. The strictly negative covariance matrix of the Dirichlet compromises the accuracy of estimating positive correlations.

3.6.2 GDPSM Experiments

Application: Web Service Selection. Business applications are increasingly being deployed as autonomous web applications that are published and used on the web (Web Services). The abundance of web services that provide similar functionalities creates a competitive market while rendering the selection of services that best meet the consumers requirements a challenging task.

A common solution to this problem considers the trustworthiness of services as a selection criterion based on the outcomes of various quality of service (QoS) metrics, including response time, throughput, reliability, availability, security, and cost. These metrics may depend on the network bandwidth, binding and processing performance, as well as underlying hardware. It may also depend on the invocation cost and the honesty of the corresponding provider in fulfilling its functional and quality claims. Based on the aforementioned, we expect a dynamic evolution of the quality of a web service during its lifetime. Therefore, we consider a scenario in which the web service environment consists of the following components:

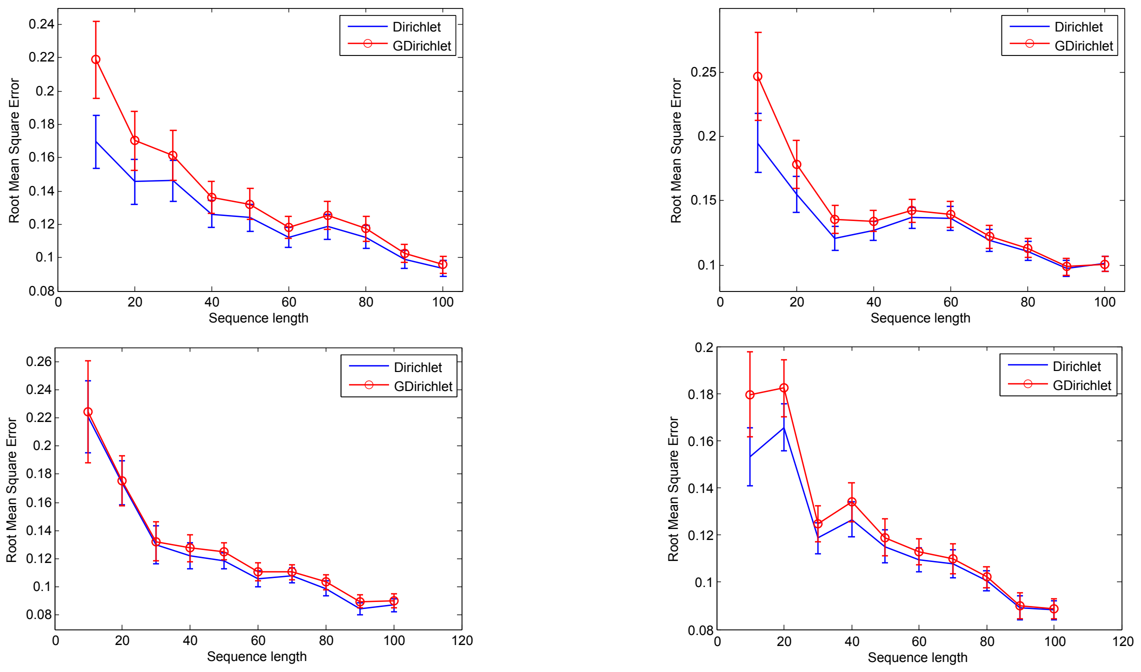


Figure 3.3: MSE of estimating negative correlations

Monitoring component:

This component is responsible for implementing various performance testing tools. Various methods for monitoring the QoS for web services are available in [63]. web service consumer can then evaluate and store, after each interaction with any web service, the values of multiple QoS metrics.

Classification component:

Each vector of QoS metrics' values is then classified into multiple, a priori defined, quality classes. This can be achieved using different classifiers [64].

Counting component:

The last component counts, based on a predefined time interval, the number of interactions with a web service that belong to each of the defined quality classes. For example, given a 1-minute time interval and 3 quality classes, the output of this component is a vector of counts that specifies the number of interactions that were classified in each of the 3 classes (information obtained from the second component).

The first two components have been covered in various studies in the literature [63–65]. We thus assume that such components exist and can be leveraged to build a compositional time series in the third component. The main objective is then to model the QoS-based behavior of web services and predict their future performance to assist the web service selection process. To evaluate our GD time series model, we run different simulations with synthetic data due to the unavailability of real QoS data sets. We are aware of two real data sets; QWS and WS-Dream. The former includes the averages over time of multiple QoS metrics' measurements of $2,507$ web services monitored over a six-day period. As such, the data set includes one quality for each of the monitored web services. The latter reports the response time, http code, and http message of 100 web services over a large number of invocations from 150 computers distributed in more than 20 countries. However, the time of each invocation is not available which makes it hard to build a realistic time-series model for each of the monitored web services. Therefore, we validate our approach with multiple simulated data that embed time-variant processes. The different simulations are summarized as follows:

Simulation 1: trigonometric functions

In this simulation we evaluate our model with the outcomes of a web service's transactions that are classified into D quality classes, that can be illustrated by adjectives such as *Very Good*, *Good*, *Average*,. . . , for the sake of clarity. We make the assumption that the proportions $\mathcal{C} = \{C_1; ...; C_D\}$ of each class during a specific period of time, follow a latent model that we specify using trigonometric functions.

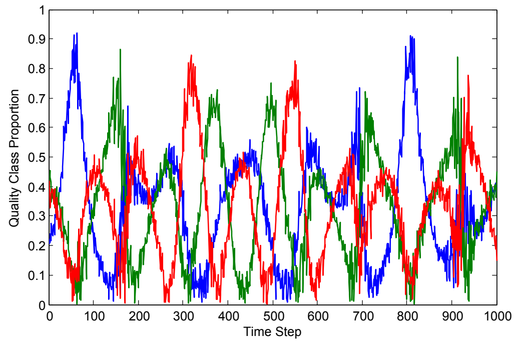
We model each quality class as a function of time steps. At each time step, the number of transactions are counted and assigned to their corresponding quality class. Their overall evolution is modeled as oscillating functions as a web service does not function with the same performance. For example, some web services will have a larger number of requests during certain times of the day or night. During their peak times, the load on the servers on which the web services are deployed will eventually increase, and due to network bandwidths limitations, the response time of these services will inflate. These factors, among others, eventually lead to fluctuations in the provided quality of service. We propose to use trigonometric functions as they are oscillating functions easy to handle and to generate with various settings (mean, amplitude, frequency). These functions can be expressed in the form:

$$C'_i(t) = F_i + \gamma_i \, trig_i(f_i \, t) + \nu_{it}, \;\; i = 1, \ldots, D \;, \tag{3.64}$$

where $trig_i(f_i \, t)$ is either the *cosine* or *sine* function of frequency f_i randomly taken in the range $[0.0001, 0.009]$. These values keep the functions variations at a reasonable level, see Figure 3.4. γ_i is a scaling factor controlling the amplitude of the number of transactions within a given class, A_i is a translation coefficient that controls the average number of transactions of a given class per time step, and ν_t is a white Gaussian noise. t represents the time steps and D the number or quality classes (equal to 3 or 6 here). As ν_{it} is unbounded, the functions C'_i can sporadically go below 0. These rare occurrences are individually handled by assigning a low random value to the sample, within the predefined range $[10, 50]$. In our experiments, we fixed $A_i = 1200$ and $\gamma_i = 1000$ for all i's, and the Signal-to-Noise Ratio has been set to 20. These values can be adapted if a more realistic model is needed without impact on the overall performance of the method presented here. All values are rounded in order to get integers which represent the number of transactions over a given period of time for a given quality class. The proportion vectors are finally obtained by normalizing the C'_i functions, $C_i(t) = \frac{C'_i(t)}{\sum_{d=1}^{D} C'_d(t)}$, where the function C_i represents the proportions of requests that have been processed in *Good*, *Average*, *Poor*,... standing by the web service among the total number of requests sent during a given period of time. The algorithm takes these proportions $C_i(t), t = 1, \ldots, 1000$, as input data, of which the first 20 samples are only used for training purpose and the 980 remaining samples are used as testing data.

In the first experiment, we compare *GDPSM* and *DPSM* with the data obtained from Simulation 1 for 5 different values of $\vec{\gamma} = \{0.001 \;\, 0.250 \;\, 0.500 \;\, 0.750 \;\, 0.999\}$, averaged over 10 runs, for the cases of 3 and 6 quality classes.

Three Quality Classes Results Figure 3.4 displays the data simulated in one of the 10 runs. It also shows the patterns by which the proportions of the quality classes evolve. The standardized residuals computed by Equation (3.59) for *GDPSM* and *DPSM* averaged over each of the $d - 1$ dimensions, are displayed in Table 3.9 (left). For all values of γ, our model's residuals are slightly smaller than the ones given by the *DPSM*. The correlations at lag 0 between the residuals of the first two dimensions computed by *DPSM* and *GDPSM* are -0.4529 and -0.0092, respectively. This shows that our model explains the time series better than *DPSM*.



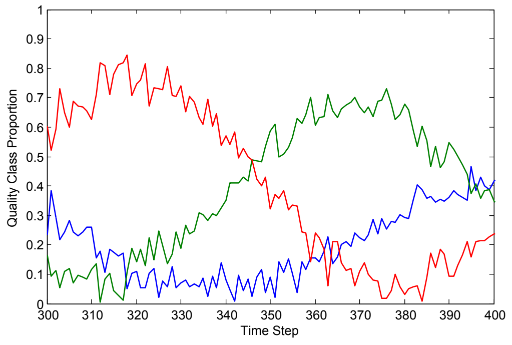


Figure 3.4: Sample data (top) with zoom (bottom)

	Dimension 1		Dimension 2	
γ	DPSM	GDPSM	DPSM	GDPSM
0.001	0.665	0.632	0.647	0.639
0.250	0.666	0.631	0.653	0.645
0.500	0.673	0.637	0.663	0.658
0.750	0.670	0.667	0.696	0.698
0.999	0.855	0.838	0.899	0.900

	Dimension 1		Dimension 2	
γ	DPSM	GDPSM	DPSM	GDPSM
0.001	0.166	**0.067**	0.190	**0.144**
0.250	0.134	**0.054**	0.154	**0.117**
0.500	0.116	**0.048**	0.132	**0.101**
0.750	0.119	**0.050**	0.132	**0.101**
0.999	0.558	**0.276**	0.587	**0.445**

Table 3.9: Standardized residuals (left) and MSE (right) of *GDPSM* and *DPSM* (3-dimensional data)

Furthermore, Table 3.9 (right) shows the mean squared error computed to evaluate the predictions of both *GDPSM* and *DPSM*. It is clear from this table that our model yields more accurate predictions than *DPSM* for both dimensions. To visualize the prediction performance of our model, we display $\vec{Z}$, the symmetric log ratio of the quality class proportions after being transformed using Equation (3.48) (actual data) versus the predicted data ($E[p(\vec{Z}_{t+1}|\vec{W}_t)]$) in Figure 3.5. This figure demonstrates that our model is capable of predicting the time series and providing a smoother distribution than that of the actual ones. The latter is actually due to the fact that we are using a noisy signal. The prediction mostly fits the functional part of the model.

Six Quality Classes Results We repeat the same experiment with another set of 10 different simulated 6-dimensional data, each of which represented by a trigonometric function as given by

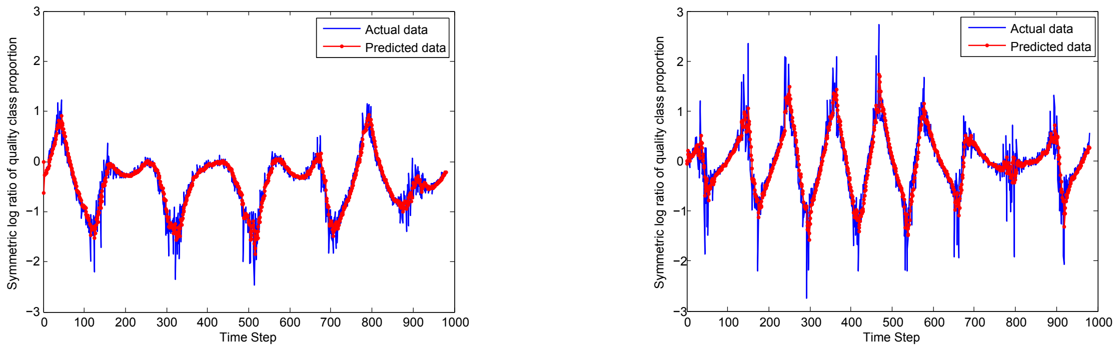


Figure 3.5: Actual versus predicted data for the first (top) and second (bottom) dimensions

Equation (3.64). This aims to further validate the efficiency of partitioning the time series model into d simpler problems to solve and thus lead to lower prediction errors. The average of the standardized residuals of *GDPSM* and *DPSM* over the 10 simulated data are displayed in Table 3.10. For clarity, we only present the results for $\gamma = 0.001$ which raised the best performance for the 3-dimensional residuals (see Table 3.9). It is noteworthy to mention that other values of γ give equivalent results with the exception of $\gamma = 0.999$ that leads to significantly degraded results. This mostly confirms what has been observed in [52]. The correlations at lag 0 between each pair of dimensions are given in Table 3.11.

	Dimension				
	1	**2**	**3**	**4**	**5**
DPSM	0.642	0.625	0.669	0.705	0.693
GDPSM	0.555	0.541	0.62	0.675	0.674

Table 3.10: Standardized residuals for *GDPSM* and *DPSM* with 6-dimensional data

DPSM					**GDPSM**				
1					1				
-0.242	1				-0.040	1			
-0.197	-0.193	1			-0.027	-0.032	1		
-0.144	-0.200	-0.160	1		-0.005	-0.014	-0.017	1	
-0.195	-0.172	-0.196	-0.182	1	0.001	0.001	0.001	0.0003	1

Table 3.11: Residuals correlations at lag 0

It is very clear that our model shows better performance due to the overall smaller correlations. Table 3.12 illustrates the out-performance of *GDPSM* in comparison to *DPSM* in terms of

	Dimension 1		**Dimension 2**		**Dimension 3**		**Dimension 4**		**Dimension 5**	
γ	**DPSM**	**GDPSM**	**DPSM**	**GDPSM**	**DPSM**	**GDPSM**	**DPSM**	**GDPSM**	**DPSM**	**GDPSM**
0.001	0.238	0.075	0.226	0.072	0.221	0.071	0.215	0.074	0.230	0.135
0.250	0.194	0.061	0.185	0.059	0.179	0.057	0.174	0.060	0.186	0.110
0.500	0.169	0.054	0.162	0.052	0.155	0.050	0.149	0.051	0.161	0.095
0.750	0.178	0.057	0.170	0.055	0.156	0.050	0.146	0.052	0.165	0.096
0.999	0.730	0.241	0.714	0.246	0.735	0.254	0.719	0.281	0.677	0.427

Table 3.12: MSE for *GDPSM* and *DPSM* with 6-dimensional data

goodness-of-fit. The MSE of *GDPSM*'s predictions for all the dimensions are two to three times smaller than those of *DPSM*. Figure 3.6 reports the actual and predicted data.

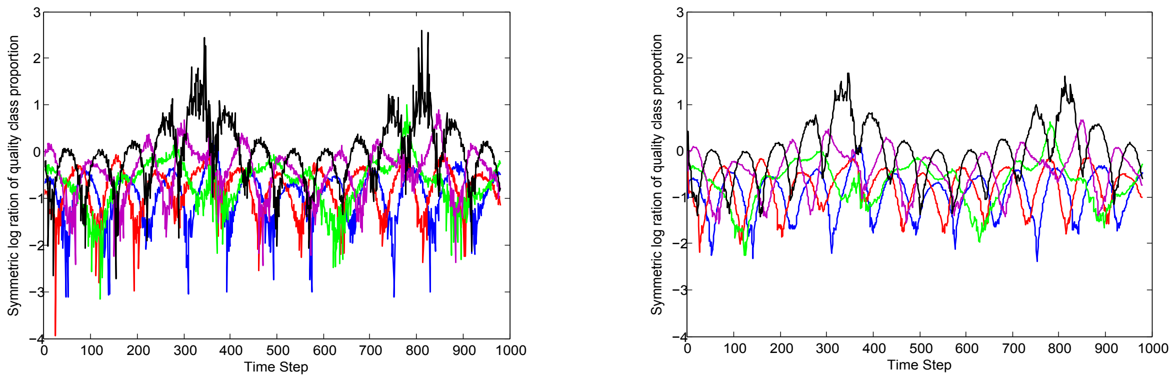


Figure 3.6: Actual (left) versus predicted (right) 6-dimensional function-based data

Simulation 2: Random Data

In this simulation, we test our model with randomly generated 3 and 6 dimensional data. The quality class of a web service interactions do vary according to the time they occurred. In Simulation 1, we showed that *GDPSM* is capable of modeling and forecasting time series generated from noisy time-varying functions. However, it is equally essential for the proposed model to perform well with random data to prove its robustness. Similar to Simulation 1, we compute the standardized residuals, the residuals correlations, and the MSE of the predictions.

Three Quality Classes Results Table 3.13 displays the *DPSM* and *GDPSM* standardized residuals (left) and MSE (right) of predictions. Figure 3.7 shows the actual and predicted data. The correlations between the residuals of the two dimensions as computed by *DPSM* and *GDPSM* are -0.4862 and 0.0116, respectively.

	Dimension 1		**Dimension 2**	
γ	**DPSM**	**GDPSM**	**DPSM**	**GDPSM**
0.001	0.798	0.792	0.801	0.810
0.250	0.799	0.792	0.800	0.810
0.500	0.800	0.792	0.799	0.810
0.750	0.804	0.794	0.802	0.810
0.999	0.808	0.798	0.804	0.804

	Dimension 1		**Dimension 2**	
γ	**DPSM**	**GDPSM**	**DPSM**	**GDPSM**
0.001	0.126	0.071	0.126	0.097
0.250	0.101	0.056	0.102	0.078
0.500	0.084	0.047	0.085	0.065
0.750	0.072	0.040	0.073	0.055
0.999	0.064	0.036	0.064	0.049

Table 3.13: Standardized residuals (left) and MSE (right) for *GDPSM* and *DPSM* with 3-dimensional random data

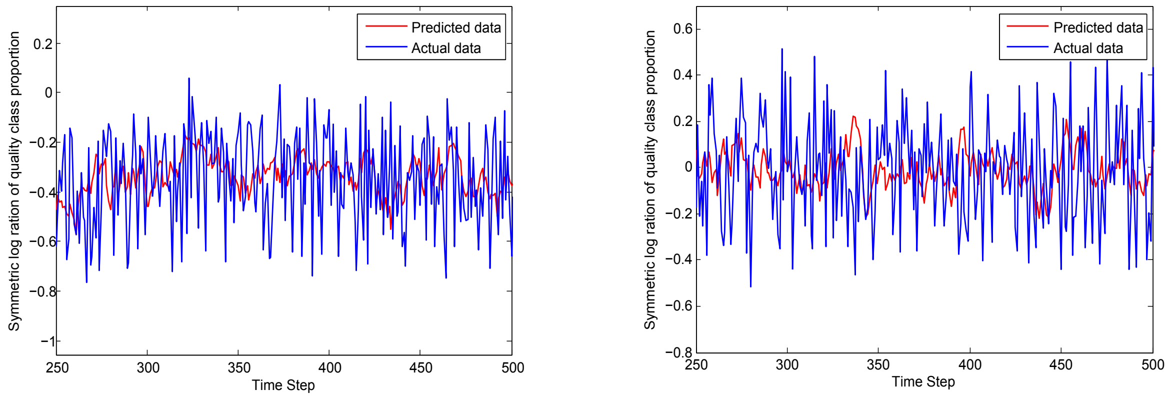


Figure 3.7: Actual (left) versus Predicted (right) 3-dimensional Random data

Six Quality Classes Results We repeat the same experiment above with 6-dimensional simulated random data. It is noteworthy to mention that we select 6 as the higher number of dimensions since it would not realistically make sense to classify a web service quality into more than 6 classes.

	Dimension				
	1	**2**	**3**	**4**	**5**
DPSM	0.801	0.816	0.803	0.794	0.789
GDPSM	0.802	0.81	0.811	0.795	0.799

Table 3.14: Standardized residuals for *GDPSM* and *DPSM* with 6-dimensional random data

DPSM					**GDPSM**				
1					1				
-0.067	1				-0.051	1			
-0.011	-0.070	1			-0.101	-0.107	1		
-0.300	-0.278	-0.299	1		-0.103	-0.103	-0.242		
-0.169	-0.201	-0.199	-0.309	1	0.015	0.0174	0.026	0.034	1

Table 3.15: Residuals correlations at lag 0

	Dimension 1		**Dimension 2**		**Dimension 3**		**Dimension 4**		**Dimension 5**	
γ	**DPSM**	**GDPSM**	**DPSM**	**GDPSM**	**DPSM**	**GDPSM**	**DPSM**	**GDPSM**	**DPSM**	**GDPSM**
0.001	0.137	0.046	0.203	0.075	0.175	0.077	0.494	0.203	0.314	0.158
0.250	0.111	0.037	0.162	0.060	0.140	0.062	0.394	0.162	0.252	0.127
0.500	0.093	0.031	0.135	0.050	0.117	0.051	0.327	0.135	0.210	0.105
0.750	0.079	0.027	0.116	0.043	0.100	0.044	0.280	0.115	0.180	0.090
0.999	0.069	0.023	0.102	0.038	0.088	0.038	0.245	0.101	0.158	0.080

Table 3.16: MSE for *GDPSM* and *DPSM* with 6-dimensional random data

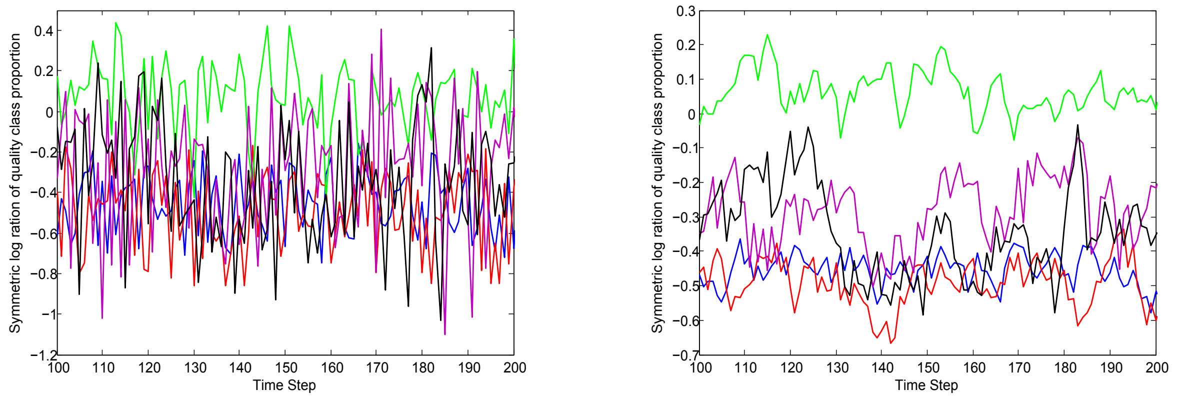


Figure 3.8: Actual (left) versus Predicted (right) 6-dimensional random data

Chapter 4

Trustworthy Composite Web Services Using Probablistic Approaches

A common practice in service oriented computing (SOC) is the integration of multiple services to provide complex business processes through composite services. As defined in [66], “a composite service is an on-demand and dynamic collaboration between autonomous Web service providers that collectively provide a value added service”. An example of a composite service scenario is the over-used intuitive online travel services that may include booking flights, hotels, car rentals, cruises and insurance. Suppose a client is searching for a vacation package that combines the best deal including the flight, hotel and possibly other services. This search function is an example of a business process that can be designed as a composite web service. The integration of these services creates some dependencies among them that will naturally influence the quality of the composed service. For instance, the hotel check-in time might depend on the flight and possibly the car rental agencies available at the airport. To optimize the service composition task, it is essential to capture the quality of the services candidates to composition transactions and their dependencies.

In this chapter, we propose a Bayesian network approach to represent the QoS ratings of composite web services. This approach is based upon the QoS model proposed in Chapter 3. Each node in the network represent the values of the QoS ratings of one of the constituent services. We first employ a the MDD to represent the network’s parameters. We then extend our approach by adding a Beta-Liouville prior to learn the network's parameters. Moreover, we construct two BN classifiers to classify the QoS ratings of composite web services. These classifiers aim to assist the construction of new composite services and the enhancement of existing ones.

4.1 Trustworthiness of Composite Services

A common practice in SOC is the collaboration among different organizations (service providers) to provide composed services. As defined in [66], "a composed service is an on-demand and dynamic collaboration between autonomous Web service providers that collectively provide a value added service". A famous example of a composed services scenario is that of online travel agencies such as Expedia, Orbitz and Travelocity. Such web sites provide multiple services which include booking flights, hotels, car rentals, cruises and insurance. The amalgamation of these services creates some dependencies between them. For instance, the hotel check-in time might depend on the flight and possibly the car rental agencies available at the airport. These dependency relationships will naturally influence the quality of the composite service. Therefore, the necessity to capture the quality of a composite service as well as the responsibility of each of its individual services becomes essential. We will solve this problem by presenting a Bayesian network model and three generative mixture models.

4.1.1 Bayesian Network Model

A BN is a graphical representation that uncovers the probabilistic relationships among different variables. The strengths of employing a BN include handling missing data, learning causal relationships and avoiding over-fitted results [67]. Figure 4.1 illustrates the BN representation of one possible composition of 4 services. This BN portrays the composition of popular real services such as booking flights, booking hotels, car rentals, and insurance. Thus, WebService5 provides both flights and hotels booking services while WebService6 is composed of the car rental and insurance services. WebService7 is the service that encompasses all the aforementioned services.

The BN structure reveals the relationships between the constituent nodes (services). Each edge

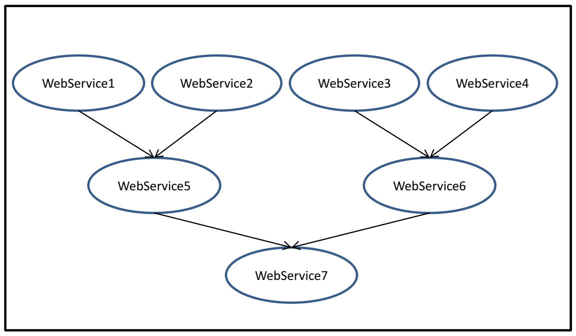


Figure 4.1: Bayesian network representation of an example of a composite web service.

directed from one node to another states that the former is a parent of the latter. The main strength of BNs consists of encoding the conditional independence between the variables represented by its nodes. To illustrate, each of the nodes in Figure 4.1 represents a variable attributed to the QoS of the corresponding web service. The conditional independence depicted by this network states that WebService7 (which represents the QoS of the consumer's interaction with WebService7) depends on WebService5 and WebService6. It also states that WebService7 is independent of the other web services given the state of its parents (WebService5 and WebService6).

Composition structure

Given the dynamic aspect of the composition process, the arrangement of the services in a composition is usually unknown. Therefore, consumers mostly encounter composite services with unknown structure. Since a composite service is represented as a BN, learning the structure of the latter uncovers that of the former. The challenge can be restated as follows: given a set of observations (vectors of counts) of the quality of a composite web service and its constituents (if available), what is the structure of the BN that represents this composite service? This question can be solved by various methods depending on the observations at hand. If the observations are complete (the quality of all web services are observed), the BN structure can be learned using various algorithms including the one derived from the maximum weight spanning tree (MWST) [68], K2 [69] and Markov Chain Monte Carlo [70] algorithms. In case of partial observation, the structural-Expectation Maximization (structural-EM) [71] and MWST-EM [72] algorithms learn the BN structure coping with incomplete data.

Complete observations

We chose the K2 algorithm to learn the structure of the BN in the case of complete data. K2 is a Bayesian method that searches heuristically for the structure with the maximum likelihood given a dataset. It begins with the assumption that all structures are equally likely in which case the joint probability of a given structure S and data D is given by [69]:

$$P(S, D) = P(S) \prod_{i=1}^{n} \prod_{j=1}^{pa_i} \frac{(r_i - 1)!}{(N_{ij} + K - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{4.1}$$

where $P(S)$ is the prior probability for every possible structure S. r_i is the maximum number of values a variable (which denotes a web service) in the network can be assigned to. In this case, all the network's variables can be given up to the maximum number of quality classes. For instance, if the number of quality classes is K, then r_i can be replaced by K. pa_i is the number of configurations of the parents of the i^{th} variable. N_{ijk} represents the number of times the i^{th} variable is assigned to the k^{th} quality class given the j^{th} configuration of i's parents. $N_{ij} = \sum_{k=1}^{r_i} n_{ijk}$ which is the number of times the i_{th} variable is assigned to any of the r_i quality classes given the j^{th} configuration of i's parents. To maximize Equation 4.1, it is sufficient to maximize the second inner product [69] as follows:

$$max(P(S, D)) = P(S) \prod_{i=1}^{n} max \left[\prod_{j=1}^{pa_i} \frac{(r_i - 1)!}{(N_{ij} + K - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \right] \tag{4.2}$$

To solve Equation 4.2, [69] used a greedy algorithm to search for the maximum $P(S, D)$. It assumes, a priori, an ordering on the variables such that a node is only allowed to be a parent of lower-ordered nodes [72]. The search starts with a node having no parents, then it adds the parent that increases the probability of the structure (the score given by the second inner product of equation 4.2). The algorithm converges when no addition of a parent increases that score.

Partial observations

In real interactions with composite web services, the consumers can rarely assess the QoS of its underlying services. Therefore, it is critical to account for these cases in which consumers partially observe the quality of the services taking part in a composition. The commonly used method to learn the BN's structure in case of incomplete data is known as the structural EM introduced in [71]. It applies the standard Expectation Maximization (EM) algorithm introduced in [73] to optimize parameters and then searches for the best structure that fits the data. Thus, at every step, the algorithm either optimizes the parameters of a current structure or selects a new structure, hence the name structural EM.

After defining the network's structure, the Bayesian analysis stipulates the network's parameters which consist, in the case of directed graphs, of the local probabilities at each node. Each variable in the network can take one value of the vector $[1, \ldots, K]$, where K is the number of quality classes. By combining the structure with the parameters of the network we can compute the joint probability distribution of the network variables using the chain rule that is given by:

$$p(\mathcal{X}) = \prod_{i=1}^{n} p(x_i|pa_i)$$

where $\mathcal{X}$ is the set of variables (nodes), x_i is the i^{th} node, and pa_i denotes the parents of x_i.

One of the main advantages of using BNs is the reduction of the number of parameters needed to characterize the joint probability distribution of a set of variables. For example, having seven variables that can take up to k values (in this case, k is the number of considered quality classes), their joint probability distribution is defined by $k^7 - 1$ parameters. However, using the BN in Figure 4.1, this number is reduced to $4(k-1) + 3(k-1)k^2$. The first term $4(k-1)$ corresponds to the probabilities of the single web services $P(WS1), P(WS2), P(WS3)$, and $P(WS4)$ (WS refers to WebService). The second term $3(k-1)k^2$ corresponds to the conditional probabilities of the composite web services $(P(WS5|WS1, WS2)$, $P(WS6|WS3, WS4)$, and $P(WS7|WS5, WS6)$. To further exemplify, if $k = 4$, the joint probability distribution requires $4^7 - 1 = 16383$ parameters while this number drops to $4(4-1) + 3(4-1)4^2 = 156$ parameters using the BN. In the QoS context, the parameters of the BN represent the conditional probabilities of the corresponding composite service QoS to belong to each of the k classes given the class its constituent services.

Parameters learning

Dirichlet Prior As mentioned earlier, we assume the network variables to be generated from a K-dimensional multinomial distribution. To compute this distribution's parameters we use the Bayesian inference to smooth the estimates and avoid poor results by adding a Dirichlet prior over the BN model parameters. A convenient way of assessing the Dirichlet prior is using the equivalent sample size. The basis of this method rely on the fact that the mean of the parameters of all variables is equal to the user's prior probability for these variables [74]. For more details about the equivalent sample size method, we refer the reader to [74] and [75]. Therefore,

$$P(\theta_{x_i|pa_i}) = \prod_{x_i} \theta_{x_i|pa_i}^{\alpha_{x_i,pa_i}-1}, \tag{4.3}$$

where $\alpha_{x_i,pa_i} = S\, q_{x_i|pa_i}$, $S > 0$ is the equivalent sample size which determines the user's confidence in his assessment of the belief network $q_{x_i|pa_i}$. In [75], q was chosen to be uniform which gives:

$$\alpha_{x_i,pa_i} = \frac{S}{|x_i||pa_i|}. \tag{4.4}$$

We are also aware that in service-oriented environments, the consumer is not aware of the quality of all web services included in a composite web service. In other words, the quality of some web services is not observed. Hence, we extend our model to learn from incomplete data. The EM algorithm is the benchmark technique suited to handle problems of this sort. First introduced in [73], the EM algorithm iterates between two steps: an expectation (E-step) and a maximization step (M-step). In the former, the missing variables are expected with respect to the current estimates of the parameters and observed variables. The latter follows and provides new estimates of the parameters given the full data (observed and expected). Various inference techniques have been suggested in the literature to solve the E-step such as the variable elimination [76] and junction tree [77] techniques. Thus, we use the standard variable elimination (VE) algorithm which consists of marginalizing out irrelevant variables from a list of factors one at a time. For instance, considering the network in Figure 4.1, the VE algorithm starts by choosing a variable ordering. For simplicity, we will use WS to replace WebService in the variables' names in the remaining of this section. Therefore, an ordering example might be $(WS1, WS2, WS5, WS3, WS4, WS6, WS7)$. Subsequently, the factors $p(WS1)$, $p(WS2)$, $p(WS5|WS1, WS2)$, $p(WS3)$, $p(WS4)$, $p(WS6|WS3, WS4)$, and $P(WS7|WS5, WS6)$ are initialized, as well as the evidence and query. Then, the variables are eliminated in the order chosen. To eliminate $WS1$, we first combine all the factors that contain $WS1$ by computing the product factor $P(WS5|WS1, WS2) * P(WS1)$. Then, we marginalize away the variable by summing over its possible values as follows $m(WS1) = \sum_{WS1} p(WS1)$ and the resulting factor will be added to the distribution. The other variables will be eliminated in the same fashion.

Beta-Liouville Prior The Dirichlet prior has been widely used as the most appropriate prior in Bayesian network learning. This is based on the fact that the Dirichlet distribution is closed under multinomial sampling. The Liouville distribution of second kind has been shown to be a generalization of the Dirichlet distribution. Like the Dirichlet, the BLD has been used in Bayesian analysis of count data. However, it hasn't been applied as a prior in Bayesian networks learning. Therefore, we extend the above BN model by considering the Beta-Liouville distribution as another prior to the multinomial for learning the network's parameters. We use the proposed BN to build a classifier for the QoS of composite web services. This classifier serves as an assessment of the quality of the collaboration among web services. The results of this classification will guide the web service composition task as follows. If the QoS was classified in a low quality class, then we know that one or more of the collaborating services is providing a low quality service. Consequently, we can either try a different combination of collaborating web services, or identify the responsibility of each web service in the quality of the composite service. In both cases, the improvement of the quality of composite services can be achieved.

It is noteworthy to mention that the integration of a new prior distribution is challenging on both conceptual and technical levels. First, on a conceptual level, the derived distribution has to logically fit the type of data being handled and be more advantageous than the replaced distribution. For instance, when using generalized Dirichlet in the place of Dirichlet the order of the variables in the distribution becomes to matter and the number of parameters is bigger which gives it more degrees of freedom to adjust the distribution. This flexibility in fitting several applications requires the derivation of a new set of equations to estimate the parameters of the new distribution (in this case, the multinomial generalized Dirichlet). The same applies to the Beta-Liouville prior especially when being introduced in the Bayesian network context. Learning the parameters of the Bayesian network is not as straightforward as learning the parameters of the distribution using the Bayesian inference method. The structure of the network, the number and type of variables need to be taken into consideration. While Dirichlet has $d + 1$ parameters, the Beta-Liouville has one extra parameter which is introduced by the Beta function employed as part of the distribution. Learning the new parameters also required new equations to be added to the learning algorithm of the Bayesian network parameters. The details of these models will be thoroughly discussed throughout the subsequent sections.

In dimension K, the Beta-Liouville has $K + 2$ parameters $\vec{\alpha} = (\alpha_1, \ldots, \alpha_K, \alpha, \beta)$. Given a vector of proportions $\vec{X} = (X_1, \ldots, X_{K+1})$, α_i can be viewed as a pseudo-count for $i = 1, \ldots, K$. Then, α, β are the parameters of the Beta distribution which is, in this case, the generating function of the Liouville distribution. The details of the methods used for assessing Dirichlet priors are available in [74]. Moreover, various methods for assessing Beta priors have been reviewed in [78]. To assess the Beta-Liouville prior, we use the equivalent sample size method. We compute the parameters $\alpha_1, \ldots, \alpha_K$ using the same equation used in the Dirichlet case, $\alpha_{x_i|pa_i} = S\, p((x_i|pa_i) = j)$. We choose a uniformly distributed prior belief network $p((x_i|pa_i) = j)$ [75], from which we obtain:

$$\alpha_{x_i|pa_i} = \frac{S}{C(pa_i)}, \tag{4.5}$$

where $C(pa_i)$ is the cardinality of the Cartesian product of the variables contained in the set of parents for each x_i. Then, the parameters of the Beta distribution are estimated by computing their maximum likelihood given the sum of $\alpha_1, \ldots, \alpha_K$ parameters for each variable x_i given by $\sum_{j=1}^{K} \alpha_{x_{ij}|pa_{ij}}$.

4.1.2 Online Bayesian Network

An additional challenge in evaluating the QoS-based trustworthiness of web services is its continuously changing quality. For instance, non-functional QoS properties such as the response time and availability may differ depending on the time the web services are called. Business related web services are generally busier during the working hours while social web services tend to be more occupied afterwards. This raises the need to be able to assess the trustworthiness of a web service at any point in time and after every interaction. This capability necessitates the construction of an online trust system to allow the updating of the QoS after each observation. In the BN context, this requirement is known as the online learning of the networks parameters. Therefore, we propose two online models; the first is based on a discounting window and the second employs an extension of the EM algorithm called Voting EM [79].

Discounting Window Model

The online approach we propose consists of updating the BN parameters after each observation. It starts with a known BN structure S, a set of discrete variables (nodes) $\boldsymbol{X}$, and a predefined number of values each variable in $\boldsymbol{X}$ can be assigned to, which in this context, is the number of QoS classes. Each variable in the network is then associated to a multinomial conditional probability distribution. It is common, in the BN literature, to add a Dirichlet prior to this distribution. Instead, our model substitutes the Dirichlet with the generalized Dirichlet distribution. Thus, the problem becomes the estimation of MGDD parameters which, in this case, represent the conditional probability tables (CPT) entries of the BN. Each entry in the CPT, denoted by θ_{ijk}, is the conditional probability of a variable x_i being equal to the QoS class k, given that the parents of x_i, pa_i are observed in their j^{th} configuration. Then, the network's parameters are updated according to the following rule:

```
Algorithm 1: Online learning of BN parameters
    Input: Empty dataset $\mathcal{D}$, a partially observed data instance $\vec{d}$, BN structure $S$ with a set of
    nodes $\boldsymbol{X}$, initial parameters $\vec{\Theta}$, window size $w$
    Add: generalized Dirichlet prior to each variable in $Z$
    if $size(\mathcal{D}) \pmod{w} > 0$ then
        Add $d$ to $\mathcal{D}$
    else
        $\mathcal{D} = d$
    end if
    for every variable $x_i$ in $\mathcal{X}$ do
        E-Step: compute the expectations of hidden variables
            $$p(x_i^k, pa_i^j|\mathcal{D}) = \frac{1}{N}\sum_{n=1}^{N} p(x_i^k, pa_i^j)$$
        M-Step: $\Theta_{new} = argmax\, p(\Theta_{old}|\mathcal{D})$
            $$\theta_{ijk} = \frac{p(x_i^k, pa_i^j|\mathcal{D})}{p(pa_i^j|\mathcal{D})}$$
        Update: the parameters of the generalized Dirichlet prior of $x_i$
            $$\alpha_k = \alpha_k + X_{ik}$$
            $$\beta_k = \beta_k + X_{ik+1} + \cdots + X_{iK+1}$$
    end for
```

$$\Theta_{new} = argmax\, p(\Theta_{old}|\mathcal{D})$$

where $\mathcal{D}$ is set of data instances $\mathcal{D} = d_n, \ldots, d_N$, Θ_{old} and Θ_{new} are the current and updated parameters respectively.

To account for the dynamic nature of the QoS, we employ an updated version of discounting window notion used in [80]. The intuition of the discounting window is to consider as many data cases as the size of the window can accommodates. For instance, a window of size 10 will store in $\mathcal{D}$ up to the 10^{th} incoming data cases. In [80], once the 11^{th} data case arrives, it replaces the oldest one stored in $\mathcal{D}$. However, in our version of the discounting window, upon arrival of the 11^{th} data case, $\mathcal{D}$ and the window will be reset allowing the latter to starts growing again to hold new 10 instances. After receiving a data case $\vec{d_i} = [d_{ik}, \ldots, d_{iK+1}]$ the BN parameters are updated using the EM update rule mentioned earlier. Then, the parameters $(\alpha_1, \ldots, \alpha_K, \beta_1, \ldots, \beta_K)$ of the generalized Dirichlet priors of each network variable are updated according to the following:

$$\alpha_k = \alpha_k + X_{ik}$$

$$\beta_k = \beta_k + X_{ik+1} + \cdots + X_{iK+1}$$

where X_{ik} is the number of times the i^{th} variable had the k^{th} value. $K + 1$ is the total number of values x_i can have (in this case, this is the total number of QoS classes). When the value of x_i is not observed, it is then replaced by its expected value given by:

$$p(x_i^k, pa_i^j | \mathcal{D}) = \frac{1}{N} \sum_{n=1}^{N} p(x_i^k, pa_i^j)$$

The steps of our online learning algorithm of the BN parameters are summarized in Algorithm 1.

Voting-EM Model:

The Voting-EM algorithm [79] is implemented as an extension to the EM(η) algorithm proposed by Bauer et al. in [81]. Voting EM depends on η, a learning rate that can be either fixed or dynamic. This learning rate controls the extent to which we consider the past observations in the learning process. In other words, past observations count less when η is close to 1. However, they count more when η is at the proximity of 0 [79]. The learning of BN parameters consists of learning the conditional probability tables (CPTs) entries given a current assignment of the $\bar{\theta}$ parameters and a set of data cases. In online learning, the set of data cases (D) is replaced by one data case which is an assignment of values to the network's variables. Therefore, the goal is to build a new parameters model $\tilde{\theta}$ based on the old model $\bar{\theta}$ and a data case. In the EM(η) algorithm and assuming a batch learning, $\tilde{\theta}$ is computed by the maximization of the following function:

$$F(\tilde{\theta}) = \eta * L_D(\tilde{\theta}) - d(\tilde{\theta}, \bar{\theta})$$

where $L_D(\tilde{\theta})$ is the normalized log likelihood of D in $\tilde{\theta}$ and is given by:

$$L_D(\tilde{\theta}) = \frac{1}{N}\sum_{j=1}^{N} \log P_{\tilde{\theta}}(y_j)$$

$d(\tilde{\theta}, \bar{\theta})$ represents the distance between the old and new models. Various distance functions have been proposed including the Chi squared, $\mathcal{L}_2$-norm, and relative entropy [81]. To derive the Voting EM maximization equation, [79] used the Chi squared distance function. As mentioned earlier, the Voting EM is a natural extension of the EM(η) to handle the online learning. In case of complete data, the update rule of the network's parameters that maximizes $F(\tilde{\theta})$ is the following:

$$\theta^{t}_{ijk} = \begin{cases} \theta^{t-1}_{ijk} + \eta + (1-\eta)\theta^{t-1}_{ijk} & \text{for } P(pa^j_i|y_t) = 1 \ \& \\ & P(x^k_i|y_t) = 1, \\ (1-\eta)\theta^{t-1}_{ijk} & \text{for } P(pa^j_i|y_t) = 1 \ \& \\ & P(x^k_i|y_t) = 0 \\ \theta^{t-1}_{ijk} & \text{otherwise.} \end{cases}$$

More details about the Voting EM online learning rules and its convergence properties are available in [79].

Afterwards, we can use the learned network parameters (CPTs) to calculate the probabilities of the composite web service to belong to each of the quality classes. For instance, suppose that we consider the four quality classes from Section III and a BN with WSC composed of WSA and WSB. Then, each of WSA and WSB contains 4 entries while WSC contains 64, each of which represents the probability of the QoS of WSC to belong to one of the 4 quality classes given one of the 16 different configurations of its parents' QoS classes. Thus, we end up with 4 probabilities which provide a descriptive representation of the quality of the composite web service. Despite the eminence of this multi-dimensional quality representation, it is of equal significance to assign each web service a uni-dimensional trust score. This provides a straightforward means to compare the quality of web services and a simplistic approach to assist the web service's selection process. Consequently, this raises the need for a function to compute the trustworthiness of a web service after each interaction. This function aims to satisfy the following desirable properties:

1. The trust function should be continuous.
2. It should increase with the increasing probabilities of the web service to belong to high quality classes. In other words, this function should increase when the BN parameters (CPTs) associated to high quality classes increase.
3. However, it should decrease with the increasing probabilities of the web service to belong to low quality classes. In other words, it should decrease when the BN parameters (CPTs) associated to low quality classes increase.

The proposed trust function is given by:

$$Trust = \sum_{i=1}^{\lceil \frac{C}{2} \rceil} w'_i p_i - \sum_{i=\lceil \frac{C}{2} \rceil + 1}^{C} w'_i p_i \tag{4.6}$$

where $w'_i = \frac{w_i}{\sum_{i=1}^{C} w_i}$ which normalizes the weights and trust values to the $[0, 1]$ interval. w_i is the weight assigned to the i^{th} good quality class ($1 \leq i \leq \lceil \frac{C}{2} \rceil$) or the i^{th} bad quality class ($\lceil \frac{C}{2} \rceil < i \leq C$). p_i is the probability of the QoS to belong to the i^{th} quality class. 1 and C represent the highest and lowest quality classes, respectively. It is worthy to note that we chose $\lceil \frac{C}{2} \rceil$ to be the divider between good and bad quality classes. The value of this divider can be increased or decreased to make the trust function looser or stricter, respectively. For example, a slightly stricter trust computation can be achieved by substituting $\lceil \frac{C}{2} \rceil$ for $\lfloor \frac{C}{2} \rfloor$. The assigned weights will be fixed during the same experiment and vary depending on the types of agents and services involved. For instance, the truster agent might be very conservative and willing to interact with the service having the highest quality. This might be the case with data critical services dealing with personal and/or financial information. In this case, the probabilities associated with the low-quality classes will be given relatively high weights to penalize the overall trust values of the web service. In our experiments, we consider a rather lenient environment where the probabilities associated with the high quality classes are assigned higher weights than the ones associated with the low quality classes. We can easily prove that this function satisfies the desirable properties by showing that its partial derivatives with respect to the weights of good classes and bad classes are positive and negative, respectively.

4.1.3 Generative Mixture Models

In this section, we apply three mixture models to compute the trustworthiness of composite services. A mixture model with M components is defined as:

$$p(\vec{X}|\Theta) = \sum_{j=1}^{M} p(\vec{X}|j, \vec{\alpha}_j)P(j), \tag{4.7}$$

where $P(j)$ are the mixing coefficients (constrained by $0 < P(j) < 1$ and $\sum_{j=1}^{M} P(j) = 1$). Θ refers to the entire set of parameters to be estimated, $\Theta = (\vec{\alpha}_1, ..., \vec{\alpha}_M, P(1), ..., P(M))$. The proposed models are based on three assumptions in which $p(\vec{X}|j, \vec{\alpha}_j)$ is either a ***MDD***, ***MGDD***, or ***MBLD***. The learning of the mixtures for each of these assumptions have been proposed in [82], [83], and [38] respectively. As an example, we describe the learning algorithm for the mixture of MDD. This algorithm starts by initializing the model parameters using the fuzzy-C Means (FCM) and method of moments (MM). After the initialization steps, the algorithm iterates, until convergence, over the steps of the EM algorithm. During the E-step, the posteriori probabilities and the complete-data log-likelihood are computed. In the M-step, the parameter estimates are updated according to Equation 4.9. The algorithm converges when the variance of the conditional expectation of the complete-data log-likelihood between two successive iterations is less than a threshold. Algorithm 2 shows the steps for learning the parameters of a mixture of MDD. Similar steps may be followed to learn the parameters of mixtures of MGDD and MBLD.

4.2 Experimental Results

In order to evaluate the models presented in this proposal, we assume the following scenario: A consumer or a software agent is willing to interact with a composite service (*CS*) which is composite of two services *A* and *B*. The proposed models can very well scale to a larger number of web services; we only used two web services for simplicity reasons. Furthermore, we adopt the four quality class model used, for instance, in the QWS data set [84]. The QWS data set classifies each web service in one of the following four classes; Platinum, Gold, Silver, and Bronze. The Platinum class represents the highest quality a web service might be assigned to and the Bronze

Algorithm 2: MMDD Estimation

```
Input: k-dimensional data $\vec{X}_i, i = 1, ..., N$ and a number of components M
Apply: the Fuzzy C-means to obtain the elements, covariance matrix and mean of each component
Apply: the Method of Moments for each component $j$ to obtain the vector of parameters $\vec{\alpha_j}$
Assign: data to clusters
$Converged \leftarrow false$
while $!Converged$ do
    E-Step: compute the posteriori probabilities
        $p^{(t)}(j|\vec{X}_i, \vec{\alpha_j}) \leftarrow \frac{p^{(t-1)}(\vec{X}_i|j, \vec{\alpha_j})P^{(t)}(j)}{\sum_{j=1}^{M} p^{(t-1)}(\vec{X}_i|j, \vec{\alpha_j})P(j)}$
        $llh^{(t)} \leftarrow \sum_{i=1}^{N} log(\sum_{j=1}^{M} p^{(t)}(\vec{X}_i|j, \vec{\alpha_j})P^{(t)}(j))$
    M-Step: $\hat{\Theta}^{(t)} \leftarrow argmax\, llh^{(t-1)}$
        $\vec{\alpha_j}^{(t)} \leftarrow \vec{\alpha_j}^{(t-1)} - \epsilon \frac{\delta}{\delta \vec{\alpha_j}} llh^{(t-1)}$
        $P^{(t)}(j) = \frac{\sum_{i=1}^{N} p^{t-1}(j|\vec{X}_i, \vec{\alpha}_j)}{N}$
    $test \leftarrow llh^{(t)} - llh^{(t-1)}$
    if $test < \epsilon$ then
        $Converged \leftarrow true$
    end if
end while
```

represents the lowest. To the best of our knowledge, there is no data set available that captures the ranking of a web service at different times with different load inputs. Therefore, we monitored real web services available online at different times for three days with one hundred different load test cases.

4.2.1 Bayesian Network Experiments

Parameters Learning

The first experiment aims to evaluate the performance of learning the BNs parameters using the proposed approach. Therefore, we consider a simple network that consists of three nodes each of which represents the quality of one of the three web services *A*, *B*, and CS. *A* and *B* are the parents nodes of *CS*. For the purpose of these experiments, *A* and *B* represent two real web services that provide currency conversion and IP address resolution respectively. Then, the quality values of *CS* are computed by a composition function. We will consider the composition functions used in [18] such as *Switch*, Sum, Product, *Min*, and *Max*. For instance, *Switch* computes the value of the composite quality by choosing the value of one of its parents based on a predefined multinomial distribution. Sum generates the composite quality value by adding the quality values of the corresponding children. Product computes the composite quality value by multiplying the quality values of the corresponding children. *Min* assigns the composite service a quality value inherited from the child with the lowest quality. *Max* assigns the composite service a quality value inherited from the child with the highest quality. In the case of *Switch*, we initialize a multinomial distribution with the parameters $[0.8\ 0.2]$. In other words, the quality values of *CS* will be chosen 80% from *A* and 20% from *B*. At each time step, the conditional probabilities of the Bayesian network are computed using the approach described earlier. To illustrate, below are the conditional probabilities tables (CPTs) of each variable in the network (*A*, *B*, and *CS*) after 10 observations where "Timestep" refers to the number of observations.

P(A=1)	P(A=2)	P(A=1)	P(A=4)
0.1710	0.4868	0.0836	0.2585

Table 4.1: CPT of constituent service *A*.

P(B=1)	P(B=2)	P(B=1)	P(B=4)
0.1775	0.1775	0.1769	0.4680

Table 4.2: CPT of constituent service *B*.

We run the same experiments with different priors; Dirichlet, generalized Dirichlet and Beta Liouville as discussed earlier. To evaluate the performance of the Bayesian model in estimating the trustworthiness of composite services, we compare it to the multinomial Naive Bayes approach [85]. This approach naively assumes that the qualities of individual services are independent given the class quality of the composite service. The parameters of the Naive Bayes are the class priors

		B=1	B=2	B=3	B=4
P(CS=1)	A=1	0.2207	0.2199	0.2209	0.1693
	A=2	0.1370	0.1371	0.1379	0.0211
	A=3	0.2370	0.2369	0.2370	0.2112
	A=4	0.2035	0.2037	0.2183	0.1416
P(CS=2)	A=1	0.2207	0.2195	0.2208	0.1693
	A=2	0.3533	0.3392	0.3744	0.9298
	A=3	0.2370	0.2369	0.2370	0.2112
	A=4	0.2035	0.2037	0.2103	0.1576
P(CS=3)	A=1	0.2207	0.2189	0.2209	0.1684
	A=2	0.1385	0.1377	0.1379	0.0211
	A=3	0.2370	0.2369	0.2370	0.2112
	A=4	0.2035	0.2037	0.2182	0.1682
P(CS=4)	A=1	0.3379	0.3416	0.3374	0.4930
	A=2	0.3712	0.3860	0.3497	0.0281
	A=3	0.2891	0.2894	0.2889	0.3665
	A=4	0.3896	0.3889	0.3533	0.5327

Table 4.3: CPT of composite service *CS*

of the composite service and the conditional probabilities of the counts of the quality class of *A* and *B*, given the quality class of *CS*. The class priors are computed from the actual data by: $prior(C = j) = \frac{N_{C_j}}{N}$ where N_{C_j} is the number of observations that belong to class C_j and N is the total number of observations. The conditional probabilities for each of the four classes are given by $\hat{\theta}_{y_i} = \frac{N_{y_i}+\alpha}{N_y+\alpha n_y}$ where N_{y_i} is the number of times the i^{th} feature appear in an observation that belongs to class y. n_y is the total number of times all features appear for class y. α is a smoothing factor that we set to be equal to 1 (this is called Laplace smoothing), and n is the total number of features.

After learning the network parameters (probabilities of *CS* belonging to each of the four quality classes), we can use various weighted functions to compute the trust value of *CS*. The weights of the probability of each class will vary depending on the types of agents and services involved. To control the variance among different functions we set the range of the weights to the $[1\ \ 10]$ range. For instance, the truster agent might be very conservative and willing to interact with agents belonging to the Platinum class only. This might be the case with data critical services dealing with personal and/or financial information. In this case, the probabilities associated with the last two classes will be given relatively high weights to emphasize low quality outcomes of a given service. In our experiments, we assume a lenient environment where the probabilities associated with the first two classes are assigned higher weights than the ones associated with the last two.

The trust function we used is given by:

$$Trust = \frac{\sum_{i=1}^{C} w_i p_i}{\max W}, \tag{4.8}$$

where w_i is the weight associated with the i^{th} class and p_i is the probability of the service quality to belong to the i^{th} class. $\max W$ is the maximum weight that normalizes trust to be in the $[0\ 1]$ range.

Figures 4.2 and 4.3 show the estimated trust using Bayesian and naive Bayes approaches for the *Switch*, *Min* (Figure 4.3-left) and *Max* (Figure 4.3-right) composition functions respectively. For a total of 100 incomplete observations, the graph shows that with 50% of missing data, the Bayesian approach still learns the quality of the composite service *CS*. However, the Naive approach, not handling missing data well, shows a low learning accuracy of trust estimates. The EM algorithm employed in the Bayesian approach succeeds to learn from incomplete data. At each time step, the algorithm iteratively imputes missing values using exact inference and then uses the complete data (observed + imputed) to update the network parameters.

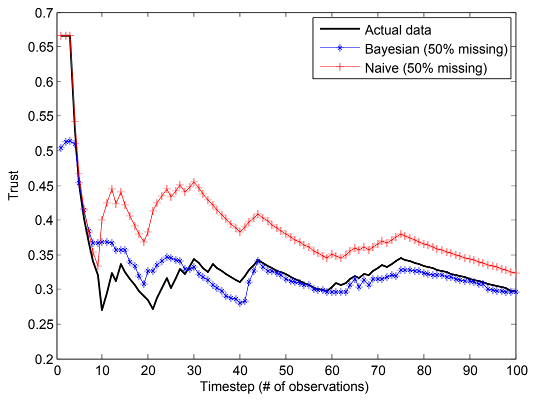
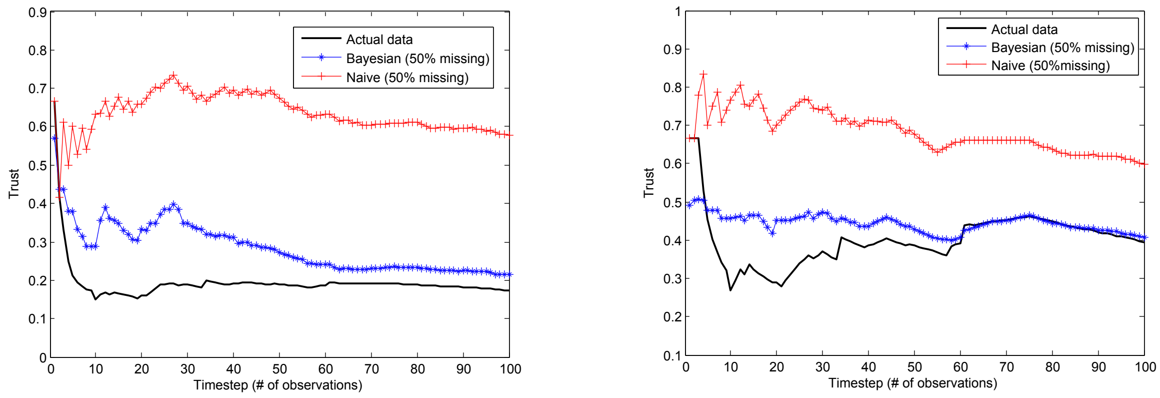


Figure 4.2: Bayesian (Dirichlet prior) vs. Naive Bayes trust estimation for *Switch* composition function

We also show in Figure 4.4 that the Bayesian approach can learn the conditional trust of a composite service based on one of the constituent services *A* and *B* for *Max* and *Min*. In this experiment, we observed that 61% of *CS* quality came from *A* in the case of *Max* and 61% from *B* in the case of *Min*. As is shown in Figure 4.4 (left), the conditional trust in *CS* given *A* is, for the



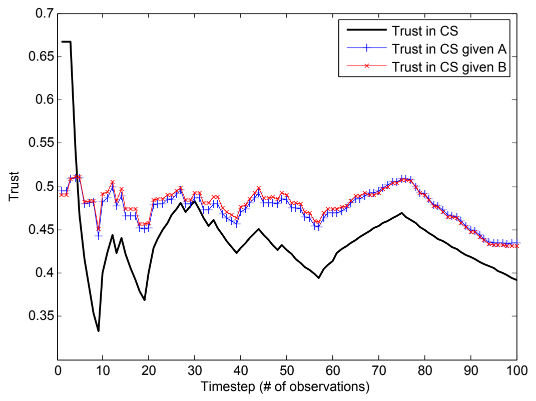
(a) *Min* composition function (b) *Max* composition function

Figure 4.3: Bayesian (Dirichlet prior) vs. Naive Bayes trust estimation

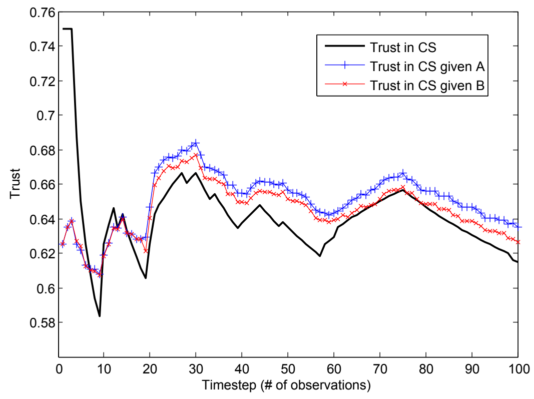
most part, higher than the trust and the conditional trust given *B*. However, the conditional trust given *B* is mostly higher than both the trust and the conditional trust given *A* in the case of *Min* figure 4.4 (right). Similar experiments were employed with a Beta-Liouville prior for which the results are shown in figures 4.5 and 4.6.

Quality of Service Classification

We will use the BN structure from Figure 4.1 to run our classification experiments. Since real composite services available in the industry do not often involve a large number of services, we believe that experimenting with 7 services is pragmatic. We assume the following scenario: A consumer is interacting with the composite web service $WS7$ and is interested in learning about its trustworthiness. After every interaction, the consumer is capable of observing the QoS of this web service which can belong to one of the four classes described earlier. It is also noteworthy to mention that the consumer is not aware of the quality of each of the constituent web services of $WS7$. This raises the necessity to deal with missing data and capture the responsibility of all web services in the overall quality of the composite web service. As we discussed in the previous section, we will apply the Bayesian approach with a Beta-Liouville priori distribution. For the purpose of our experiments, we used a synthetic data set that is sampled and generated as follows: The quality classes of the four single web services are sampled from different multinomial



(a) *Min* composition function



(b) *Max* composition function

Figure 4.4: Conditional trust estimation based on constituent services *A* and *B*

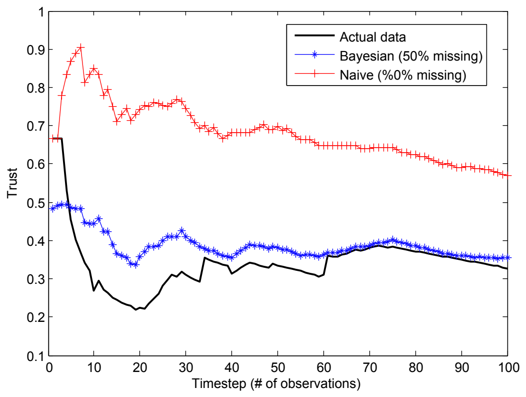
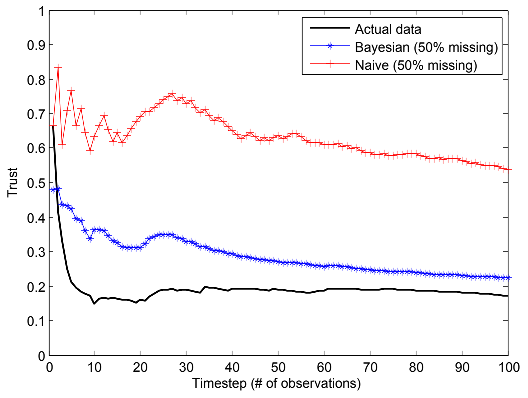


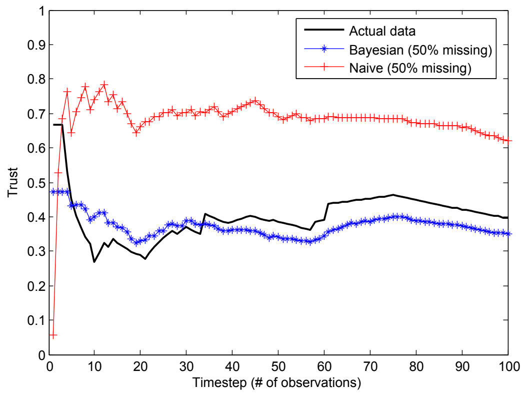
Figure 4.5: Bayesian (Beta Liouville prior) vs. Naive Bayes trust estimation for SWITCH composition function

distributions with the parameters exposed in Table 4.4.

The first column asserts that the quality of $WS1$ will be assigned to the first class 40% of the times, to the second class 30% of the times, so on and so forth. The advantage of working with a synthetic dataset and knowing the distributions' parameters that generated it is two-fold. First, it allows us to evaluate the learned parameters using the Bayesian approach by comparing them to the initial ones from Table 4.4. Second, it awards us the ability to capture the responsibilities of the constituent web services in the quality of the composite web services.



(a) *Min* composition function



(b) *Max* composition function

Figure 4.6: Bayesian (BL prior) vs. Naive Bayes trust estimation

WS1	WS2	WS3	WS4
[0.4 0.3 0.1 0.2]	[0.1 0.1 0.4 0.4]	[0.3 0.3 0.2 0.2]	[0.2 0.2 0.3 0.3]

Table 4.4: Multinomials used to generate the classes of the four single web services.

After sampling the values of the single web services, the quality values of the three composite web services $WS5$, $WS6$, and $WS7$ are computed using one of the composition functions; $Switch$, Sum, $Product$, Min, and Max that are discussed in [18]. For instance, $Switch$ computes the value of the composed quality by choosing the value of one of its parents based on a predefined multinomial distribution. Sum generates the composite quality value by adding the quality values of the corresponding children. $Product$ computes the composite quality value by multiplying the quality values of the corresponding children. Min assigns the composite service a quality value inherited from the child with the lowest quality. Max assigns the composite service a quality value inherited from the child with the highest quality. To complete the data needed for training our classifiers, we will apply three Switch composition functions, one for each composite service. Therefore, we initialize three multinomial distributions with the parameters $[0.8\ 0.2]$, $[0.2\ 0.8]$, and $[0.7\ 0.3]$. Each of the previous distributions is used to choose the quality of the composite web services $WS5$, $WS6$, and $WS7$ respectively. In other words the quality values of $WS5$ will be chosen 80% from $WS1$ and 20% from $WS2$. The same concept applies to the other composite web services.

Classifier1 This classifier consists of applying the Bayesian approach with the Beta-Liouville distribution as a prior to the conditional probabilities of the BN variables. The experiment begins by the construction of the network and the initialization of the conditional probabilities. As we mentioned before, the only observed network variable is the composite web service $WS7$. The dataset which consists of 300 samples is then partitioned into two parts; a training set (200 samples) and a testing set (100 samples of which the values of $WS7$ are removed). Afterwards, the network's parameters are learned using the EM algorithm discussed earlier and the training data set. Subsequently, the class of each data sample in the testing set is predicted as follows. At first, the evidence (observed values) is added to the network. Then, the marginal on $WS7$ is computed which devotes the probabilities of the quality of $WS7$ being equal to each of the predefined four quality classes. Finally, the class with the highest probability is assigned to the corresponding data sample. The learned parameters of the four single web services are compared in the Table 4.5 to the initial parameters from Table 4.4. The conditional probabilities at $WS5$, $WS6$, and $WS7$ are displayed in the tables 4.6, 4.7, and 4.8, respectively.

WS1	0.4086	0.2646	0.1006	0.2262
WS2	0.0868	0.0868	0.4240	0.4024
WS3	0.2795	0.3442	0.1454	0.2309
WS4	0.2009	0.2409	0.2558	0.3024

Table 4.5: CPT of WS1, WS2, WS3 and WS4.

These results prove the efficiency of the Bayesian approach in learning the conditional probabilities corresponding to each variable despite that only the values of $WS7$ were observed and the rest were hidden.

Classifier 2 We compare the first classifier with another BN classifier that applies the Bayesian approach with a Dirichlet prior instead of the Beta-Liouville. This experiment follows the same steps as with the first classifier. Also, it uses the same dataset described earlier for both training and testing phases. The learned parameters of the four single web services are displayed in the

		WS2=1	WS2=2	WS2=3	WS2=4
P(WS5=1)	WS1=1	0.9583	0.5882	0.8362	0.7750
	WS1=2	0.0500	0.0357	0.0125	0.0104
	WS1=3	0.6661	0.0500	0.0208	0.0227
	WS1=4	0.0625	0.0625	0.0139	0.0132
P(WS5=2)	WS1=1	0.0139	0.3214	0.0086	0.0083
	WS1=2	0.8500	0.8929	0.7625	0.8021
	WS1=3	0.0833	0.0500	0.0208	0.0227
	WS1=4	0.0001	0.0625	0.0139	0.0132
P(WS5=3)	WS1=1	0.0139	0.0357	0.1466	0.0083
	WS1=2	0.0500	0.0357	0.2125	0.0104
	WS1=3	0.7500	0.8500	0.9375	0.6591
	WS1=4	0.0625	0.0625	0.2917	0.0132
P(WS5=4)	WS1=1	0.0139	0.0357	0.0086	0.2083
	WS1=2	0.0500	0.0357	0.0125	0.1771
	WS1=3	0.0833	0.0500	0.0208	0.2955
	WS1=4	0.8125	0.8125	0.6806	0.9605

Table 4.6: CPT of composite service WS5.

Table 4.9. The conditional probabilities assigned to $WS5$, $WS6$, and $WS7$ are displayed in the tables 4.10, 4.11, and 4.12, respectively.

The mean errors of the differences between the actual parameters and those estimated using the Bayesian approaches with Dirichlet and Beta-Liouville priors are compared in Figure 4.7. The mean errors of the estimated parameters with the Beta-Liouville prior are less than those of the estimated parameters with the Dirichlet prior except for the mean error of $P(WS = 1)$, where WS can be $WS1$, $WS2$, $WS3$, or $WS4$.

Classifier 3 To further evaluate the performance of the above classifiers, we compare them to the state of the art naive Bayes (NB) classifier. This approach naively assumes that the features are conditionally independent given the class labels. To evaluate the performance of the three classifiers, we display their confusion matrices in Table 4.13 which present the true classes (rows) versus the predicted ones (columns). Therefore, the numbers in the first diagonal show the correctly predicted classes. The classification rate of the NB classifier (66%) is shown to be smaller than

		WS4=1	WS4=2	WS4=3	WS4=4
P(WS6=1)	WS3=1	0.9998	0.5882	0.6874	0.9410
	WS3=2	0.5554	0.0001	0.0001	0.0001
	WS3=3	0.6661	0.0002	0.0001	0.0001
	WS3=4	0.2857	0.0001	0.0001	0.0001
P(WS6=2)	WS3=1	0.0001	0.4117	0.0001	0.0001
	WS3=2	0.4444	0.9998	0.5384	0.6362
	WS3=3	0.0003	0.3999	0.0001	0.0001
	WS3=4	0.0001	0.3000	0.2103	0.0001
P(WS6=3)	WS3=1	0.0001	0.0001	0.3125	0.0001
	WS3=2	0.0001	0.0001	0.4615	0.0001
	WS3=3	0.3332	0.5997	0.9998	0.5832
	WS3=4	0.0001	0.0001	0.4999	0.0001
P(WS6=4)	WS3=1	0.0001	0.0001	0.0001	0.0589
	WS3=2	0.0001	0.0001	0.0001	0.3636
	WS3=3	0.0003	0.0002	0.0001	0.4166
	WS3=4	0.7142	0.6998	0.4999	0.5327

Table 4.7: CPT of composite service WS6.

both of the classifier 1 and 2. Both BN classifiers with the Dirichlet and Beta-Liouville priors have approximately a similar performance with the classification rate of classifier 1 (80%) being slightly higher than that of classifier 2 (79%). This proves that the Beta-Liouville distribution can be another effective prior to the multivariate variables of BNs.

To further validate the high classification performance of our classifiers, we run an additional experiment with a different network structure (Figure 4.8). This network represents the composition of a more complex service with a larger number of constituents (more than double the services from the first experiment). The confusion matrices of the three classifiers are also reported in Table 4.14. These results show that the number of services does not compromise the classification performance of the classifiers. In contrast, the additional data available from the added services improved the classification accuracy of all classifiers.

Online Bayesian Network Experiments

Online parameters learning In the following experiment, we test the performance of the Voting EM online learning by considering the following scenario. Suppose there is a composite web

		WS6=1	WS6=2	WS6=3	WS6=4
P(WS7=1)	WS5=1	0.9942	0.0842	0.3330	0.1741
	WS5=2	0.9256	0.0019	0.0007	0.0009
	WS5=3	0.8702	0.0011	0.0014	0.0017
	WS5=4	0.7461	0.0038	0.0013	0.0011
P(WS7=2)	WS5=1	0.0019	0.9132	0.0010	0.0007
	WS5=2	0.0722	0.9942	0.1741	0.3526
	WS5=3	0.0019	0.8544	0.0014	0.0017
	WS5=4	0.0019	0.7423	0.0013	0.0011
P(WS7=3)	WS5=1	0.0019	0.0013	0.6649	0.0007
	WS5=2	0.0011	0.0019	0.8245	0.0009
	WS5=3	0.1260	0.1433	0.9958	0.2224
	WS5=4	0.0019	0.0038	0.9132	0.0011
P(WS7=4)	WS5=1	0.0019	0.0013	0.0010	0.8245
	WS5=2	0.0011	0.0019	0.0007	0.6456
	WS5=3	0.0019	0.0011	0.0014	0.7741
	WS5=4	0.2500	0.2500	0.0842	0.996

Table 4.8: CPT of composite service WS7.

WS1	0.3889	0.2593	0.1435	0.2083
WS2	0.1389	0.1065	0.3657	0.3889
WS3	0.2898	0.3097	0.2102	0.1903
WS4	0.1944	0.1944	0.3009	0.3102

Table 4.9: CPT of WS1, WS2, WS3 and WS4.

service C composed of the two web services A and B. The Bayesian network's nodes represent the quality classes of each of these web services. The quality classes (QoS ratings) are the same ones we used in the previous experiments. We set the initial parameters of the network as follows:

$$CPTA = [0.1\ 0.3\ 0.4\ 0.2]\quad CPTB = [0.5\ 0.1\ 0.2\ 0.2]$$

$$CPTC(:,:,1) = \begin{bmatrix} 0.13 & 0.01 & 0.41 & 0.45 \\ 0.20 & 0.40 & 0.32 & 0.08 \\ 0.29 & 0.15 & 0.18 & 0.38 \\ 0.19 & 0.29 & 0.08 & 0.44 \end{bmatrix}$$

		WS2=1	WS2=2	WS2=3	WS2=4
P(WS5=1)	WS1=1	0.9583	0.5882	0.8362	0.7750
	WS1=2	0.0500	0.0357	0.0125	0.0104
	WS1=3	0.6661	0.0500	0.0208	0.0227
	WS1=4	0.0625	0.0625	0.0139	0.0132
P(WS5=2)	WS1=1	0.0139	0.3214	0.0086	0.0083
	WS1=2	0.8500	0.8929	0.7625	0.8021
	WS1=3	0.0833	0.0500	0.0208	0.0227
	WS1=4	0.0001	0.0625	0.0139	0.0132
P(WS5=3)	WS1=1	0.0139	0.0357	0.1466	0.0083
	WS1=2	0.0500	0.0357	0.2125	0.0104
	WS1=3	0.7500	0.8500	0.9375	0.6591
	WS1=4	0.0625	0.0625	0.2917	0.0132
P(WS5=4)	WS1=1	0.0139	0.0357	0.0086	0.2083
	WS1=2	0.0500	0.0357	0.0125	0.1771
	WS1=3	0.0833	0.0500	0.0208	0.2955
	WS1=4	0.8125	0.8125	0.6806	0.9605

Table 4.10: CPT of composite service WS5.

$$CPTC(:,:,2) = \begin{bmatrix} 0.13 & 0.16 & 0.55 & 0.16 \\ 0.39 & 0.08 & 0.39 & 0.14 \\ 0.31 & 0.29 & 0.02 & 0.38 \\ 0.26 & 0.18 & 0.34 & 0.22 \end{bmatrix}$$

$$CPTC(:,:,3) = \begin{bmatrix} 0.06 & 0.34 & 0.22 & 0.38 \\ 0.11 & 0.29 & 0.22 & 0.38 \\ 0.28 & 0.25 & 0.18 & 0.29 \\ 0.25 & 0.17 & 0.03 & 0.55 \end{bmatrix}$$

$$CPTC(:,:,4) = \begin{bmatrix} 0.38 & 0.42 & 0.13 & 0.07 \\ 0.32 & 0.28 & 0.12 & 0.28 \\ 0.46 & 0.34 & 0.16 & 0.04 \\ 0.23 & 0.27 & 0.32 & 0.18 \end{bmatrix}$$

Afterwards, we generate 1000 samples from the constructed network. We also build another network with the same structure as the one above with randomly generated parameters. We run

		WS4=1	WS4=2	WS4=3	WS4=4
P(WS6=1)	WS3=1	0.9998	0.5882	0.6874	0.9410
	WS3=2	0.5554	0.0001	0.0001	0.0001
	WS3=3	0.6661	0.0002	0.0001	0.0001
	WS3=4	0.2857	0.0001	0.0001	0.0001
P(WS6=2)	WS3=1	0.0001	0.4117	0.0001	0.0001
	WS3=2	0.4444	0.9998	0.5384	0.6362
	WS3=3	0.0003	0.3999	0.0001	0.0001
	WS3=4	0.0001	0.3000	0.2103	0.0001
P(WS6=3)	WS3=1	0.0001	0.0001	0.3125	0.0001
	WS3=2	0.0001	0.0001	0.4615	0.0001
	WS3=3	0.3332	0.5997	0.9998	0.5832
	WS3=4	0.0001	0.0001	0.4999	0.0001
P(WS6=4)	WS3=1	0.0001	0.0001	0.0001	0.0589
	WS3=2	0.0001	0.0001	0.0001	0.3636
	WS3=3	0.0003	0.0002	0.0001	0.4166
	WS3=4	0.7142	0.6998	0.4999	0.5327

Table 4.11: CPT of composite service WS6.

the Voting EM algorithm using the complete 1000 data samples generated 100 times. Figure 5 compares the true parameter CPTC(3,3,3) which is equal to 0.18 to the estimated parameters using the online update rule (we show the average of the 100 runs). CPTC(3,3,3) is the conditional probability that web service C belongs to the quality class 3 given that both web services A and B belong to the same class 3. The straight line represents the true parameter while the thick and the dashed curves depict the estimated parameters with $\eta = 0.01$ and $\eta = 0.05$, respectively. This figure shows that the estimated parameters with the larger η tend to converge to the true one faster than the ones estimated with the smaller one. However, with the smaller η the estimates are less noisier.

Figure 6 displays the average of the estimated parameter from the 100 runs in function of the number of samples with error bars plotted at 10 different times. The error bars are specified by the standard deviation of the estimated parameters. It is also noteworthy to mention that the variance of the estimated parameters decreases when the number of sample increases. The error bars are longer with the smaller η than with the larger one. We also used the unpaired two-samples t-test to verify the statistical significance to our results. The test revealed an $h = 0$ which indicates that

		WS6=1	WS6=2	WS6=3	WS6=4
P(WS7=1)	WS5=1	0.9942	0.0842	0.3330	0.1741
	WS5=2	0.9256	0.0019	0.0007	0.0009
	WS5=3	0.8702	0.0011	0.0014	0.0017
	WS5=4	0.7461	0.0038	0.0013	0.0011
P(WS7=2)	WS5=1	0.0019	0.9132	0.0010	0.0007
	WS5=2	0.0722	0.9942	0.1741	0.3526
	WS5=3	0.0019	0.8544	0.0014	0.0017
	WS5=4	0.0019	0.7423	0.0013	0.0011
P(WS7=3)	WS5=1	0.0019	0.0013	0.6649	0.0007
	WS5=2	0.0011	0.0019	0.8245	0.0009
	WS5=3	0.1260	0.1433	0.9958	0.2224
	WS5=4	0.0019	0.0038	0.9132	0.0011
P(WS7=4)	WS5=1	0.0019	0.0013	0.0010	0.8245
	WS5=2	0.0011	0.0019	0.0007	0.6456
	WS5=3	0.0019	0.0011	0.0014	0.7741
	WS5=4	0.2500	0.2500	0.0842	0.996

Table 4.12: CPT of composite service WS7.

30	3	0	0
0	20	2	3
4	3	9	2
2	1	0	21

30	1	0	2
2	18	2	4
4	1	11	1
1	2	1	20

25	7	0	3
3	14	5	4
0	5	7	3
3	0	1	20

Table 4.13: Confusion Matrices: classifier1 (left) 82%, classifier2 (middle) 84%, classifier3 (right) 77%

the the null hypothesis (estimated and actual parameters have the same mean) cannot be rejected at 5% significance level. The test statistic value returned is 0.6274 with a standard deviation equal to 0.0296.

We further test the online algorithm by running the same simulation (100 runs and 1000 samples) with two larger networks that consists of 8 and 15 variables (nodes). The new networks represent two composite services that are composed of more services. Aligned with the travel example illustrated in Figure 2, the added variables represent the following services; booking hostels, renting apartments, booking cruises, reserving activities, and booking vacation packages. The error bars of the two estimated parameters using both networks are displayed in Figure 7, 8, 9 and 10. These figures confirm that even with a larger number of services, the estimated parameters still

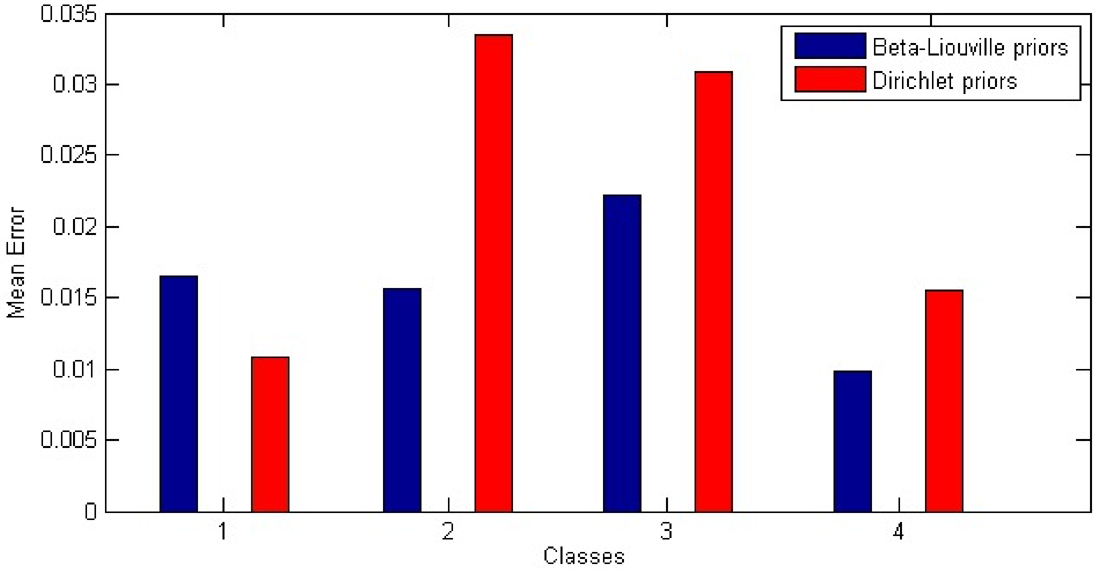


Figure 4.7: Mean Error of parameters estimation.

31	0	0	0
0	126	0	0
1	2	13	0
1	5	9	12

31	0	0	0
0	26	0	0
1	2	11	2
1	5	5	16

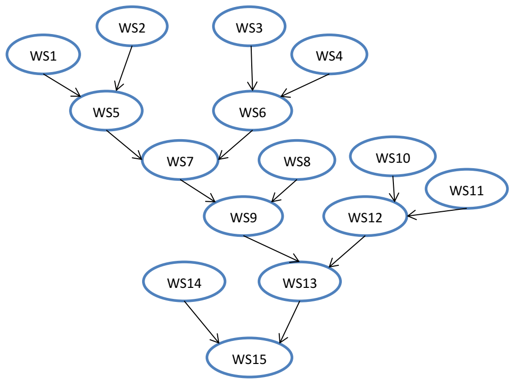
30	0	0	1
1	20	3	2
3	1	5	7
1	2	2	22

Table 4.14: Confusion Matrices: classifier1 (left) 82%, classifier2 (middle) 84%, classifier3 (right) 77%

converged to the true probabilities with an increase of the number of samples at convergence. We also applied the unpaired two-samples t-test with each of the estimated and actual parameters. For both CPT(1,4,3) and CPT(2,1,1), the test returned $h = 0$, the statistic values -1.9255 and 0.8506, and the standard deviations 0.0318 and 0.0234, respectively.

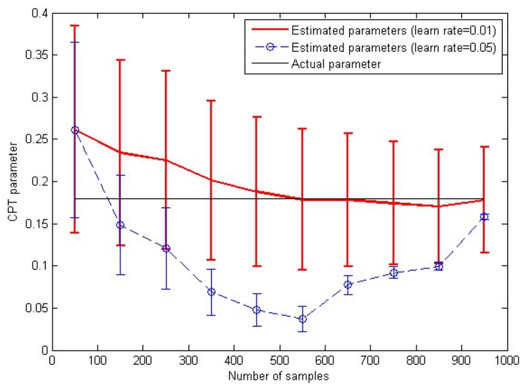
Online Computation of Trust

In this section, we design an experiment to compute the QoS-based trustworthiness of composite web services online. Given the QoS ratings of the composing services, learning the conditional probabilities of the QoS ratings of the composite service yields the following; prediction of the trustworthiness of a composite service after each observation, understanding the influence of the constituent services on the overall quality of the composite service. Thus, the preceding provides

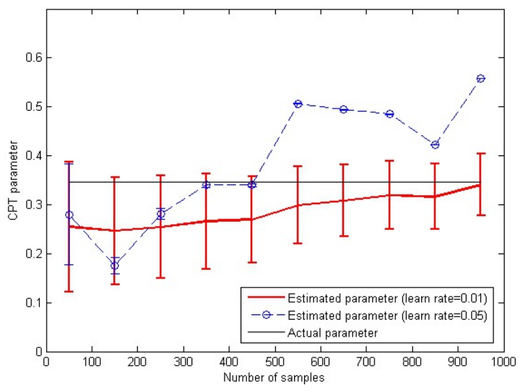


$$\begin{tikzcd}
WS1 \arrow[dr] & WS2 \arrow[d] & WS3 \arrow[d] & WS4 \arrow[dl] & & \\
& WS5 \arrow[dr] & WS6 \arrow[dl] & WS8 \arrow[d] & WS10 \arrow[d] & WS11 \arrow[dl] \\
& & WS7 \arrow[r] & WS9 \arrow[d] & WS12 \arrow[dl] & \\
& & WS14 \arrow[dr] & WS13 \arrow[d] & & \\
& & & WS15 & &
\end{tikzcd}$$

Figure 4.8: Bayesian network representation of a more complex composite web service.



(a) True vs. estimated parameter of CPT(3,3,3) using Voting EM.

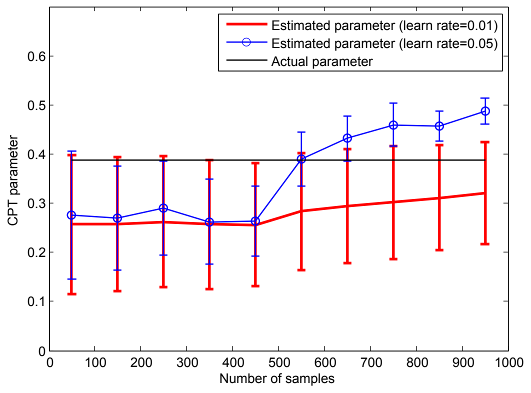


(b) Error bars of the estimated parameter CPT(1,4,3) using Voting EM (8 services).

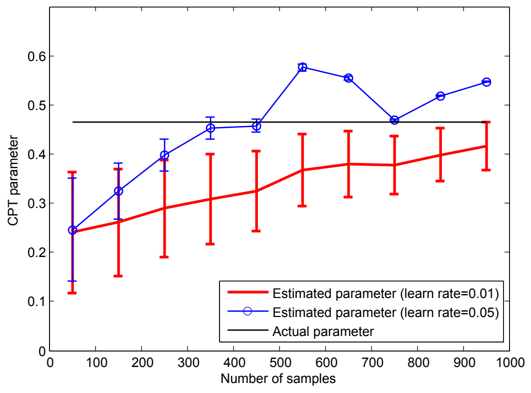
Figure 4.9: Error bars of the estimated parameter CPT(1,4,3) using Voting EM (8 services).

a means to enhance the construction of composite services. Then, given a Bayesian network structure, the conditional probabilities of the network's variables using the online learning rule of the Voting EM algorithm are estimated. Afterwards, the trust function from Equation 4.6 will be employed to estimate the trust values of the composite web service after each observation. In this experiment we construct a BN with the same structure as the one depicted in Figure 2 (we use WS in lieu of WebService). Subsequently, we sample the data as follows:

- The quality values (classes) of WS1 and WS4 are sampled from a multinomial distribution with the parameters $[0.4\ 0.3\ 0.1\ 0.2]$.



(a) Estimated parameter CPT(1,4,3).

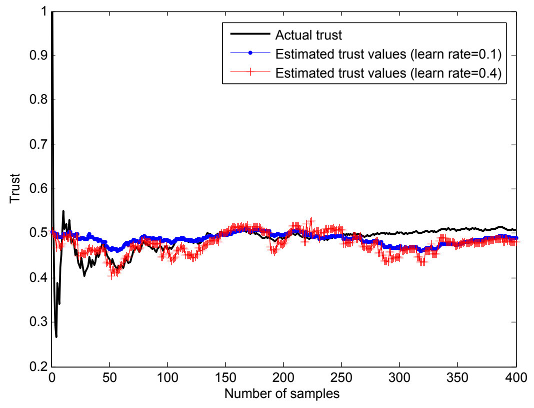


(b) Estimated parameter CPT(2,1,1).

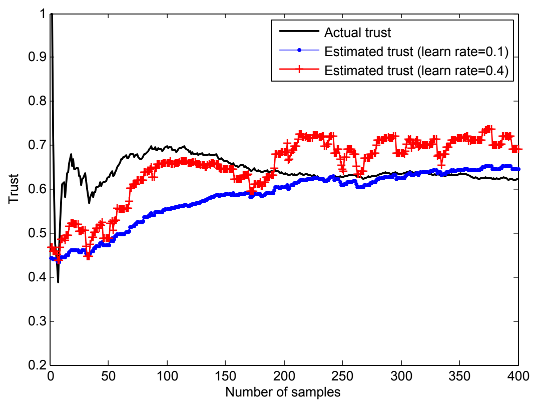
Figure 4.10: Error bars of the estimated parameter CPT(1,4,3) and CPT(2,1,1) using Voting EM (15 services).

- The quality values (classes) of WS2 and WS5 are sampled from a multinomial distribution with the parameters $[0.1\ 0.1\ 0.4\ 0.4]$.
- The quality values of WS3 are computed using the Switch composition function that chooses the values either from WS1 or WS2 based on the multinomial distribution with the parameters $[0.8\ 0.2]$.
- The quality values of WS6 are computed using the Switch composition function that chooses the values either from WS4 or WS5 based on the multinomial distribution with the parameters $[0.5\ 0.5]$.
- The quality values of WS7 are computed using the Switch composition function that chooses the values either from WS5 or WS6 based on the multinomial distribution with the parameters $[0.8\ 0.2]$.

Afterwards, we take each data instance at a time and compute the CPTs of the constructed network using the Voting EM update rule described earlier. These CPTs are then used to compute the probabilities of the corresponding service to belong to each of the 4 quality classes which are then employed to estimate a trust value using Equation 4.6. The actual trust values are computed using the same equation. However, the probabilities, in this case, are the sufficient statistics computed from the data. In other words, p_1 is equal to the number of observations on which WebService7 belonged to class 1 over the total number of observations. Furthermore, we test our approach with different composition operators, Max, and Min. Therefore, we follow the same steps outlined above, however, the quality values of WS6 and WS7 are sampled using the Max and Min operators, respectively. The learned trust values with $\eta(0.1)$ and $\eta(0.4)$ are compared to the real ones in Figure 4.11. This figure shows the convergence of the estimated trust values to the actual ones. It is also noticeable that the variance of the estimates obtained with the smaller $\eta(0.1)$ have less variance than the case of larger $\eta(0.4)$. These results are consistent with the convergence analysis of the Voting EM algorithm in [79]. The closer η gets to 1 leads to faster convergence and larger variance. However, the closer η gets to 0 yields a slower convergence and a smaller variance.



(a) Switch composition



(b) Max-Min

Figure 4.11: True vs. estimated trust scores with $\eta = 0.1$ and $\eta = 0.4$.

4.2.2 Mixture Models Experiments

To the best of our knowledge, no previous mixture models of multi-dimensional distributions have been previously applied in the context of multi-agents computational trust. To evaluate the accuracy of these models, we first compare the mixture of *MDD* to the multinomial mixture approach. The same experiments were developed for the *MGDD* and *MBLD* mixture models. For the data required in these experiments, we used the quality rankings of two web services. These were used to compute the quality rankings of the composite service according to the composition function used. The proposed mixture models can also scale to any number of web services. Tables 4.15, 4.16, and 4.17 show the results of the *MDD* mixture model with each of the *Switch*, *Min* and *Max*, and *Product* and *Sum* composition functions, respectively. The parameters of the actual data are the maximum likelihood estimations computed using the Newton iteration algorithm [86]. It is noticeable that the composition operator influences the performance of the mixture model with the highest being associated with the *Switch* operator. In the case of *Min* and *Max* operators, the α parameters estimates are not very accurate. However, the P estimates capture the responsibilities of each of the mixture components. These estimates show that in case of *Min* and *Max*, one of the agents has the most influence on the quality of the composite service. However, with the *Sum* and *Product* functions we can see that these responsibilities move towards equality as both web services contribute to the overall quality of the composite service.

	SWITCH				
	α_1	α_2	α_3	α_4	P
Actual	0.0367	0.3250	0.5700	0.0683	0.3000
	0.0384	0.1022	0.8300	0.0294	0.7000
Multinomial Mixture	0.0440	0.2865	0.6133	0.0562	0.3881
	0.0342	0.0945	0.8400	0.0314	0.6119
MDD Mixture	0.0145	0.3685	0.5408	0.0762	0.3099
	0.1515	0.0989	0.5883	0.1613	0.6901

Table 4.15: Estimated parameters by multinomial mixture and *MDD* mixture with *Switch* composition operator.

	MIN					MAX				
	α_1	α_2	α_3	α_4	P	α_1	α_2	α_3	α_4	P
Actual	0.0390	0.3370	0.5590	0.0650		0.0380	0.2600	0.6660	0.0360	
	0.0251	0.1024	0.8405	0.0320		0.0410	0.3950	0.4880	0.0760	
Multinomial Mixture	0.0331	0.3329	0.5401	0.0939	0.4968	0.0376	0.3088	0.6222	0.0315	0.5001
	0.0252	0.0979	0.8184	0.0586	0.5032	0.0414	0.3463	0.5318	0.0805	0.4999
MDD Mixture	0.0349	0.3425	0.5283	0.0943	0.1051	0.0245	0.1015	0.8380	0.0360	0.8502
	0.0234	0.0968	0.8203	0.0596	0.8949	0.0376	0.3284	0.5766	0.0574	0.1498

Table 4.16: Estimated parameters by multinomial mixture and *MDD* mixture with *Min* and *Max* composition operators.

Figures 4.12 to 4.17 illustrate the results of learning the trust of a composite web service using the three proposed mixture models. We compute the quality of the composite service using

	Product					SUM				
	α_1	α_2	α_3	α_4	P	α_1	α_2	α_3	α_4	P
Actual	0.0410	0.3490	0.5620	0.0480		0.0375	0.2989	0.6000	0.0636	
	0.0292	0.1079	0.8390	0.0239		0.0312	0.1714	0.7643	0.0330	
Multinomial Mixture	0.0000	0.1674	0.8287	0.0039	0.3284	0.0294	0.2279	0.6783	0.0645	0.4999
	0.0039	0.0288	0.9657	0.0017	0.6716	0.0386	0.2271	0.7057	0.0285	0.5001
MDD Mixture	0.0055	0.1301	0.8587	0.0057	0.4650	0.0379	0.3531	0.5508	0.0581	0.2765
	0.0009	0.0300	0.9680	0.0011	0.5350	0.0310	0.1020	0.8330	0.0340	0.7235

Table 4.17: Estimated parameters using multinomial mixture and *MDD* mixture with *Product* and *Sum* composition operators.

the different composition functions discussed earlier. After learning the mixture parameters, the probability of each quality class is evaluated using the equations (3.4), (3.6), and (3.8) for each of the mixture of *MDD*, *MGDD*, and *MBLD* respectively. The learned trust is computed using the weighted trust function given by equation (4.8). These results show the ability of the mixture models to learn the trust of composite services. They also capture the responsibility of each of the constituent service in the quality of the composite service. This is very clear in the *Switch* graphs in which the quality of the composite service is inherited from one of the constituent web services.

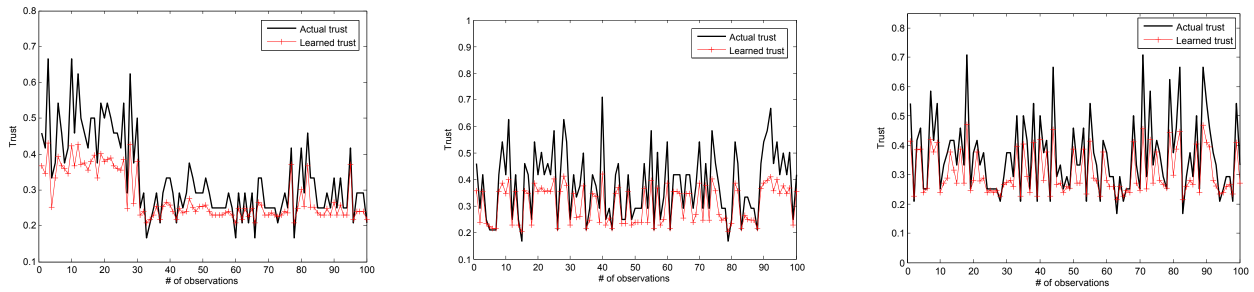


Figure 4.12: Actual vs. learned trust using the mixture of *MDD* with *Switch* (left), *Min* (center), and *Max* (right) composition functions

We evaluate the effectiveness of modeling the QoS using the MBLD by designing a BN classifier that employs the Bayesian approach with a Beta-Liouville prior distribution for each CPT entry. We compare the performance of this classifier to another BN classifier with a Dirichlet prior and to the naive Bayes classifier. We also test the Voting EM algorithm in learning the BN parameters online. In all the experiments, we adopt the four quality class model used in the QWS data

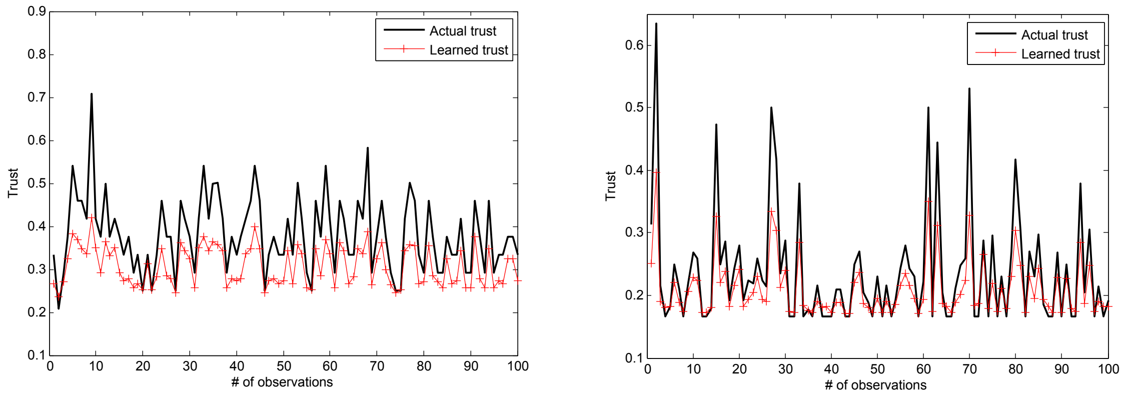


Figure 4.13: Actual vs. learned trust using the mixture of *MDD* with *Sum* (left) and *Product* (right) composition functions

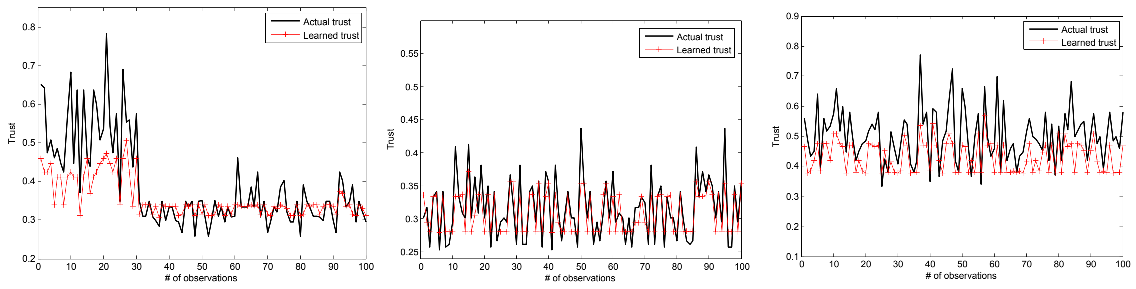


Figure 4.14: Actual vs. learned trust using the mixture of *MGDD* with various *Switch* (left), *Min* (center), and *Max* (right) composition functions

set [84] which classifies the QoS of a web service in one of four classes; Platinum, Gold, Silver, and Bronze. The Platinum class represents the highest quality a web service can be assigned to and the Bronze represents the lowest. The data used in the experiments is described in the following section.

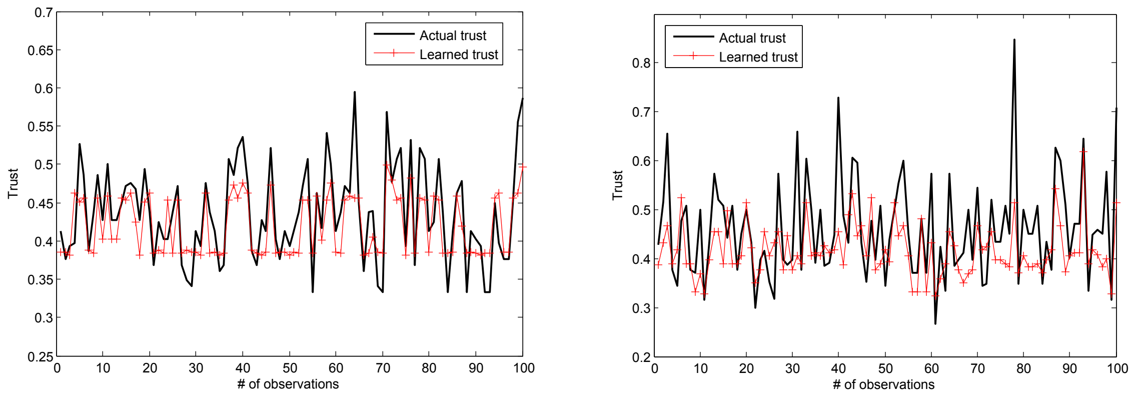


Figure 4.15: Actual vs. learned trust using the mixture of *MGDD* with *Sum* (left) and *Product* (right) composition functions

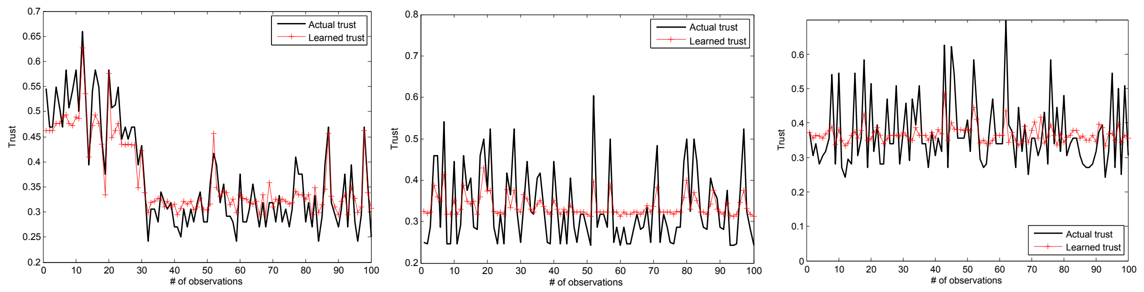


Figure 4.16: Actual vs. learned trust using the mixture of *MGDD* with various *Switch* (left), *Min* (center), and *Max* (right) composition functions

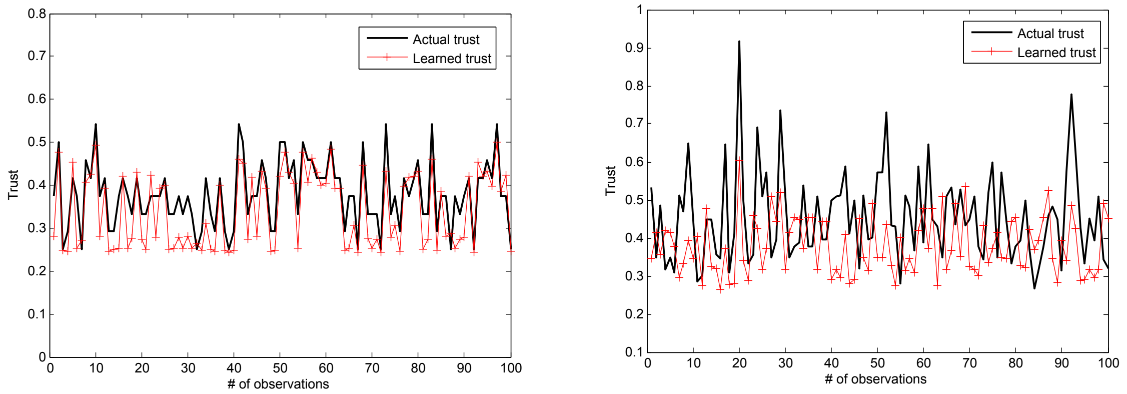


Figure 4.17: Actual vs. learned trust using the mixture of *MBLD* with *Sum* (left) and *Product* (right) composition functions

Chapter 5

Reputation Aggregation and Filtering Approaches

We described, in the previous chapters, various models in which each service consumer assesses the trustworthiness of a web service based on their direct interactions. However, it is not always sufficient to rely on trust scores that are subjective and personalized. For example, a consumer might have no personal experience with a specific provider. It is then risky to interact with this provider without having any insights about its quality. Therefore, to compensate the lack of personalized opinion about a service provider, consumers might request the public opinion. This is referred to as the service providers reputation which is a composite evaluation of the QoS delivered based on its interactions with a group of consumers.

5.1 Reputation Background

In QoS-based trust systems, the reputation of agents is equivalent to the estimated value(s) of a single or multiple QoS metric(s) [18, 87]. A popular method for aggregating reputation feedback is the summation method. However, the embraced simplicity of this method opens the door for malevolent agents to maneuver it for their own benefits [88]. This could be achieved by deceitfully increasing their own reputation or decreasing the reputation of others. eBay is a popular commercial example of reputations systems that employ the summation method. In eBay buyers and sellers rate each others based on their transactions as positive, neutral, or negative. The overall reputation of a seller is mostly represented by a feedback score that is computed by summing all the positive ratings minus all the negative ones. Eigentrust, a reputation management framework in P2P networks, also adopted the summation method for feedback aggregation [89]. In Eigentrust, the truster weighs the trust feedbacks received from other peers by their corresponding trust scores. The aggregated trust score assigned to a trustee is then the sum of the product of the trust feedbacks and the trust scores of the feedbacks senders. PeerTrust, another P2P trust system, also employs the summation method with various trust metrics, taking into account the credibility of the feedbacks senders [90]. Alternatively, [91, 92] introduced multiple operators to handle different scenarios of trust propagation in a network of interacting agents. For instance, the "Concatenation" operator is used when computing the trust of an agent A in agent C based on the trust of agent B in C discounted by the trust of A in B.

Driven by the uncertainty nature of trust, Bayesian reputation models, based on binary or multi-valued ratings, compute the reputation scores by statistically updating the Beta [93] or Dirichlet [87, 94] probability density functions, respectively. When a new rating arrives after an interaction with a service, it is added to the previous ratings (a priori) to compute the new reputation score (a posteriori). The Bayesian approach of updating reputation scores also follows the summation concept. To overcome the vulnerability of the summation method, other studies exploited the Kalman filter capabilities to aggregate the reputation feedbacks [95, 96]. These studies argued that the Kalman-based reputation aggregation repels malicious feedbacks by keeping track of the estimates' variances.

However, the above methods lack the capability of dealing with the unknown correlations between reputation feedback provided by different sources. Consider the following scenario: let X_{abc} and X_{adc} be the reputation estimates of agent *a* supplied by agents *b* and *d* to agent *c*, respectively. Suppose, *c* aggregates X_{abc} and X_{adc}, then supplies the result, X_{acb}, back to *b*. This scenario exposes the dependency between X_{acb} and X_{abc}. Therefore, there should be a method to combine X_{abc} and X_{acb} while taking into consideration their unknown correlation. To handle such cases, we extend the literature by the following main contributions:

- We present a QoS-based reputation model that considers the subjectivity of interacting agents in their QoS requirements. As such, two agents communicating with the same service might be interested in different QoS metrics. Our model suggests a customized aggregation of feedbacks based on the QoS requirements of the agent.
- We propose a feedback aggregation approach based on the covariance intersection (CI) and ellipsoidal intersection (EI) data fusion methods. The former aims to handle the aggregation of two reputation estimates in cases where the error's ellipse of one of the estimates is contained in the error's ellipse of the other. The latter aggregates the reputation estimates based on their exclusive information by introducing their mutual mean and covariance.

In this chapter, we extend our trust models by integrating the reputation of a service provider sP in its overall trustworthiness. this is achieved by aggregating its trust scores estimated by M service consumers $\{sC_1, sC_2, \ldots sC_M\}$ who are willing to share their experiences. We present three reputation models:

- Model 1: The first reputation model is built upon the trust approach proposed in Chapter 1. This model aggregates the reputation feedback of a web service after factoring in the uncertainty assigned to each feedback. This model is strengthened by an extension that captures the unfair (malicious) feedback using clustering and outlier detection methods. Another extension was developed to apprehend various dynamic behaviors of web services.
- Model 2: The second reputation model presents two algorithms for dynamiz aggregation of reputation feedback. These are based on two data fusion methods; Covariance Intersection (EI) and Ellipsoidal Intersection (EI).
- Model 3: The third model considers the reputation of web services that are engaged in Communities of Web Services (CWS). This model is based on various data mining techniques including clustering and outlier detection.

5.2 Reputation Aggregation: Model 1

Due to the flood of web services that offer similar functionalities, service consumers are left with a challenging selection decision. A popular approach to assist them with the service selection task is based on the reputation of web services. However, the propagation of reputation feedback in an open and distributed system of web services yield correlated reputation estimates. The existing web service reputation literature still lacks a system that handles the aggregation of reputation feedback under unknown correlation. To fill this gap, we employ two data fusion algorithms, the covariance intersection and ellipsoidal intersection, to aggregate QoS-based reputation feedback. Our experimental results endorse the advantageous capability and scalability of the proposed methods in aggregating reputation estimates, and show an enhanced performance when compared with the Kalman filter method.

5.2.1 QoS-Based Reputation

In this section, we propose a QoS-based reputation model that incorporates the subjectivity matter of interacting agents and maintains a consistent representation of reputation reports.

1. We model the reputation feedback as vectors of multiple QoS metrics and their corresponding values. As such, Each agent will consider the QoS metrics that fit its requirements. These metrics are either monotonically increasing or decreasing metrics [97]. Monotonically increasing and decreasing refer to QoS metrics whose values are positively and negatively correlated with the overall quality of a service, respectively. Popular QoS metrics include *response time*, *throughput*, *availability*, *reliability*, and *cost*.

2. The reported values of the QoS metrics are then scaled to different ranges each of which is defined by two thresholds. These represent a lower and upper bounds on the values of these metrics, T_l and T_u, respectively. The scaling of these values allows each agent to define the ranges of values of the various QoS metrics it deals with. The scaling to the range $[T_l, T_u]$ is given by:

$$Q'_{m_i} = \begin{cases} Q_{m_i} & \text{if } i = 1 \quad \& \quad T_l < Q_{m_i} < T_u \\ T_l & \text{if } i = 1 \quad \& \quad Q_{m_i} < T_l \\ T_u & \text{if } i = 1 \quad \& \quad Q_{m_i} > T_u \\ \frac{(T_u - T_l)(Q_{m_i} - \min(Q_{m_i}))}{\max(Q_{m_i}) - \min(Q_{m_i})} + T_l & \text{otherwise,} \end{cases} \tag{5.1}$$

where Q_{m_i} is the value of the observed QoS metric m after the i^{th} interaction. $\min(Q_{m_i})$ and $\max(Q_{m_i})$ are the minimum and maximum of the values of the QoS metric m up to the i^{th} interaction.

3. Afterwards, the scaled values are normalized to the $[0, 1]$ range. The values of monotonically increasing metrics are normalized by $Q''_{m_i} = \frac{Q'_{m_i} - \min(Q'_{m_i})}{\max(Q'_{m_i}) - \min(Q'_{m_i})}$. The normalization of the scaled values of monotonically decreasing metrics is given by $Q''_{m_i} = 1 - \frac{Q'_{m_i} - \min(Q'_{m_i})}{\max(Q'_{m_i}) - \min(Q'_{m_i})}$.

This approach allows each agent to select and later aggregate the values of specific QoS metrics. Scaling and normalizing the values of the QoS metrics give the agents that receive reputation feedback the choice of aggregating the values of a selective set of QoS metrics. They also decrease the impact of malicious feedback by restricting the values to fit within specific ranges. These ranges could be based on prior direct interactions between the agent that requests the feedback and the evaluated agent.

In most probability-based trust systems, the reputation is communicated through the sufficient statistics of either the Beta or Dirichlet distributions [93, 94]. However, applying the sufficient statistics undergoes the issue of redundant information when passing reputation among agents. The authors in [62] proposed the partitioning of information between *private* and *shared*. The former denotes information that has not been communicated to other agents. The latter represents information that could have been sent to or received from other agents. Despite being attractive, this solution requires each agent to keep track of two separate information. It also involves identity issues of sending and receiving agents.

To overcome these shortcomings, we propose an alternative solution to the aggregation of reputation feedback of web services. We exploit the CI and EI fusion methods commonly used for information fusion in distributed networks. We compare the results of these methods with those of the Kalman filter reputation models proposed in [96, 98]. The next section is dedicated to overview the Kalman filter method for reputation aggregation.

5.2.2 Kalman Filter for Service Reputation

Kalman filter is considered a form of a Gaussian process model and a predictive filter based on recursive algorithms [34]. Given a noisy dynamic system with unknown states, the Kalman filter predicts the state using the dynamic model of the corresponding system. Afterwards, the prediction results are corrected and updated by considering a noisy measurement in what is called the observation model. For example, given a moving robot, the dynamic model can be employed to estimate the robot's position (the unknown state) at a certain time. This estimate is then updated using the measurements of the robot's position that are supplied by a camera (observation model). In the web service reputation context, the unknown states represent the reputation scores of an agents QoS metrics. The dynamic, the unknown states represent the reputation scores of an agent's QoS metrics. The dynamic and observation models are given by:

- **Dynamic model:**
$$x_{t+1} = Fx_t + w_t, \tag{5.2}$$
where x_t is the state vector at time t, $w_t \sim N(0, Q)$ is the Gaussian noise associated with the dynamic model at time t, and F is the state transition matrix.
- **Observation model:**
$$y_t = Hx_t + v_t, \tag{5.3}$$
where y_t is the observation at time t, $v_t \sim N(0, R)$ is the Gaussian noise associated with the observation model at time t, and H is the observation transition matrix.

The Kalman filter iterates recursively over the following two steps to minimize the covariance of the estimation errors:

1. Prediction: This step computes the a priori estimates of the current state by ignoring the dynamic noise. The prediction equations are obtained by solving the dynamic model's differential equations, and are given by:

$$\hat{x}_{t+1} = Fx_t \tag{5.4}$$

$$\hat{P}_{t+1} = FP_tF^T + Q, \tag{5.5}$$

where $\hat{P}_{t+1}$ is the covariance matrix of the predicted state.

2. Correction: This is where the a priori estimates are enhanced by adding the measurements observed at time t. The correction equations are the following:

$$x_{t+1} = \hat{x}_{t+1} + K(y_t - \hat{x}_{t+1}) \tag{5.6}$$

$$P_{t+1} = (I - KH)P_t, \tag{5.7}$$

where K, the Kalman gain matrix, is given by $K = P_tH^T(HP_tH^T + R)^{-1}$.

This model is suitable for the estimation problem of the reputation of web services and has been employed in multiple studies such as [98] and [95]. This is due to the dynamic nature of reputation and the possibility of modeling it by a linear function disturbed by a Gaussian noise.

5.2.3 Proposed Aggregation Methods

The main limitation of the Kalman filter is the independence assumption between the estimation error at time t and the measurement error at time $t+1$ [99]. Let's consider the following scenario: *Ag4* receive two feedback reports of *Ag1*'s reputation from both *Ag2* and *Ag3*. Afterwards, *Ag2* requests reputation feedback about *Ag1*. *Ag4* then responds to *Ag2*'s request and sends the information it formerly received from the latter as it did not have additional experience with *Ag1* since then. Using Kalman filter, when *Ag2* combines this information with its own estimates of *Ag1*'s reputation, the covariance matrix is unjustifiably decreased. Suppose *Ag2*'s estimates of the availability *AV* and response time *RT* of *Ag1* are given by the state vector $x_t = \{0.8504, 0.7154\}$ and the covariance matrix is given by: $P_t = \begin{pmatrix} 0.0367 & 0.0044 \\ 0.0044 & 0.0367 \end{pmatrix}$. Also, let the feedback sent by *Ag4* be the observation vector $y_t = \{0.8504, 0.7154\}$. Moreover, let the following be the state transition and covariance matrices, and the observation transition and covariance matrices, respectively: $F = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $w_t = \begin{pmatrix} 0.02 & 0 \\ 0 & 0.02 \end{pmatrix}$, $H = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $v_t = \begin{pmatrix} 0.03 & 0 \\ 0 & 0.03 \end{pmatrix}$. The Kalman filter yields the same state estimates $x_{t+1} = \{0.8504, 0.7154\}$. However, the covariance matrix becomes $P_{t+1} = \begin{pmatrix} 0.0164 & 0.0009 \\ 0.0009 & 0.0164 \end{pmatrix}$. This means the uncertainty in the corrected estimates is reduced when it should have remained unchanged. The methods below are proposed to deal with such scenarios.

Covariance Intersection

The CI algorithm was developed to overcome the independence assumption of the Kalman filter and the under/overestimation of the covariance matrix. CI combines estimates (mostly Gaussian) from different sources when the error ellipse of one estimate includes the error ellipse of the other [100]. In a nutshell, CI creates a convex combination of the means and covariances of two estimates to provide consistent fused estimates with appropriate covariance matrices:

$$P_f^{-1} = wP_i^{-1} + (1-w)P_j^{-1} \tag{5.8}$$

$$\hat{p}_f = P_f(wP_i^{-1}\hat{p}_i + (1-w)P_j^{-1}\hat{p}_j) \tag{5.9}$$

where $\hat{p}_f$ and P_f are the fused mean and covariance of the unknown state and are based on the estimates $\hat{p}_i$ and $\hat{p}_j$ and their respective covariances P_i and P_j. $w \in [0,1]$ is a regulating parameter. The two equations of CI yield consistent estimates regardless of the cross-correlation errors among the combined estimates [101]. This is achieved by having P_f include the intersection of P_i and P_j. The CI algorithm also provides the uncertainty in the combined estimates through the new covariance matrix P_f. More details about the derivation of the CI equations are available from [100, 102].

Ellipsoidal Intersection

Despite solving the consistency of the covariance matrix of fused estimates, CI still considered a combination of two local estimates without distinguishing between accurate and inaccurate estimates. The EI fusion method was proposed in [103] to handle the separation of the mutual information that might be included in the two local estimates. The update of the estimates is then only based on the exclusive information received to avoid incorrect and "over confident" estimates. The mutual information introduces two estimates, the mutual covariance and mutual mean. The mutual covariance is used to optimize the fused covariance by maximizing the effect of the mutual information, $P_f = (P_i^{-1} + P_j^{-1} - \Gamma^{-1})^{-1}$,

where P_i and P_j are the covariances of the state vector estimated by agents i and j. Γ is the mutual covariance and is defined as $\Gamma = S_i D_i^{0.5} S_j D_\Gamma S_j^{-1} D_i^{0.5} S_i^{-1}$,
where S_i and D_i contain the eigenvectors and eigenvalues of the covariance matrix P_i, respectively, such that $P_i = S_i D_i S_i^{-1}$. S_j is a matrix that contains the eigenvectors of $D_i^{-0.5} S_i^{-1} D_j S_i D_i^{-0.5}$, and D_Γ is given by:

$$[D_\Gamma]_{qr} = \begin{cases} \max([D_\Gamma]_{qr}, 1) & \text{if } q = r, \\ 0 & \text{otherwise.} \end{cases} \tag{5.10}$$

The mutual mean aims at representing an estimate of the mean of the state vector based on mutual information between initial estimates $\hat{p}_i$ and $\hat{p}_j$, and is given by:

$$\begin{aligned} \gamma \;=\; & (P_i^{-1} + P_j^{-1} - 2\Gamma^{-1} + 2\eta I)^{-1} \times \\ & ((P_j^{-1} - \Gamma^{-1} + \eta I)\hat{p}_i + (P_j^{-1} - \Gamma^{-1} + \eta I)\hat{p}_j), \end{aligned} \tag{5.11}$$

where ηI is added to guarantee $P_i^{-1} - \Gamma^{-1}$ and $P_j^{-1} - \Gamma^{-1}$ are both positive definite. Let $H = P_i^{-1} + P_j^{-1} - 2\Gamma^{-1}$ and $\lambda_{0_+}(H)$ be the smallest eigenvalue of H, η is defined in [103] as:

$$\eta = \begin{cases} 0, & \text{if } \det(H) \neq 0 \\ c \ll \lambda_{0_+}(H), & \text{if } \det(H) = 0 \end{cases} \tag{5.12}$$

Γ and γ are then employed to optimize the fusion of both estimates $\hat{p}_i$ and $\hat{p}_j$:

$$x_f = (P_i^{-1} + P_j^{-1} - \Gamma^{-1})^{-1}(P_i^{-1}\hat{p}_i + P_j^{-1}\hat{p}_j - \Gamma^{-1}\gamma). \tag{5.13}$$

5.3 Reputation Aggregation: Model 2

In Chapter 3, we proposed a correlation-based trust model that computes a web services trust score based on the correlations of the pairwise outcomes of its QoS metrics. Another score that is presented in this model is the variance of the estimated trust scores. Therefore, a reputation feedback about the service provider sP, supplied by consumer sC_i, consists of the trust score $Tr(sP, sC_i)$ (Equation (3.19)), and the variance associated with this score, $Var(Tr(sP, sC_i))$ (Equation (3.20)). Since the latter represents the uncertainty in the estimated score, then $1 - Var(Tr(sP, sCi))$ represents the certainty or confidence in the trust score. Therefore, prior to aggregating the trust scores estimated by of sPs consumers, we discount each score by its confidence level. As such, $Tr(sP; sCi)$ becomes:

$$Tr(sP, sC_i) = Tr(sP, sC_i)(1 - Var(Tr(sP, sC_i)))$$

5.3.1 Unfair referrals

Suppose N web services interact among themselves and each maintains a reputation of each web service it interacts with. After a time window (W), web service sC_1 sends a request to all the other web services for referrals about the reputation of sP based on their interactions with the latter within W. A fraction of N will respond to sC_1's request and share their experiences with sP. However, the received referrals will be most probably contaminated by malicious ones. These include magnified or diminished trust referrals that are dishonestly supplied to improve the reputation of the sP or to bad-mouth it, respectively. Therefore, the direct aggregation of the referrals will yield a biased and unjust reputation score. This raises the need for a technique that diminishes the impact of unfair referrals on the final reputation score of a web service. In PeerTrust [90], the authors introduced a credibility score that the feedback aggregator peer assigns to each referee peer based on the similarity between its referrals about other common peers and those of the aggregator. One limitation of this approach is that the number of similarity checks will increase as the number of agents or web services increases. We propose an alternative credibility score that is based on the own experiences of the requester (sC_1 in the example). This is achieved by a test request sent by sC_1 about the reputation of another service provider sP_{test} with which it has a large number of previous interactions and hence is certain about its reputation. The similarities between the received referrals and the reputation maintained by sC_1 are computed to depict the credibility of the corresponding referees. It is noteworthy to mention that a trust referral encloses the truster (sC), the trustee (sP), and the trust score, $Tr(sP, sC)$. Using this approach, sC_1 keeps learning about the credibility of other web services by constantly comparing his own referrals with theirs. Moreover, we leverage the capabilities of LDCOF [104], a local density cluster-based outlier factor algorithm, to further detect malicious referrals as anomalies. In our approach, we use the LDCOF with the K-means clustering algorithm. The motivation for considering the density factor of clustered referrals is to handle cases when the rate of malicious referees is larger than that of honest one. In such cases, malicious referrals might outnumber honest ones and hence end up forming a larger cluster. Moreover, malicious referees, in the majority of no-collusion cases, tend to assign random low or high reputation scores depending on their disparate intentions for lying. Therefore, considering the size of the cluster to be the sole discriminator between honest and malicious referrals can be problematic. However, by considering the cluster density factor, the algorithm assumes that honest referrals related to the same service should have small variance and thus reside within a dense area. Malicious referrals will most probably have a larger variance and reside in a large and dispersed area. The LDCOF score represents the minimum distance of a reputation to a large cluster divided by the average distance between all the elements of that large cluster and its centroid; this is given by:

$$LDCOF(r) = \begin{cases} \frac{\min(d(r,C_i))}{Avg_d(C_i)} & \text{if } r \in C_S \text{ and } C_i \in C_L, \\ \frac{(d(r,C_i))}{Avg_d(C_i)} & \text{if } r \in C_i \in C_L, \end{cases} \tag{5.14}$$

where C_S and C_L refer to sets of small and large clusters, respectively, and $Avg_d(C_i)$ is defined by:

$$Avg_d(C_i) = \frac{\sum_{j \in C_i} d(j, C_i)}{|C_i|}. \tag{5.15}$$

The proposed approach for reducing the impact of unfair referrals on the reputation of the referred web service is summarized in Algorithm 3.

Algorithm 3: Reputation aggregation of sP feedback

```
Data: Two sets $\mathcal{T}_{sP} = \{Tr(sP, sC_1), \ldots, Tr(sP, sC_n)\}$ and
      $\mathcal{T}_{sP_t} = \{Tr(sP_t, sC_1), \ldots, Tr(sP_t, sC_m)\}$
Result: Aggregated reputation of $sP$
foreach $sC \in (\mathcal{T}_{sP_t} \cap \mathcal{T}_{sP})$ do
    $Cr(sC) \leftarrow 1 - |Tr(sP_t, sC)|$;
end
$\mathcal{C} \leftarrow Xmeans(\mathcal{T}_{sP})$;
foreach $c \in \mathcal{C}$ do
    for $i \leftarrow 1$ to $|c|$ do
        $outlier\_score(sC_i) \leftarrow LDCOF(sC_i)$;
        if $outlier\_score(sC_i) > 1$ and $Cr(sC_i) > threshold$ then
            Remove $sC_i$ from $\mathcal{T}_{sP}$;
        end
    end
end
$Rep(sP) = \frac{\sum_{sC \in \mathcal{T}_{sP}} Cr(sC) Tr(sP, sC)}{\sum_{sC \in \mathcal{T}_{sP}} Cr(sC)}$;
```

5.3.2 Dynamic Behavior

The dishonest behavior of web services discussed in the previous section revolves around reporting malicious feedback about the quality of other service providers. Algorithm 3 suggests a series of steps to handle such behavior and diminish its impact on the web services reputation. However, service providers could also be malicious by providing non-consistent quality during their interactions with service consumers. The authors in [90] described three strategic manners by which service providers change their quality of service:

- Milking reputation: The web service performs well for a long time to have a good reputation and then starts to degrade its performance.
- Building reputation: The web service improves its performance and shifts from bad quality service to good quality one.
- Oscillating reputation: The web service oscillates between the previous strategies; milking and building its reputation.

To handle such behaviors, an adaptive algorithm, PeerTrust adaptive, was introduced in [90] that implements the following logic. The trust of a peer is computed twice; the first is based on all the feedback received within a time window and the second considers the feedback from a smaller time window. If the second trust value is smaller than the first by a threshold, then the peer is assigned the former, and the latter otherwise. This approach harshly penalizes any web service that drops its quality even if unintentionally and due to an exceptional failure of some sort. In such cases, if the quality drops at time t and picks up at $t+1$, the adaptive algorithm requires a large number of good quality interactions before it builds the reputation of the corresponding service provider back up.

We propose a simplistic approach that avoids such harsh punishments while staying conservative in cases of quality drops and elevations. This approach complements Algorithm 3 by assigning a time decreasing weight to each of the reputation feedback within a time window. As such, this follows the intuitive assumption that older feedback weigh less than newer ones. The weight, W_t of $Tr_t(SP, sC)$, a feedback about an interaction that occurs at time t, is given by:

$$W_t = \frac{t+1}{|Win| - t + 1}, \tag{5.16}$$

where $|Win|$ is the size of the considered time window. The reputation of a service provider at time t, $Rep(sP_t)$, as given in Algorithm 1 is computed by Equation 5.17. For simpler notation, we substitute $Cr_t(sC)$ by Cr_t to represent the credibility of the service provider sP at time t. We also refer to the feedback given by sC at time t by Tr_t rather than $Tr_t(SP, sC)$. Thus, $Rep(sP_t)$ is defined as follows:

$$Rep(sP_t) = \frac{\sum_{sC \in \mathcal{T}_{sP}} Cr_t Tr_t W_t}{\sum_{sC \in \mathcal{T}_{sP}} Cr_t W_t} \tag{5.17}$$

5.4 Reputation Aggregation: Model 3

We present in this Section a reputation model for agent-based web services grouped into communities by their equivalent functionalities. The reputation of each web service is based on the non-functional properties of its interactions with other web services from the same community.

We exploit various clustering and anomaly detection techniques to analyze and identify the quality patterns provided by each service. This enables the master of each community to allocate the requests it receives to the web service that best fulfill the quality requirements of the service consumers. Our experiments present realistic scenarios based on synthetic data that characterizes the reputation feedback of the quality provided by a web service at different times. The results showcase the capability of our reputation model in portraying the quality of web services that reside in a community and characterizing their fair and unfair feedback reports.

Communities of web services (CWS) were introduced to group web services with the ultimate goal of enhancing the discovery, selection, and composition of web services. The notion of community in the context of web services has been handled from various perspectives. In [105], a community groups agents that share "similar interests and judgment criteria" to build a consistent web service community-based reputation model which aims to improve the web service selection process. We follow the definition in [106] by which a community groups web services that provide the same functionality. However, services within the same community could be distinguished by their non-functional properties also referred to as quality of service (QoS) properties [107].

Each community is managed by a master that is responsible for marketing its community to attract more web services, distributing and allocating users' requests to specific web services, and identifying web services to be part of a composition. The master can be one of the web services in a community promoted to hold the master's role. The other web services, referred to as slaves, compete and collaborate to fulfill users' requests. The advantages of being part of a community include, among others, gaining exposure to a wider spectrum of users as well as participating in a larger number of compositions. The "concepts" and "operations" of web service communities are detailed in [106]. Various studies have exploited the architecture and communication protocols involved in CWS to address the community building aspects and the reputation of these communities [108, 109].

Trust and reputation have been extensively considered in the multiagent systems and web services research communities [10, 110]. One of the main challenges in trust and reputation models is estimating the trustworthiness of new agents entering a system. In such cases, the quality provided by this agent is unknown and the majority of trust systems assume equal a priori trust for all agents.

However, [111] characterized trustworthy and untrustworthy agents by discovering patterns using graph mining and knowledge representation. These patterns are then employed to train a regression model for the agents' trustworthiness. This model would be interesting to be considered in the formation of CWS which is outside the scope of this paper.

Concerning the reputation of web services, CWS were utilized in [112] to maintain high availability and overall web service quality that is up to the users' standards. The authors proposed that the community's master keeps track of the number of services within the community and that of the users' requests. If the former is less than a threshold and the latter is higher than another threshold, then the master invites more services to join its community. When the opposite scenario occurs, the master excludes the web services with low performances. The authors in [113] defined various quality metrics that characterize a community from the user and service provider's perspectives. Three metrics were identified to matter for users who are trying to select a community based on its reputation; the responsiveness, indemand, and satisfaction metrics. The first assesses the time the master of a community spends to select a web service to fulfill a user's request. The second metric evaluates the popularity of a community in comparison to other communities, and the third captures the users' opinions about a community. The authors in [114] considered the same metrics described previously. They also proposed an aggregation approach to combine these metrics and proposed an extended CWS architecture to to ensure a reliable logging mechanism. This extension allows the detection of fake user feedback and protects the reputation of the communities.

Nevertheless, the aforementioned reputation models either handle one quality metric of the slave web services in a community or focus on the reputation of the community as a whole. In this paper, we propose a reputation model that is specific to each individual web service in a community. This enables the community's master to be aware of the quality of the web services in its community, alert the ones that show unstable or low QoS patterns, and allocate the incoming requests to the ones the are best suitable to the users' QoS requirements.

5.4.1 Gaussian Reputation

Since web services within the same community provide similar functionalities, delegating one of them to fulfill a user's request based on non-functional requirements sounds rational. It is very common for service consumers to have certain constraints that translate into quality prerequisites to be met by the prospective web service. For instance, a time-critical application demands highly available and responsive services, and a data-critical system calls for reliable and secure services. To enable the selection, from communities of web services, of the ones that best match the quality requirements of service consumers requests, we propose that each web service maintains a reputation model of all the services it interacts with. This model consists of multivariate Gaussians which attributes represent various QoS metrics. The QoS-based reputation literature examined and dealt with a wide range of QoS metrics that are also referred to as QoS attributes or properties. The following is a partial list of the most prominent metrics:

One of two interacting web services in a community is the consumer and the other is the provider. After each interaction, the former can measure the *RT*, *LA*, and *CO* of the latter as well as report whether it received a valid response. After a number of interactions with the same service, the service consumer can compute the values of all the QoS metric mentioned above. For example, *RT*, *LA*, and *CO* would be equal to the average of the response time, latency, and cost of all the interactions, respectively. *TH* is then the number of requests handled by the service provider between the first and the last interaction per unit of time, *AV* is the total number of responses over the total number of requests the service consumer has sent, and *RE* is the number of valid responses over the total number of responses received by the consumer.

It is noteworthy to mention that the details of monitoring the various QoS metrics are outside the scope of this thesis. We assume that the web services agree on the monitoring approaches to be used within a community. They can also decide on the degree of invasiveness of these approaches that can also affect their performance. According to [115] an invasive approach is one in which the execution of a service and the monitoring activities are closely connected. When the monitoring occurs independently of the service execution process, it is then considered less invasive. For further details on current monitoring approaches and their characteristics, we refer the readers to [115].

Aside from the monitoring approach deployed, the reputation of a web service j as evaluated by a web service i can be modeled by a multivariate Gaussian:

$$Rep_{ij}^{t} = \{RT, TH, LA, AV, RE, CO, SE\},$$

where t is the time at which the QoS metrics were computed after the same predefined number of interactions with the web service j.

Modeling the reputation of web services as QoS-based Gaussians empowers the master of a community with the capability of learning and analyzing the quality of the web services in its community. It then enables the detection of the quality patterns of each of the community's web services. These patterns could be exploited to match a service consumer request with the service which quality patterns best fit this request's quality requirements.

5.4.2 Web Service QoS profile

The behavior (quality) of web services may change during its life time due to many factors. These include operational factors such as the addition/removal of certain parameters to/from the web services methods and the change in their deployment strategies. Other factors may be related to the lack of resources allocated to handle the service consumers requests. The reputation approach based on the QoS described in the previous section allows the modeling of reputation profile for each of the web services in a community. This could be achieved by training a model based on the values of the QoS metrics of the interactions between all the web services in a community. Figure 5.1 illustrates a use case diagram of the QoS-based reputation model whose steps can be summarized as follows:

The reputation of each web service is associated with multiple feedback reports each of which is supplied by a different service consumer (in this case, another web service in the community). These reports are then analyzed to detect possible outliers, QoS measurements that don't follow their prominent distribution(s). Many outliers detection techniques have been proposed in the literature [116]. For our purposes, we employ a clustering-based outlier detection algorithm that computes an outlier score for each of the data points based on the clustering results. First, the reported QoS values are normalized to the $[0, 1]$ range, so it can be modeled by a Gaussian distribution. We then cluster the normalized data using the Expectation-Maximization (EM) algorithm, an unsupervised approach that is widely used in clustering data by fitting it to a mixture model of multiple distributions. Formally, let $REP = [Rep_{ij}^{t}, Rep_{ij}^{t+1}, \ldots, Rep_{ij}^{t+n}]$ represents all the outcomes of the reputation of web service i evaluated by web service j from the time t until $t + n$ (n

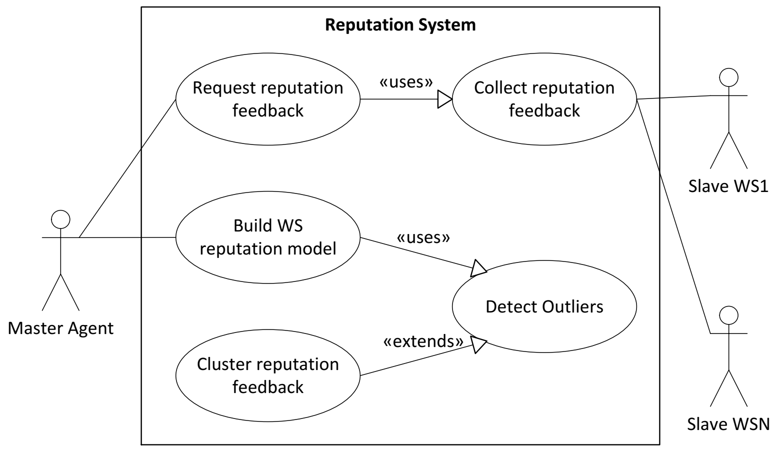

Reputation System
Request reputation feedback
«uses»
Collect reputation feedback
Build WS reputation model
«uses»
Detect Outliers
«extends»
Cluster reputation feedback
Master Agent
Slave WS1
Slave WSN

Figure 5.1: UML Use Case diagram of the proposed Reputation system

is the number of interactions of j with i). We assume that REP is generated by a k-component mixture of Gaussian, hence, its probability density function is given by:

$$p(Rep|\theta) = \sum_{m=1}^{k} \pi_m p(Rep|\theta_m), \tag{5.18}$$

where $\pi_1, \dots, \pi_k$ are the mixing coefficients, θ_m is the set of parameters of the m^{th} component, θ is the set of parameters that defines the mixture model. $p(Rep|\theta_m)$ is the density function associated with the m^{th} component. More details about mixture models are available in [117]. The clustering results can be expected to vary according to different scenarios:

- **Scenario 1:** The web service exhibits a reliable and stable QoS behavior. Formally, the QoS measurements of the web service follow one distribution.
- **Scenario 2:** The web service's behavior is not steady, hence, it provides QoS measurements that may be generated from different distributions. Therefore, multiple clusters with disparate number of data points will be returned.

In both scenarios, the clustering challenge is two fold; determining the best number of clusters and deciding whether each of them represents a real QoS behavior of the service. It can occur that the reputation feedback includes falsified QoS measurements contained in one or more of the clusters. This is known in the trust and reputation literature as unfair feedback which is best described as augmented feedback to promote a friend or reduced feedback to weaken a foe.

To determine the best number of clusters, the EM algorithm starts with a relatively large number of clusters and remove, after each iteration, the ones that contain few data points. However, given the first scenario, one would assume that the data follows one distribution rather than a mixture model. Depending on the convergence criteria of the EM algorithm, the model might fail to converge to 2 clusters if the data is actually generated from one distribution. If the algorithm converges, we end up with two components (clusters) that contain, approximately, equal number of data points.

Other clustering techniques that are capable of determining the number of clusters could be also used. For instance, an extended version of K-means [118], X-means, estimates the number of clusters (k) that best represent the data. X-means take a data and a range for the number of clusters as input, and returns a set of centroids with the best k, optimized by the Bayesian Information Criterion (BIC) model selection technique. This algorithm starts with k at the lower bound of the specified range and proceeds by recursively splitting each cluster. The BIC scores are then employed to decide upon keeping or discarding a split. The BIC formula used in [118] is also known as the Schwarz criterion and defined in [119] as the approximation of the posterior probabilities of each cluster model (with different k):

$$BIC(M_j) = l_j(D) - \frac{|\theta_{M_j}|}{2}\log(|D|), \tag{5.19}$$

where M_j is the j^{th} cluster model associated with a specific k and $|\theta_{M_j}|$ is the number of parameters in M_j. D is the set of $|D|$ data points in the analyzed feedback reports and $l_j(D)$ is the log-likelihood of D according to M_j. $l_j(D)$ is given by the sum of the log-likelihood of the set of points that belong to each of the centroids in M_j, $\sum_{c=1}^{k} l(D_c)$, where $l(D_c)$ is given by:

$$\begin{aligned} l(D_c) = -\frac{|D_c|}{2}\log(2\Pi) - \frac{|D_c|}{d}\log(\hat{\sigma}) - \frac{|D_c| - k}{2} \\ + |D_c|\log(|D_c|) - |D_c|\log(|D|), \end{aligned} \tag{5.20}$$

where D_c is the set of $|D_c|$ data points associated with the centroid c. d is the number of dimensions in D and $\hat{\sigma}$ is the maximum likelihood of the variance defined by the following equation:

$$\hat{\sigma} = \frac{1}{|D| - k} \sum_i (x_i - \mu_i)^2, \tag{5.21}$$

where μ_i is the centroid to which the data point x_i is associated.

The X-means clustering algorithms could be used to validate the results obtained by the EM-based clustering. It can also be used as a prior step to define the number of clusters that best define a given data. Since the lower bound of the range in which k resides is greater or equal to 2, we are certain to end up with at least 2 clusters. In the QoS-based reputation context, having 2 clusters means that the analyzed web service provides a QoS with a pattern that displays 2 different behaviors. However, the 2 clusters might be too close if the data is actually generated by one distribution only. Therefore, there is still a need to assess how close these 2 clusters are to decide upon keeping both clusters or grouping them in one cluster.

The similarity/distance between the 2 distributions of the corresponding components can be measured by the Kullback-Leibler (KL) divergence metric, also referred to as the relative entropy. The KL-divergence is widely used to compute the similarity between two density distributions. Since we are dealing with Gaussian distributions, the KL-divergence is expressed as follows:

$$KL(p|q) = \frac{1}{2}\left(\mathrm{Tr}(\Sigma_q^{-1}\Sigma_p) + \log\frac{\det(\Sigma_q)}{\det(\Sigma_p)} - d + (\mu_q - \mu_p)\Sigma_q^{-1}(\mu_q - \mu_p)\right), \tag{5.22}$$

where p and q are two Gaussian distributions each of which are related to one component of the mixture model. μ_p and Σ_p are the respective mean and covariance of the distribution p. $\mathrm{Tr}(A)$ is the trace of the matrix A, which is equal to the sum of its diagonal entries as well as the sum of its eigenvalues. $\det(A)$ is the determinant of the matrix A. The KL-divergence may be interpreted with the concepts of maximum likelihood and likelihood ratios. As such, this measure explains the lack of fit between a data generated by one distribution and model that encodes this data. In other words, $KL(p|q) = 0$ if and only if $p = q$, and it is greater than zero otherwise. It is also noteworthy to mention that the KL-divergence metric is not symmetric; $KL(p|q) \neq KL(q|p)$. The larger $KL(p|q)$, the further apart are p and q. Using the KL-divergence enables, for our purposes, to fuse the mixture component which distributions are shown to be very close, i.e, their KL-divergence is very small (less than a predefined value).

After having the final number of clusters, the clustered data could be further analyzed to detect and remove all possible outliers. [120] overviews various anomaly detection approaches and dedicates a section for clustering-based anomaly detection techniques. According to the authors of this survey, three main assumptions govern these techniques; the first considers anomalies (or outliers) do not belong to any cluster, the second claims that outliers are far from the centroid of their closest cluster, and the third expects outliers to form small or sparse clusters. Following the first assumption, many clustering algorithms were proposed to assign *normal* data to clusters and exclude outliers from any cluster. These include ROCK [121], FindOut [122], and SNN [123].

However, such algorithms focus on the clustering task more than on the outlier detection. Therefore, other techniques were proposed under the umbrella of the second assumption to optimize the outlier detection performance. After a clustering step, these techniques compute the distance between each data point and the centroid of its closest cluster; this is referred to as the anomaly or outlier score. Further details about the different types of anomaly detection techniques, their advantages, and disadvantages are available in [120]. Among the algorithms that follow the third assumption is FindCBLOF [124], that considers two main factors in assigning a data point an outlier score; the size of its cluster and its distance to the centroid of this cluster. The distance measure could be the same one used in the clustering algorithm. Popular measures include the Euclidean distance, Manhattan distance, and cosine similarity. In this paper, we use the clustering-based multivariate Gaussian outlier score (CMGOS) algorithm. CMGOS is an anomaly detection operator available in RapidMiner, a comprehensive open-source software platform for data mining and machine learning techniques. This algorithm calculates the outlier score of a data point by computing the covariance matrix of the cluster to which it was assigned. The sensitivity of the covariance matrix to outliers is the basis of the computed score. As described by RapidMiner's documentation, this algorithm could be perceived as a multivariate Grubb's Test [125], a statistical test to detect outliers from univariate normally distributed data.

This approach allows the analysis of the QoS behavior of a web service based on the quality it provided to all its colleagues in a community. The number of clusters is a first indicator of the stability of the quality granted by a web service. The higher the number of clusters, the less reliable the quality of the service. Afterwards, each cluster should be validated as a fair representative of the QoS behavior patterns of the service.

5.5 Experimental Evaluation

5.5.1 Model 1 Experiments

In this section, we evaluate the performance of CI and EI by running two simulations. The first implements the scenario discussed earlier in which we model the aggregation of *Ag4*'s reputation feedback sent by *Ag2* and *Ag3* to *Ag1*. The second simulation extends the same scenario to the case of 100 agents rather than 4. These simulations were executed with the following three estimators. The first, *KF*, consists of a regular Kalman filter reputation model. In this estimator, *Ag1* receives the reputation observations of *Ag4* provided by *Ag2* and *Ag3* based on which a Kalman filter is employed to predict and correct the reputation estimation. This is similar to the models proposed in [96, 98]. The second estimator, *KFCI*, encompasses two modules the first of which is a local Kalman filter (LKF) applied to obtain the reputation estimations of *Ag2* and *Ag3*. The second module employs the CI algorithm to fuse the LKF estimation with each of the estimations sent by *Ag2* and *Ag3*. The third estimator, *KFEI*, is similar to the second except CI is substituted by the EI in the second module.

The simulation setup consists of various parameters that are defined as follows. First, we assume a two dimensional reputation where each dimension represents one QoS metric, $R = \{QoS_1, QoS_2\}$. In real settings, the values of these metrics are scaled and normalized to the $[0, 1]$ range as discussed in the QoS-based Reputation Section. In this simulation, the dynamic model that represents the reputation of *Ag4* is assumed to be a vector of two time varying linear functions, one for each of QoS_1 and QoS_2. We employ the functions used in the Kalman feedback model proposed in [98]: $R = \{\log_{100}(0.02 * t + n), \log_{100}(0.03 * t + n)\}$, where t is the time step and n is a random number between 15 and 30. We used n rather than a constant to reflect the fluctuating (increasing and decreasing) nature of QoS metrics. Thus, the model consists of two unknown states that will be estimated. It also requires observations of the values of the two states supplied by *Ag2* and *Ag3*. The state and observation matrices are both set to 2×2 identity matrices.

The system covariance is set to 0.02 and the observations covariances are set to 0.01 and 0.03 for *Ag2* and *Ag3*, respectively. Figure 5.2 showa the aggregated reputation scores given by the three estimators for each of QoS_1 and QoS_2 at 100 time steps. The reputation estimates of *KFCI* and *KFEI* are much smoother and more accurate than those of *KF*. The accuracy aspect is confirmed by the mean square errors (MSE) of the three estimators displayed in the first two columns of Table 5.1.

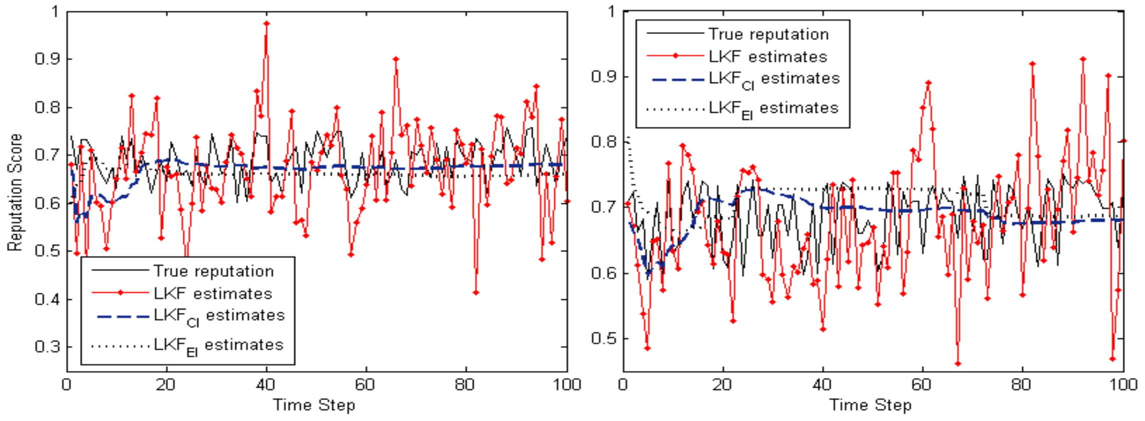


Figure 5.2: QoS_1 (left) and QoS_2 (right) Reputation scores (4 agents).

	4 agents		100 agents	
Estimator	QoS_1	QoS_2	QoS_1	QoS_2
KF	0.96	0.928	1.87	1.84
KFCI	0.525	0.526	1.17	1.5
KFEI	0.499	0.572	0.97	1.23

Table 5.1: MSE of the three estimators with 4 and 100 agents

To demonstrate the scalability of our approach, we run the same experiment with 100 agents. In this simulation, 98 agents send reputation feedback of the 99^{th} agent to the 100^{th} that aggregates them using each of *KF*, *KFCI*, and *KFEI*. Furthermore, to evaluate the robustness of these estimators, we increase the number of feedback sessions to 300 and have the dynamic model of the 99^{th} agent's reputation drop significantly half way through these sessions. The results of this simulation are displayed in Figure 5.3 and the last two columns of Table 5.1. For clarity purposes, we only show the estimations between the time steps 100 and 200. The smoothness of the *KFCI* and *KFEI* curves is noticeable in comparison with that of *KF*. These figures also show that *KFEI* detects the drop in the reputation scores faster than *KFCI* which is more conservative in changing the aggregated estimates. Table 5.1 (last 2 columns) shows that *KFEI* has the smallest MSE followed by *KFCI* and *KF*.

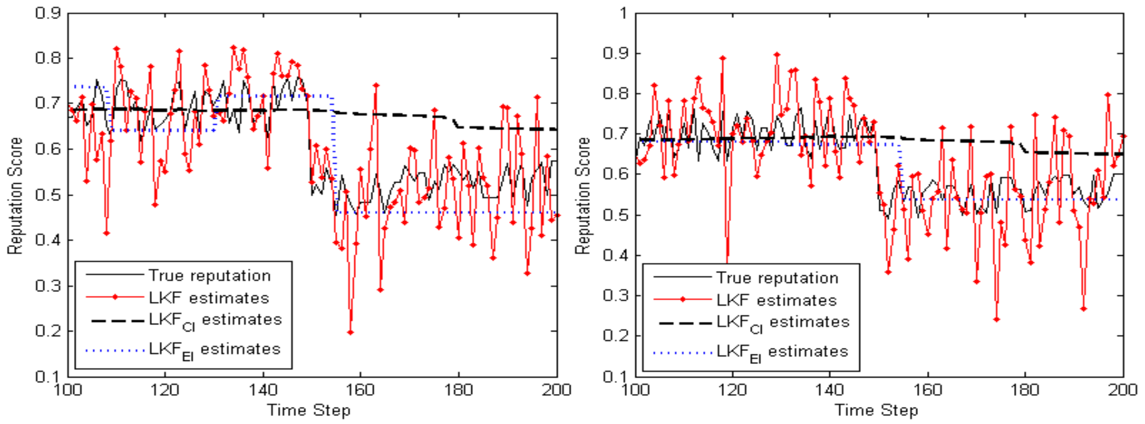
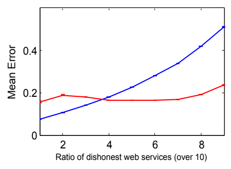


Figure 5.3: QoS_1 (left) and QoS_2 (right) Reputation scores (100 agents).

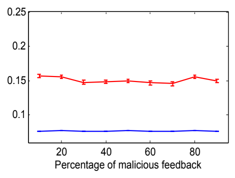
It is worthy to mention that the challenge of the CI algorithm is the selection of an appropriate w for Equations 5.8 and 5.9. We set w to $\frac{\text{Tr}(P_b)}{\text{Tr}(P_a)+\text{Tr}(P_b)}$ as proposed in [101]. As for the EI algorithm, one drawback is the extensive inversion of matrices involved in its equations. To avoid the inversion of singular matrices, we used an approximate inversion function which gave better results than those of the Kalman filter estimator.

5.5.2 Model 2 Experiments

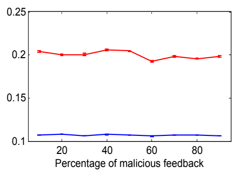
In the following experiments, we compare our reputation aggregation approach with, PeerTrust PSM, an algorithm proposed in [90] and is based on a personalized similarity measure (PSM). In PeerTrust PSM, a peer p_1 that is evaluating the reputation of another peer p_2 computes the credibility of every peer in the system. As such, the credibility of a peer p_3 is based on the similarity between the feedback about all other peers with which both p_1 and p_3 have interacted with. Two experiments have been conducted to evaluate the effectiveness of our reputation aggregation approach represented by Algorithm 3. The first evaluates the accuracy of the aggregated reputation when dealing with a variable number of dishonest web services. In this experiment, a dishonest web service reports a malicious feedback 100% of the times it responds to a reputation request. If the quality of the evaluated service provider is high, the dishonest web services report a low feedback and vice versa. The second experiment further evaluates the performance of our approach in dealing with dishonest web services by varying the rate by which these web services report malicious feedback. Due to the lack of real world datasets that report the QoS of real web services, we test our approach with a synthetic data. The simulation setup is summarized as follows:



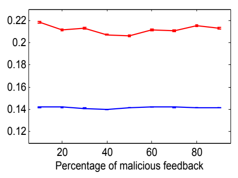
(a) 100% malicious feedback



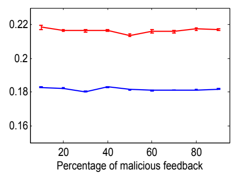
(b) 10% malicious web services



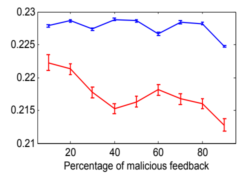
(c) 20% malicious web services



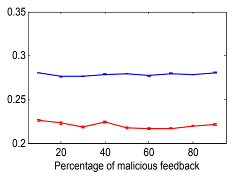
(d) 30% malicious web services



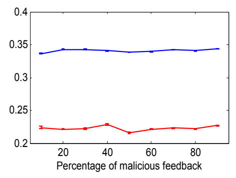
(e) 40% malicious web services



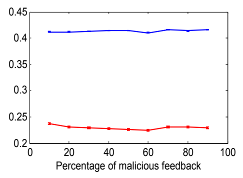
(f) 50% malicious web services



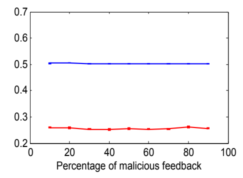
(g) 60% malicious web services



(h) 70% malicious web services



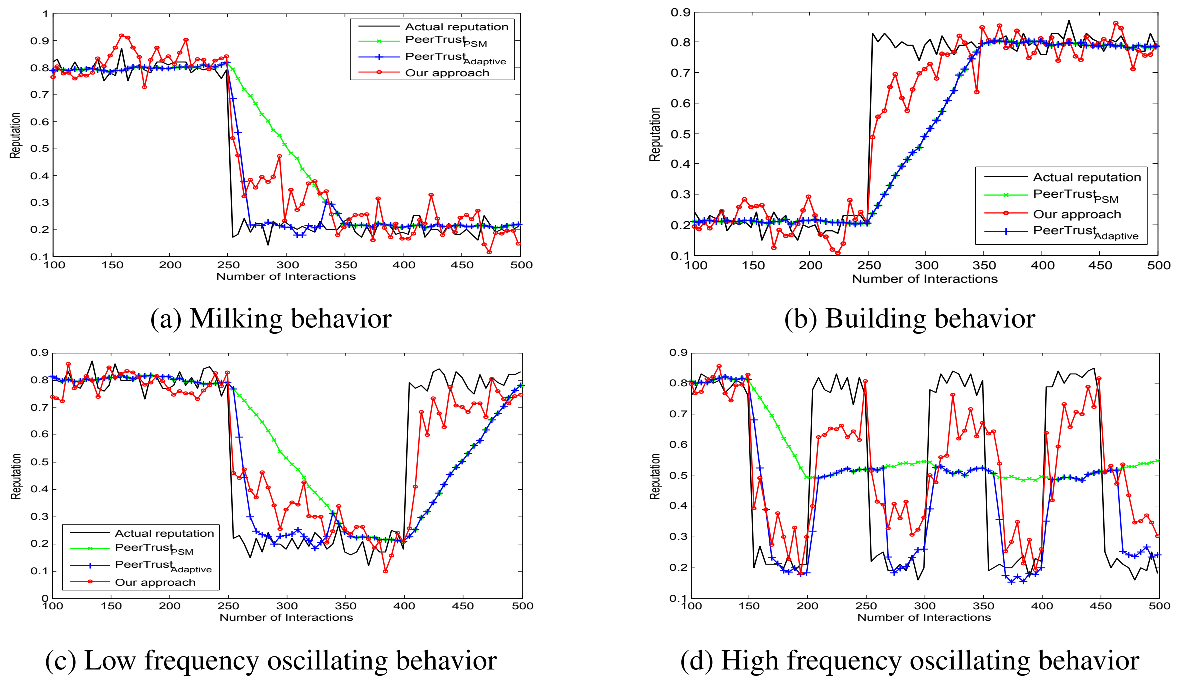
(i) 80% malicious web services



(j) 90% malicious web services

Figure 5.4: Reputation aggregation with varied ratio of malicious web services and reputation feedback

- Number of interacting web services is set to 100.
- Percentage of dishonest web services varies, in the first experiment, between 10% and 90%. In the second experiment, this is fixed to 50%.
- Percentage of malicious feedback from dishonest web services is set to 100% in the first experiment and varies between 10% and 90% in the second.
- In both experiments, we report the estimation error of the reputation of one web service as aggregated by another web service.
- We simulate the reputation of the 100 web services in 500 different time steps. At each time step, a random number of interactions occur with each of the web services.



(a) Milking behavior

(b) Building behavior

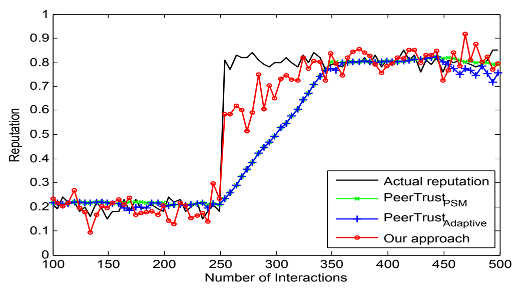
(c) Low frequency oscillating behavior

(d) High frequency oscillating behavior

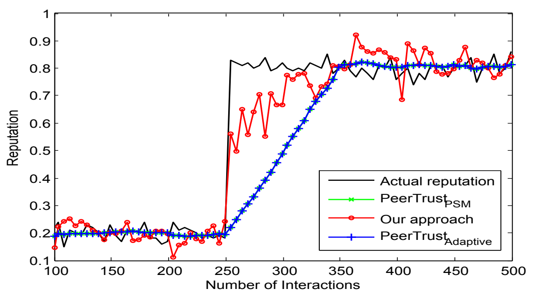
Figure 5.5: Reputation with dynamic behaviors

- The reputation is aggregated based on the feedback of interactions within a time window. We set the time window to include 5 time steps.
- The reported results are averaged over 100 runs of the experiments.

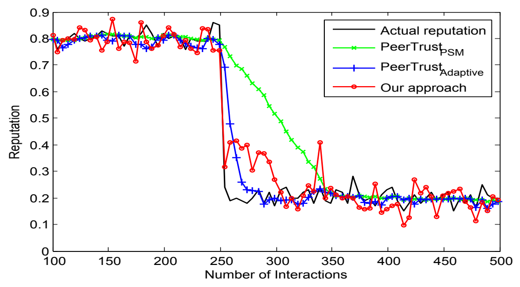
Figure 5.4 (a) displays the mean errors of the reputation aggregation based on feedback from 99 web services with varied ratio of dishonest ones. A noteworthy observation is that $PeerTrust_{PSM}$ performs slightly better than our approach when the ratio of dishonest web services is less than 40% of the total number of web services. This out-performance is not actually affected by this ratio; it is rather related to the time needed by the evaluating web service to learn the credibility of all the web services in the system. It also compares both approaches while keeping the ratio of dishonest web services fixed at 30% to 90% (b) to (j), and varying the ratio of malicious feedback provided by these web services. In cases where less than 40% of the web services exhibit malicious behavior ((b) to (e)), $PeerTrust_{PSM}$ has smaller errors than our approach. However, our approach performed better regardless of the ratio of malicious feedback in the other cases ((f) to (j)).



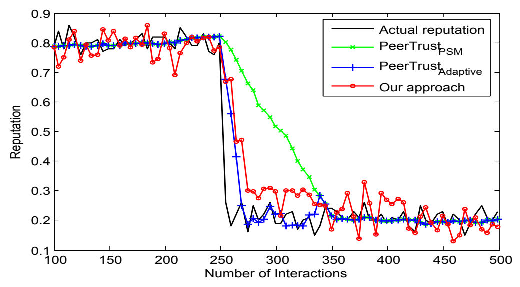
(a) Building behavior with $th = 0.01$



(b) Building behavior with $th = 0.05$



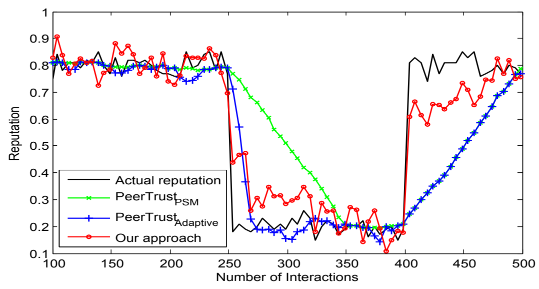
(c) Milking behavior with $th = 0.01$



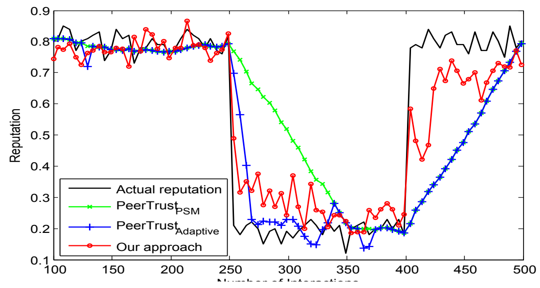
(d) Milking behavior with $th = 0.05$

Figure 5.6: Reputation with dynamic behaviors

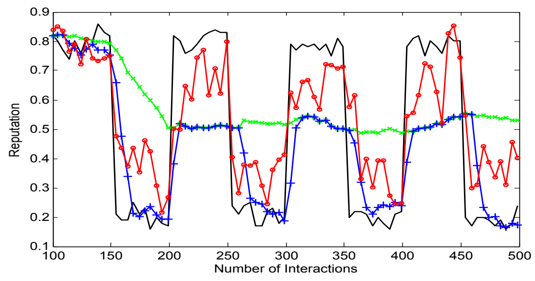
The third experiment evaluates the reputation computation of our approach when the trustee web service follows a dynamic behavior. Therefore, we compare our time-based decaying weights approach to $PeerTrust_{PSM}$ and $PeerTrust_{Adaptive}$ algorithms. For the latter, we employ a threshold $th = 0.1$ to decide which trust value to be returned; the one based on the larger (T_{win_l}) or smaller (T_{win_s}) window. Therefore, if $T_{win_l} - T_{win_s} > th$, then T_{win_s} is returned, and T_{win_l} is returned otherwise. We set the threshold to 0.1 because the authors of $PeerTrust_{Adaptive}$ used the epsilon symbol to represent this threshold. The objective of this threshold is to increase the sensitivity of the reputation model to precipitous drops in reputation values. Thus, in cases of steep reputation declines, setting the threshold to any value below 0.1 yields comparable results. The same experiment is run 4 times to handle each of the behavioral strategies discussed above with the oscillating behavior being examined twice with different quality change frequencies. The results are displayed in Figure 5.5; Figure 5.5 (a) represents the milking behavior and shows that our time decreasing weights approach captures the drop in the reputation as it happens like the 2 PeerTrust algorithms. However, $PeerTrust_{PSM}$ takes 100 interactions before it decreases the



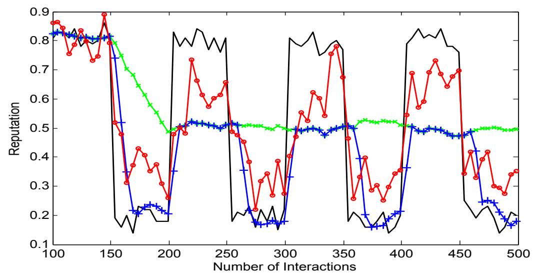
(a) Oscillating Behavior (low frequency) with $th = 0.01$



(b) Oscillating Behavior (low frequency) with $th = 0.05$



(c) Oscillating Behavior (high frequency) with $th = 0.01$



(d) Oscillating Behavior (high frequency) with $th = 0.05$

Figure 5.7: Reputation with dynamic behaviors

reputation to the actual one. $PeerTrust_{Adaptive}$ decreases the reputation down to the actual one shortly after the drop. Our approach minimizes the reputation to a low level also shortly after the collapse without going down to the lowest level of the actual reputation. This is a fair model that gives the chance for non-strategic drops to pick up faster. Figure 5.5 (b) shows the reputation of a web service that starts by providing low quality services and picks up after 250 interactions to build a higher reputation. Both PeerTrust algorithms perform similarly as their curves are coincident, and wait around 100 interactions to pick the reputation of the evaluated web service up. This is explained by the adaptive algorithm making a difference when the trust value from the smaller time window is smaller than the one from the larger time window. In the case of building reputation, 5.5 (b), the trust from the small time window is always higher which makes the adaptive and regular algorithms of PeerTrust similar. In contrast, our approach elevates the reputation of the web service shortly after it picks up its quality without reaching as high as the actual reputation.

Figure 5.5 also compares the reputation scores estimated by different approaches in case of low (c) and high (d) oscillating frequencies. In both cases, $PeerTrust_{Adaptive}$ captures the drop in the reputation value and approaches the new values within 10 to 15 interactions. Figure 5.5 (c) shows that $PeerTrust_{PSM}$ requires approximately 100 interactions before its estimations approach the actual reputation values. This algorithm fails to follow the dynamic behavior when it oscillates with high frequencies (as illustrated in Figure 5.5 (d)). In contrast, our approach was able, in both cases, to capture the drops immediately after their occurrence. It was also less harsh in penalizing the web service by waiting more than 15 interactions before decreasing the estimated reputation values to the lowest actual ones. However, both $PeerTrust_{Adaptive}$ and $PeerTrust_{PSM}$ gradually increment their estimations to reach the actual reputation value after 100 interactions when dealing with a hasty increase in the reputation value (Figure 5.5 (c)). Instead, our approach promptly follows the rise of the reputation value without reaching the actual abrupt peak. Similar observations can be marked when the oscillating frequency is every 50 interactions instead of 150 (Figure 5.5 (d)). The only difference is that $PeerTrust_{PSM}$ failed to update its estimations after the first drop in the reputation value. To further illustrate the differences between the compared approaches we use the mean absolute percentage error (MAPE) of the estimated reputations. Specifically, we compute two MAPE for each approach; one for reputations that are bigger than 0.7 and another for reputations that are smaller than 0.3. In the former case, our approach has the smallest MAPE (17.6%) while the MAPE of $PeerTrust_{Adaptive}$ and $PeerTrust_{PSM}$ are 31.2% and 28.9%, respectively. However, in the latter, $PeerTrust_{Adaptive}$ has the smallest MAPE (55.2%). The MAPE of $PeerTrust_{PSM}$ is 163.3% and the MAPE of our approach is 72.2%. The MAPE evaluation shows that our approach is less harsh in penalizing the oscillating behavior by following a reputation increase faster than the $PeerTrust$ approaches. Also, after a sudden drop in the reputation value, our approach takes a longer time to bring its estimates to the lowest reputation values. This gives the web service the benefit of the doubt that the drop might possibly not be related to malicious intentions. Furthermore, it is worth to note that the estimations using the $PeerTrust$ algorithms are smoother than the ones computed by our approach. The oscillatory aspect of the estimated reputation curve using our approach is, in part, due to the actual reputation values that don't follow a smooth curve. This is also caused by the time-based decaying weight that we used to emphasize newer reputation values and depreciate older ones. Since the actual reputation is continuously changing, the estimated values are following the same changing patterns. Moreover, our approach employs a small window to enclose the past reputation values used in the estimation of the current reputation value. The larger the window's size, the smoother the estimation curve. We repeated the same experiments with different thresholds 0.01 and 0.05 to evaluate their impact on the performance of $PeerTrust_{Adaptive}$. Figures 5.6, and 5.7 show that no significant difference can be observed between the results obtained with $th = 0.01$ and $th = 0.05$.

5.5.3 Model 3 Experiments

We evaluate, in this section, the cluster-based outlier detection approach in modeling the reputation of slave web services in a CWS. Due to the absence of real datasets with the QoS values of web services at different times, we employ synthetic data for testing purposes. We examine one CWS that groups 100 web services that interact among each other as well as with users from outside the community. However, only the quality of interactions with the services from the same community are considered in the reputation model of each web service. Let us assume that the Master of this community requests the feedback each web service maintains about its history of interactions with the other services. The amalgamation of the sampled data is clustered using EM algorithm that group data points according the most likely distribution from which they were generated. We initialize the number of clusters using the X-means clustering algorithm. Afterwards, the pairwise KL-divergence of the returned clusters are computed, and the ones with a small KL-divergence are fused. Finally, the clustered data is submitted to the CGMOS algorithm to assign each of the data points an outlier score.

Scenario 1

In this scenario, we analyze the reputation of *WS*, one of the 100 web services, based on feedback of all the remaining 99 web services. We also assume that *WS* exhibits a stable behavior and all the other web services are honest and submit fair QoS values. Therefore, we sample these

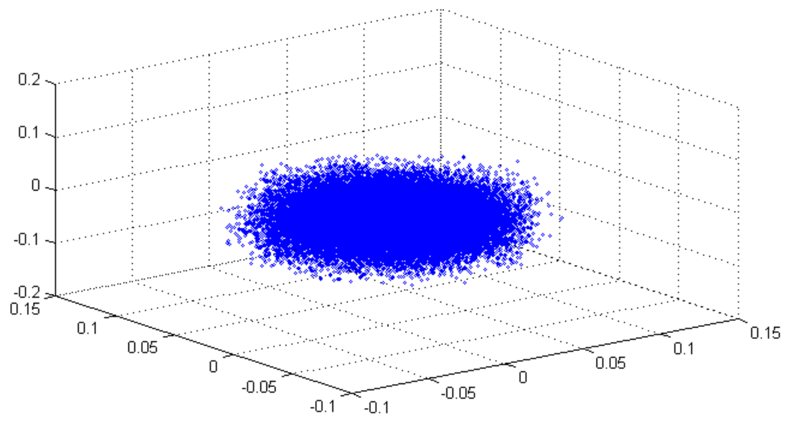


Figure 5.8: Data based on stable QoS behavior and fair feedback reports

	RT	TH	LA	AV	RE	CO	SE
Component 1 (C1)	0.8006	0.7005	0.7994	0.7283	0.7001	0.6000	0.9001
Component 2 (C2)	0.8001	0.7000	0.8004	0.6806	0.6998	0.6001	0.9000

Table 5.2: Means of the mixture components - scenario 1

feedback from the same normal distribution each of which with a random number of samples from the $[50, 500]$ range (each service has a different number of interactions with *WS*). To visualize the data, we use the principal component analysis (PCA) procedure to reduce its dimensionality from 7 to 3. The mapped data is illustrated in the Figure 5.8.

The EM algorithm revealed 2 mixture components with the means displayed in Table 5.2. This shows that these components are very similar, which is validated by calculating the KL-divergence between both components using equation 5.22; $KL(C1|C2) = 2.7465$ and $KL(C2|C1) = 7.1937$. These small values confirm the similarity between the 2 components which can be grouped in one cluster that represents the reputation profile of the analyzed web service.

Scenario 2

Let us assume a scenario similar to the one above with one main difference; the feedback provided one or more web services are not fair. This might occur when a web service is acting maliciously to elevate or degrade the reputation of its counterparts. Detecting a malicious behavior of a web service can be a challenging task especially that it might be acted according to the various scripts:

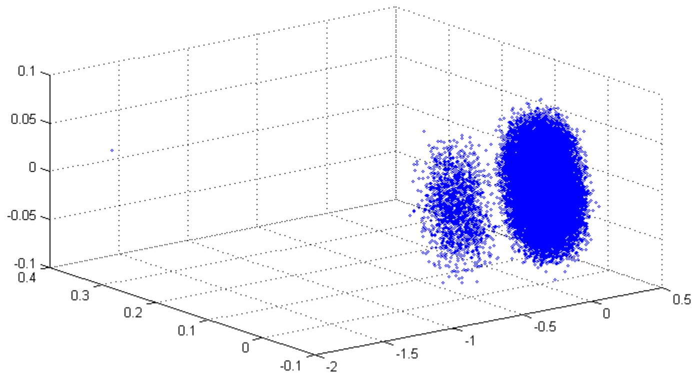


Figure 5.9: Data based on stable QoS behavior and both fair and unfair feedback reports

	RT	TH	LA	AV	RE	CO	SE
Component 1 (C1)	0.5932	0.4933	0.4949	0.3958	0.4964	0.3966	0.5941
Component 2 (C2)	0.8001	0.6999	0.8000	0.6999	0.7000	0.5998	0.8999

Table 5.3: Means of the mixture components - Scenario 2a

- a) Malicious web services report sequences of fair feedback followed by others of unfair feedback. A loop over the same script might be observed. Alternatively, they might report unfair feedback by randomly altering the values of all QoS metrics.
- b) Malicious web services report unfair feedback by altering the values of specific QoS metrics.

The sampled data for the purpose of this scenario is displayed in Figure 5.9 after being mapped to 3 dimensions using the PCA procedure. Running the EM algorithm on this data yielded two mixture components with the means displayed in Table 5.3.

Table 5.3 affirms that the 2 components are distinct and generated from 2 distributions. The KL-divergence between $C1$ and $C2$ are $KL(C1|C2) = 283.1887$ and $KL(C2|C1) = 23.8829$, which confirms the observation of two clusters in figure 5.9. However, the dispute becomes whether the web service behaves in two distinct ways or one of the cluster only exists due to unfair feedback. In reality, a web service might provide different quality depending on certain factors such as the availability of its resources, the time of the day, and the number of requests it is handling. One could argue that the size of the cluster is an indicator of the nature of its data points. As such, a large cluster represents fair feedback and a relatively very small cluster groups unfair feedback (which

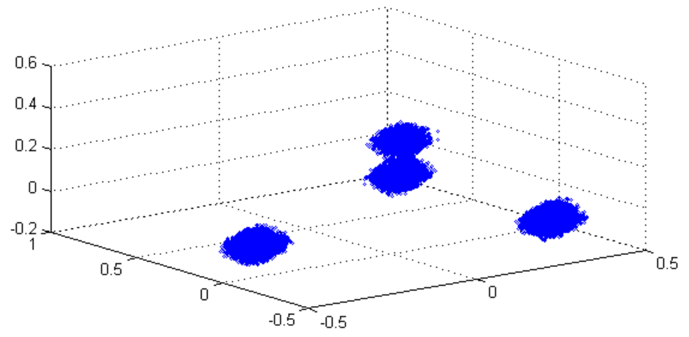


Figure 5.10: Data based on unstable QoS and both fair and unfair feedback reports

are considered outliers). However, in real scenarios, web services could behave in a consistent manner with the exception of few times due to the factors mentioned earlier. $C2$ contains the majority of the data points, 25407 (95.06%), leaving only 1320 (4.94%) in $C1$. One way to deal with this conflict would be by checking the source of the data points in the small cluster. If they were supplied by the majority of the web services then it is most probable that both cluster represent real feedback. However, if only the minority provided all the points in the small cluster, one could assume a malicious behavior. This requires adding the id of the web service to the attributes of its reputation feedback. Feeding the clustered data to the CMGOS algorithm is another validation approach. In this experiment, the outlier scores assigned to all the data points belong to the range $[0.012, 16.585]$. The top 5% outliers consist of 1319 data points from $C1$, which means that all the data points in C1 with the exception of one were detected as outliers.

Finally, we sample a data set in which 25% of the web services partially falsify their feedback reports. In other words, they alter the values of some QoS metrics leaving the remaining ones unchanged. In this experiment, the web service is assumed to behave inconsistently providing, at different times, low and high QoS values. The distribution of the sampled data after being mapped to 3 dimensions is displayed in Figure 5.10. The centroids of the three components given by the EM algorithm are shown in Table 5.4 and Figure 5.11.

Table 5.4 and Figure 5.11 show that the analyzed web service exhibits two different behaviors depicted by $C1$ (red line) and $C2$ (green line). The red and green lines follow the same pattern with the former being higher than the latter. Contrarily, the blue line show that the means of the

	RT	TH	LA	AV	RE	CO	SE
Component 1 (C1)	0.7738	0.5946	0.7997	0.5951	0.7002	0.5999	0.5000
Component 2 (C2)	0.5000	0.3998	0.3999	0.3000	0.4997	0.3999	0.3001
Component 3 (C3)	0.1999	0.6996	0.8001	0.2994	0.6998	0.6000	0.9000

Table 5.4: Means of the mixture components - scenario 2b

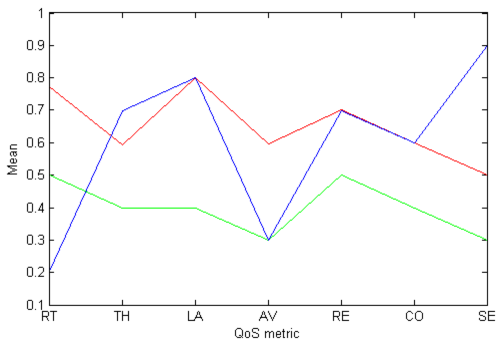


Figure 5.11: Centroids plot view

RT and SE QoS metrics have been significantly altered. It is noteworthy to mention that $C1$ and $C2$ are close in size with the former having 12489 data points and 11567 in the latter. $C3$ only contains 5478 data points. Table 5.5 shows the high values of the pairwise Kl-divergence between the 3 components.

	C1	C2	C3
C1	0	331.9950	357.6906
C2	196.9144	0	453.6297
C3	349.7278	459.2957	0

Table 5.5: KL-divergence between the 3 components

Chapter 6

Conclusions

This thesis presented various solutions to the web service selection problem based on probabilistic trust and reputation approaches. Chapter 1 described the web service environment and the web service selection problem. The literature of trust and reputation systems was reviewed in Chapter 2 with a focus on probabilistic approaches. The remaining chapters proposed different trust and reputation models that are based on probabilistic approaches. These models are mainly motivated by the Bayesian learning of proportional or count data, especially the employment of different conjugate priors such as Dirichlet, generalized Dirichlet and Beta-Liouville. The main conclusions drawn in this thesis are:

QoS-Based Trust Models The trust models proposed in this thesis are based on the probabilistic modeling of the outcomes of various QoS metrics. Three composite distributions were introduced to represent the quality classes of the web services outcomes; multinomial Dirichlet, multinomial generalized Dirichlet, and multinomial Beta Liouville. The Bayesian inference method was then employed to estimate the probability that the quality of the evaluated web service belong to each of the considered classes. Using these probabilities, a weighted function was employed to compute the trustworthiness of this web service.

Correlation Model We further extended our trust model to incorporate the correlations among the QoS metrics. This extension allows the model to provide, along the computed trust scores, accurate estimates of the uncertainties in these scores. Using two real data sets of web services QoS, we demonstrated and evaluated the correlations that exist between various pairs of metrics. Afterwards, we introduced a hybrid trust model that captures the correlations among the QoS metrics based on two conjugate priors to the multinomial that represents the pair-wise combined QoS outcomes; generalized Dirichlet and Dirichlet distributions. The former was shown to better model positive correlations while the latter yields better modeling of negative correlations. The empirical results endorse the effectiveness of our approach. We then presented an algorithm inspired by anomaly detection algorithms to aggregate reputation feedback and deal with unfair referrals. We further extended this algorithm by adding time decreasing weights to the reputation feedback to capture the dynamic behavior of some web services. In comparison to a well-known reputation system, PeerTrust, our algorithm showed very promising results. We will extend, in a future work, the pairwise correlations modeling to multi-dimensional correlations at a time rather than pairwise correlations.

Bayesian Network Models We also proposed a BN model for a QoS-based classification of composite web services. We employed new priors to the distributions of the networks variables; generalized Dirichlet and Beta-Liouville. The BN classifier with the Beta-Liouville prior also provided a relatively high classification accuracy which is slightly higher than the accuracy of the BN classifier with a Dirichlet prior. This classification provides a way to assess the quality of composite web services during the composition task based on the quality of the constituent services.

Online Trust Models Two online approaches were introduced to deal with the estimation of QoS-based trust scores. The first is a time-series approach built upon a generalized-Dirichlet state space model to predict the future QoS ratings of web services based on their last ratings. Substituting the the Dirichlet time series model by a generalized-Dirichlet model, that is transformed into multiple Dirichlet model in lower dimensional space, proved to enhance the predictions in terms of standardized residuals and predictions MSE. This approach was tested with simulated data due to unavailability of QoS ratings in the form of compositional time-series. An extension of this approach would be to first test it with real data that could be collected by monitoring the QoS performance of real web services over a significant period of time (one week). Another extension might include the addition of new parameters to model the time-series trends and covariates. The second online model is based on a BN model that applied the Voting EM update rule to learn the conditional probabilities of the networks variables based one data instance. This models experimental results showed the convergence of the learned parameters to the true probabilities. We acknowledge the limitation of the online approach in learning the BN parameters when the QoS ratings of the web services are not completely observable. Also, since the choice of the learning rate is not trivial, a variable learning rate can be adopted instead.

Reputation Models This thesis proposed three different reputation aggregation models. The first is designed to deal with unfair referrals and web services dynamic behaviors. It considers the credibility of the feedback senders as an additional criterion to confirm the outliers detected by an LDCOF algorithm. The model also handles four different malicious behaviors a web service may exhibit to maximize its own interests. This is achieved using a time window that will assign higher weights to newer transactions. As such, any abrupt drop or rise in the delivered quality is penalized by reflecting them in the computed trust scores. The second model aims to aggregate reputation feedback while ensuring consistent estimates. Two data fusion method were employed for this purpose, namely, the covariance and ellipsoidal intersection. These methods will avoid over and under estimated confidence in the combined reputation estimates. The third is a QoS-based reputation model for web services grouped into communities that administer homogeneous functionalities. This model implements various clustering and anomaly detection algorithms to grant the masters of these communities the capability of characterizing the quality of their web services. We argue that this approach improves the satisfaction of the service consumers by assigning their requests to the ones that best suit their quality requirements. This approach could be further validated through experiments with real data and comparisons with other approaches in a future work.

List of References

[1] Babak Khosravifar, Jamal Bentahar, and Ahmad Moazin. Analyzing the relationships between some parameters of web services reputation. In *Proceedings of the IEEE International Conference on Web Services (ICWS)*, pages 329–336, 2010.

[2] Babak Khosravifar, Jamal Bentahar, Ahmad Moazin, Zakaria Maamar, and Philippe Thiran. Analyzing communities vs. single agent-based web services: Trust perspectives. In *Proceedings of the IEEE International Conference on Services Computing (SCC)*, pages 194–201, 2010.

[3] Kwang Mong Sim. Agent-based cloud computing. *IEEE Transactions on Services Computing*, 5(4):564–577, 2012.

[4] Minder Chen, Andrew N. K. Chen, and Benjamin B. M. Shao. The implications and impacts of web services to electronic commerce research and practices. *J. Electron. Commerce Res.*, pages 128–139, 2003.

[5] Sarvapali D. Ramchurn, Dong Huynh, and Nicholas R. Jennings. Trust in multi-agent systems. *Knowl. Eng. Rev.*, 19:1–25, March 2004.

[6] Sanjiva Weerawarana, Francisco Curbera, Frank Leymann, Tony Storey, and Donald F. Ferguson. *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More*. Prentice Hall PTR, Upper Saddle River, NJ,

USA, 2005.

[7] Shirin Sohrabi and Sheila A. McIlraith. Preference-based web service composition: A middle ground between execution and search. In *Proceedings of the 9th International Semantic Web Conference on The Semantic Web - Volume Part I*, ISWC'10, pages 713–729, Berlin, Heidelberg, 2010. Springer-Verlag.

[8] Hamdi Yahyaoui. A trust-based game theoretical model for web services collaboration. *Knowledge-Based Systems*, 27:162–169, 2012.

[9] Yonghong Wang and Munindar P. Singh. Formal trust model for multiagent systems. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 1551–1556, San Francisco, CA, USA, 2007.

[10] Sarvapali D. Ramchurn, Dong Huynh, and Nicholas R. Jennings. Trust in multi-agent systems. *Knowl. Eng. Rev.*, 19(1):1–25, March 2004.

[11] Lik Mui, Mojdeh Mohtashemi, and Ari Halberstadt. Notions of reputation in multi-agents systems: a review. In *Proceedings of the international conference on AAMAS*, pages 280–287, 2002.

[12] Yao Wang and Julita Vassileva. Toward trust and reputation based web service selection: A survey, 2007.

[13] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, 2006.

[14] Donovan Artz and Yolanda Gil. A survey of trust in computer science and the semantic web. *Web Semant.*, 5(2):58–71, 2007.

[15] Jordi Sabater and Carles Sierra. Review on computational trust and reputation models. *Artificial Intelligence Review*, 24:33–60, 2005.

[16] Isaac Pinyol and Jordi Sabater-Mir. Computational trust and reputation models for open multi-agent systems: a review. *Artificial Intelligence Review*, July 2011.

[17] T. Balke, S. Knig, and T. Eymann. A survey on reputation systems for artificial societies. *Bayreuth Reports on Information Systems Management*, 2009.

[18] Chung-Wei Hang and Munindar P. Singh. Trustworthy service selection and composition. *ACM Transactions on Autonomous and Adaptive Systems*, 6(1):5:1–5:17, 2011.

[19] Yao Wang and Julita Vassileva. Bayesian network trust model in peer-to-peer networks. In *In Proceedings of Second International Workshop Peers and Peer-to-Peer Computing*, pages 23–34, 2003.

[20] Achim Rettinger, Matthias Nickles, and Volker Tresp. Statistical relational learning of trust. *Machine Learning*, 82(2):191–209, 2010.

[21] W. T. Luke Teacy, Jigar Patel, Nicholas R. Jennings, Michael Luck, and Multiagent Systems. Coping with inaccurate reputation sources: Experimental analysis of a probabilistic trust model. In *AAMAS 05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 997–1004, 2005.

[22] Lei Li and Yan Wang. A subjective probability based deductive approach to global trust evaluation in composite services. In *ICWS*, pages 604–611. IEEE Computer Society, 2011.

[23] Giorgos Zacharia, Alexandros Moukas, Ros Moukas, and Pattie Maes. Collaborative reputation mechanisms in electronic marketplaces. In *in HICSS*, page 8026, 1999.

[24] L.H Vu, Manfred Hauswirth, and Karl Aberer. Qos-based service selection and ranking with trust and reputation management. In *Proceedings of the Cooperative Information System Conference*, pages 446–483, 2005.

[25] Alfarez Abdul-Rahman and Stephen Hailes. Supporting trust in virtual communities. *Hawaii International Conference on System Sciences*, 6:6007, 2000.

[26] Babak Esfandiari and Sanjay Chandrasekharan. On how agents make friends: Mechanisms for trust acquisition, 2001.

[27] Bin Yu and Munindar P. Singh. Detecting deception in reputation management, 2003.

[28] Mao Chen. Computing and using reputations for internet ratings. In *In Proceedings of the 3rd ACM Conference on Electronic Commerce*, pages 154–162. ACM Press, 2001.

[29] Chrysanthos Dellarocas Sloan. Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior, 2000.

[30] Andrew Whitby, Audun Jsang, and Jadwiga Indulska. Filtering out unfair ratings in bayesian reputation systems. 2004.

[31] Fabrizio Cornelli, Ernesto Damiani, Sabrina De Capitani, Stefano Paraboschi, and Pierangela Samarati. Choosing reputable servents in a p2p network. In *In Proceedings of the 11th World Wide Web Conference*, pages 376–386, 2002.

[32] Xiaofeng Wang, Ling Liu, and Jinshu Su. Rlm: A general model for trust representation and aggregation. *IEEE Transactions on Services Computing*, 5(1):131–143, 2012.

[33] Ehsan Khosrowshahi Asl, Jamal Bentahar, Hadi Otrok, and Rabeb Mizouni. Efficient coalition formation for web services. In *Proceedings of the 2013 IEEE International Conference on Services Computing*, SCC '13, pages 737–744, Washington, DC, USA, 2013. IEEE Computer Society.

[34] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[35] N. Bouguila. Clustering of count data using generalized dirichlet multinomial distributions. *Knowledge and Data Engineering, IEEE Transactions on*, 20(4):462 –474, april 2008.

[36] Rasmus E. Madsen, David Kauchak, and Charles Elkan. Modeling word burstiness using the dirichlet distribution. In *Proceedings of the 22nd international conference on Machine learning*, pages 545–552, New York, NY, USA, 2005.

[37] James E. Mosimann. On the compound multinomial distribution, the multivariate - distribution, and correlations among proportions. *Biometrika*, 49(1/2):pp. 65–82, 1962.

[38] N. Bouguila. Count data modeling and classification using finite mixtures of distributions. *Neural Networks, IEEE Transactions on*, 22(2):186 –198, feb. 2011.

[39] Robert H. Lochner. A generalized dirichlet distribution in bayesian life testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 37(1):pp. 103–113, 1975.

[40] Robert J. Connor and James E. Mosimann. Concepts of independence for proportions with a generalization of the dirichlet distribution. *Journal of the American Statistical Association*, 64(325):pp. 194–206, 1969.

[41] N. Bouguila, D. Ziou, and E. Monga. Practical bayesian estimation of a finite beta mixture through gibbs sampling and its applications. *Statistics and Computing*, 16(2):215–225, 2006.

[42] Mohamad Mehdi, Nizar Bouguila, and Jamal Bentahar. Trustworthy web service selection using probabilistic models. In *ICWS*, pages 17–24. IEEE, 2012.

[43] David M. Blei, Andrew Y. Ng, Michael I. Jordan, and John Lafferty. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:2003, 2003.

[44] N. Bouguila, D. Ziou, and J. Vaillancourt. Unsupervised learning of a finite mixture model based on the dirichlet distribution and its application. *Image Processing, IEEE Transactions on*, 13(11):1533 –1543, nov. 2004.

[45] J. Aitchison. The Statistical Analysis of Compositional Data. *Journal of the Royal Statistical Society. Series B (Methodological)*, 44(2):139–177, 1982.

[46] Brunsdon Teresa M. and Smith T.M.F. The time series analysis of compositional data. *Journal of Official Statistics*, 14(3):pp. 237–253, 1998.

[47] G. C. Tiao and G. E. P. Box. Modeling multiple times series with applications. *Journal of the American Statistical Association*, 76(376):pp. 802–816, 1981.

[48] J. M. Quintana and M. West. Time series analysis of compositional data. *Bayesian Statistics 3*, 1988.

[49] Huiwen Wang, Qiang Liu, Henry M. K. Mok, Linghui Fu, and Wai Man Tse. A hyperspherical transformation forecasting model for compositional data. *European Journal of Operational Research*, 179(2):459–468, 2007.

[50] Amitabha Bhaumik, Dipak K. Dey, and Nalini Ravishanker. Joint statistical meetings-bayesian statistical science time series analysis of compositional data using a dynamic linear model approach, 1999.

[51] Jan G De Gooijer and Rob J Hyndman. 25 years of time series forecasting. *International Journal of Forecasting*, 2006.

[52] Gary K. Grunwald, Adrian E. Raftery, and Peter Guttorp. Time Series of Continuous Proportions. *Journal of the Royal Statistical Society, Series B*, 55:103–116, 1993.

[53] Tzu-Tsung Wong. Parameter estimation for generalized Dirichlet distributions from the sample estimates of the first and the second moments of random variables. *Computational Statistics and Data Analysis*, 54(7):1756 – 1765, 2010.

[54] N. Bouguila and D. Ziou. High-dimensional unsupervised selection and estimation of a finite generalized dirichlet mixture model based on minimum message length. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(10):1716–1731, Oct 2007.

[55] Christian P. Robert. *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*. Springer, 2nd edition, May 2007.

[56] P. J. Harrison and C. F. Stevens. Bayesian forecasting. *Journal of the Royal Statistical Society. Series B (Methodological)*, 38(3):pp. 205–247, 1976.

[57] J. Q. Smith. A generalization of the bayesian steady forecasting model. *Journal of the Royal Statistical Society. Series B (Methodological)*, 41(3):pp. 375–387, 1979.

[58] J. Q. Smith. The multiparameter steady model. *Journal of the Royal Statistical Society. Series B (Methodological)*, 43(2):pp. 256–260, 1981.

[59] S. Boutemedjet, N. Bouguila, and D. Ziou. A hybrid feature extraction selection approach for high-dimensional non-gaussian data clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(8):1429–1443, Aug 2009.

[60] Luke Tierney and Joseph B. Kadane. Accurate Approximations for Posterior Moments and Marginal Densities. *Journal of the American Statistical Association*, 81(393):82–86, March 1986.

[61] P. Diaconis and D. Ylvisaker. Conjugate Priors for Exponential Families. *The Annals of Statistics*, 7(2), 1979.

[62] Steven Reece, Alex Rogers, Stephen Roberts, and Nicholas R. Jennings. Rumours and reputation: evaluating multi-dimensional trust within a decentralised reputation system. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, AAMAS '07, pages 165:1–165:8, New York, NY, USA, 2007. ACM.

[63] Liangzhao Zeng, Hui Lei, and Henry Chang. Monitoring the qos for web services. In *Proceedings of the 5th International Conference on Service-Oriented Computing*, ICSOC '07, pages 132–144, Berlin, Heidelberg, 2007. Springer-Verlag.

[64] Mohamad Mehdi, Nizar Bouguila, and Jamal Bentahar. Probabilistic approach for qos-aware recommender system for trustworthy web service selection. *Applied Intelligence*, 41(2):503–524, 2014.

[65] Molood Makhlughian, Seyyed Mohsen Hashemi, Yousef Rastegari, and Emad Pejman. Web service selection based on ranking of qos using associative classification. *CoRR*, abs/1204.1425, 2012.

[66] Xumin Liu, Athman Bouguettaya, Qi Yu, and Zaki Malik. Efficient change management in long-term composed services. *Serv. Oriented Comput. Appl.*, 5:87–103, June 2001.

[67] David Heckerman. A tutorial on learning with bayesian networks. Technical report, Learning in Graphical Models, 1995.

[68] C. I¡. Chow, Sexior Member, and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968.

[69] Gregory F. Cooper and Tom Dietterich. A bayesian method for the induction of probabilistic networks from data. In *Machine Learning*, pages 309–347, 1992.

[70] Kevin P. Murphy. Active learning of causal bayes net structure. Technical report, 2001.

[71] Nir Friedman. The bayesian structural em algorithm, 1998.

[72] Olivier François. Bayesian network structural learning and incomplete data, 2005.

[73] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society, series B*, 39(1):1–38, 1977.

[74] D. Heckerman, D. Geiger, and D.M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. 1994.

[75] Wray Buntine. Theory refinement on bayesian networks. pages 52–60. Morgan Kaufmann, 1991.

[76] N L Zhang and D Poole. A simple approach to bayesian network computations. In *Proceedings of the Tenth Canadian Conference on Artificial Intelligence*, pages 171–178, Canada, 1994.

[77] F V Jensen, S L Lauritzen, and K G Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, 4(4):269–282, 1990.

[78] R.L. Winkler. The assessment of prior distributions in bayesian analysis. *Journal of the American Statistical association*, page 776800, 1967.

[79] Re Bronstein, Ira Cohen, Fabio G. Cozman, Fabio G. Cozman, Escola Politecnica, and Alexandre Bronstein Fabio. Adaptive online learning of bayesian network parameters, 2001.

[80] Chung-Wei Hang, Anup K. Kalia, and Munindar P. Singh. Behind the curtain: Service selection via trust in composite services. In *Proceedings of the IEEE International Conference on Web Services (ICWS)*, pages 9–16, Honolulu, HI, June 2012.

[81] Eric Bauer, Daphne Koller, and Yoram Singer. Update rules for parameter estimation in bayesian networks. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 3–13. Morgan Kaufmann, 1997.

[82] N. Bouguila and D. Ziou. Improving content based image retrieval systems using finite multinomial dirichlet mixture. In *Machine Learning for Signal Processing, 2004. Proceedings of the 2004 14th IEEE Signal Processing Society Workshop*, pages 23 –32, 29 2004-oct. 1 2004.

[83] N. Bouguila. Clustering of count data using generalized dirichlet multinomial distributions. *Knowledge and Data Engineering, IEEE Transactions on*, 20(4):462–474, 2008.

[84] E. Al-Masri and Q.H. Mahmoud. Qos-based discovery and ranking of web services. In *Proc. 16th International Conference on Computer Communications and Networks (ICCCN' 2007)*, pages 529–534, Honolulu, HI, August 2007.

[85] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *In AAAI-98 Workshop On Learning For Text Categorization*, pages 41–48, 1998.

[86] T. Minka. *Estimating a Dirichlet distribution*. Technical report, MIT, 2000.

[87] M. Mehdi, N. Bouguila, and J. Bentahar. A qos-based trust approach for service selection and composition via bayesian networks. In *Web Services (ICWS), 2013 IEEE 20th International Conference on*, pages 211–218, 2013.

[88] Kevin Hoffman, David Zage, and Cristina Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Comput. Surv.*, 42(1):1:1–1:31, December 2009.

[89] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, pages 640–651, New York, NY, USA, 2003. ACM.

[90] Li Xiong and Ling Liu. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Transactions On Knowledge And Data Engineering*, 16:843–857, 2004.

[91] Yonghong Wang and Munindar P. Singh. Trust representation and aggregation in a distributed agent system. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, pages 1425–1430. AAAI Press, 2006.

[92] Chung-Wei Hang, Yonghong Wang, and Munindar P. Singh. Operators for propagating trust and their evaluation in social networks. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '09, pages 1025–1032, Richland, SC, 2009.

[93] Bled Electronic Commerce, Audun Jøsang, and Roslan Ismail. The beta reputation system. In *In Proceedings of the 15th Bled Electronic Commerce Conference*, 2002.

[94] Audun Josang and Jochen Haller. Dirichlet reputation systems. In *Proceedings of the The Second International Conference on Availability, Reliability and Security*, ARES '07, pages 112–119, Washington, DC, USA, 2007. IEEE Computer Society.

[95] Bo Ye, Anjum Pervez, Mohammad Ghavami, and Maziar Nekovee. A trust-based model for quality of web service. In *International Conference on Advanced Service Computing*, pages 39–45, Valencia, Spain, 2013.

[96] Xiaofeng Wang, Ling Liu, and Jinshu Su. Rlm: A general model for trust representation and aggregation. *IEEE Transactions on Service Computing*, 5(1):131–143, January 2012.

[97] Ziqiang Xu, Patrick Martin, Wendy Powley, and Farhana H. Zulkernine. Reputation-enhanced qos-based web services discovery. In *ICWS*, pages 249–256. IEEE Computer Society, 2007.

[98] Huan Zhou, Xiaofeng Wang, and Jinshu Su. A general self-adaptive reputation system based on the kalman feedback. In *International Conference on Service Sciences*, pages 7–12, Los Alamitos, CA, USA, 2013.

[99] Lingji Chen, Pablo O. Arambel, and Raman K. Mehra. Estimation under unknown correlation: covariance intersection revisited. *IEEE Transactions on Automatic Control*, 47(11):1879–1882, 2002.

[100] Simon J. Julier and Jeffrey K. Uhlmann. A non-divergent estimation algorithm in the presence of unknown correlations. In *In Proceedings of the American Control Conference*, pages 2369–2373, 1997.

[101] W. Niehsen. Information fusion based on fast covariance intersection filtering. In *Proceedings of the Fifth International Conference on Information Fusion*, volume 2, pages 901–904, 2002.

[102] S. J. Julier and J. K. Uhlmann. *General Decentralized Data Fusion with Covariance Intersection*. Handbook of Multisensor Data Fusion. CRC Press, Boca Raton, FL, 2001.

[103] J. Sijs, M. Lazar, and P.P.J. v.d.Bosch. State fusion with unknown correlation: Ellipsoidal intersection. In *American Control Conference (ACC), 2010*, pages 3992–3997, 2010.

[104] N. Amer and M. Goldstein. Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer. In *Proceedings of the 3rd RapidMiner Community Meeting and Conference (RCOMM)*, pages 1–12, 2012.

[105] Yao Wang, Jie Zhang, and Julita Vassileva. Effective web service selection via communities formed by super-agents. In Jimmy Xiangji Huang, Irwin King, Vijay V. Raghavan, and Stefan Rueger, editors, *Web Intelligence*, pages 549–556. IEEE, 2010.

[106] Zakaria Maamar, Mohammed Lahkim, Djamal Benslimane, Philippe Thiran, and Sattanathan Subramanian. Web services communities - concepts & operations. In Joaquim Filipe, Jos Cordeiro, Bruno Encarnao, and Vitor Pedrosa, editors, *WEBIST (1)*, pages 323–327. INSTICC Press, 2007.

[107] Jamal Bentahar, Zakaria Maamar, Wei Wan, Djamal Benslimane, Philippe Thiran, and Sattanathan Subramanian. Agent-based communities of web services: an argumentation-driven approach. *Service Oriented Computing and Applications*, 2(4):219–238, 2008.

[108] Zakaria Maamar, Sattanathan Subramanian, Philippe Thiran, Djamal Benslimane, and Jamal Bentahar. An approach to engineer communities of web services: Concepts, architecture, operation, and deployment. *IJEBR*, 5(4):1–21, 2009.

[109] Abdelghani Benharref, M. Adel Serhani, Salah Bouktif, and Jamal Bentahar. A new approach for quality enforcement in communities of web services. In *Proceedings of the 2011 IEEE International Conference on Services Computing*, SCC '11, pages 472–479, Washington, DC, USA, 2011. IEEE Computer Society.

[110] Yao Wang and J. Vassileva. A review on trust and reputation for web service selection. In *Distributed Computing Systems Workshops, 2007. ICDCSW '07. 27th International Conference on*, pages 25–25, June 2007.

[111] Murat Sensoy, Burcu Yilmaz, and TimothyJ. Norman. Discovering frequent patterns to bootstrap trust. In Longbing Cao, Yifeng Zeng, AndreasL. Symeonidis, VladimirI. Gorodetsky, PhilipS. Yu, and MunindarP Singh, editors, *Agents and Data Mining Interaction*, volume 7607 of *Lecture Notes in Computer Science*, pages 93–104. Springer Berlin Heidelberg, 2013.

[112] Erbin Lim and Philippe Thiran. Sustaining high-availability and quality of web services. In Florian Daniel and FedericoMichele Facca, editors, *Current Trends in Web Engineering*, volume 6385 of *Lecture Notes in Computer Science*, pages 560–565. Springer Berlin Heidelberg, 2010.

[113] Said Elnaffar, Zakaria Maamar, Hamdi Yahyaoui, Jamal Bentahar, and Philippe Thiran. Reputation of communities of web services - preliminary investigation. In *AINA Workshops*, pages 1603–1608. IEEE Computer Society, 2008.

[114] B. Khosravifar, J. Bentahar, P. Thiran, A. Moazin, and A. Guiot. An approach to incentive-based reputation for communities of web services. In *Web Services, 2009. ICWS 2009. IEEE International Conference on*, pages 303–310, July 2009.

[115] Carlo Ghezzi and Sam Guinea. Run-time monitoring in service-oriented architectures. *Test and Analysis of Web Services*, pages 237–264, 2007.

[116] Manish Gupta, Jing Gao, Charu C. Aggarwal, and Jiawei Han. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 99:1–20, 2013.

[117] G.J. McLachlan and K.E. Basford. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, New York, 1988.

[118] Dan Pelleg and Andrew W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 727–734, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[119] Robert E. Kass and Larry Wasserman. A reference bayesian test for nested hypotheses and its relationship to the schwarz criterion. *Journal of the American Statistical Association*, 90(431):928–934, 1995.

[120] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[121] Rock: A robust clustering algorithm for categorical attributes. In *Proceedings of the 15th International Conference on Data Engineering*, ICDE '99, pages 512–, Washington, DC, USA, 1999. IEEE Computer Society.

[122] Dantong Yu, Gholamhosein Sheikholeslami, and Aidong Zhang. Findout: Finding outliers in very large datasets. *Knowl. Inf. Syst.*, 4(4):387–412, October 2002.

[123] Levent Ertöz, Michael Steinbach, and Vipin Kumar. Finding topics in collections of documents: A shared nearest neighbor approach. In *Clustering and Information Retrieval*, volume 11 of *Network Theory and Applications*, pages 83–103. Springer US, 2004.

[124] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(910):1641 – 1650, 2003.

[125] Frank E. Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, February 1969.