Réseaux de service web : construction, analyse et applications

Hafida Naim

Thèse présentée pour obtenir le grade de docteur

Université Aix-Marseille

Discipline : Informatique

Réseaux de services web : construction, analyse et applications

Par : **Hafida NAIM**

Membres du jury :

Rapporteur : Claude GODART, Professeur, LORIA, Université de Lorraine.

Rapporteur : Samir TATA, Professeur, Télécom SudParis, Institut Mines-Télécom.

Examinateur : Daniela GRIGORI, Professeur, LAMSADE, Université Paris Dauphine.

Examinateur : Djamal BENSLIMANE, Professeur, LIRIS, Université Lyon 1.

Directeur : Mohamed QUAFAFOU, Professeur, LSIS, Université Aix-Marseille.

Co-Directeur : Nicolas DURAND, Maître de conférences, LSIS, Université Aix-Marseille.

Date de soutenance : 13/12/2017

Laboratoire des Sciences de l'Information et des Systèmes (LSIS UMR 7296)
Ecole Doctorale en Mathématiques et Informatique de Marseille - ED 184

Remerciements

Je souhaite exprimer ici ma gratitude et reconnaissance à ceux et celles qui ont contribués de près ou de loin à l'accomplissement de cette thèse. Je remercie **Dieu** pour son aide à travers toutes ces personnes. Sans le soutien et l'aide de toutes ces personnes, cette thèse n'aurait certainement pas été possible car il n'y a pas de travail doctoral solitaire.

Je tiens à remercier très chalheureusement mon directeur de thèse, Monsieur **Mohamed QUAFAFOU**, Professeur à l'Université d'Aix-Marseille, pour la confiance qu'il m'a témoignée en m'accueillant dans son laboratoire dans le cadre de cette thèse de doctorat non financée et en acceptant la direction scientifique de mes travaux. Je suis vraiment incapable de trouver les mots adéquats pour lui exprimer ma reconnaissance et ma gratitude. J'ai beaucoup appris à ses côtés. Il est l'exemple de chercheur passionné que je souhaite devenir un jour. Je le remercie pour avoir guidé mes premiers pas dans la recherche. Je le remercie également pour sa rigueur scientifique, son savoir-faire, ses conseils, sa disponibilité, sa patience et son dynamisme qui m'ont été très bénéfiques pour faire avancer cette thèse. J'ai été extrêmement sensible à sa qualité humaine, sa compréhension et son soutien qui m'ont permis de dépasser des conditions personnelles qui n'ont pas été toujours favorables. Il était très arrangeant en me laissant travailler chez moi quand je ne pouvais pas venir au bureau, et je l'en remercie vivement. Je le remercie pour tout ce qui'il a fait pour moi tout au long de cette thèse.

Je tiens également à exprimer mon immense gratitude à mon co-directeur de thèse, Monsieur **Nicolas DURAND**, Maitre de conférences à l'Université Aix-Marseille, pour son soutien, ses encouragements, ses conseils et ses remarques qui ont été bénéfiques au cours de ces années. Son encadrement et son investissement dans cette thèse m'ont été d'une aide très précieuse. Je le remercie également pour sa grande disponibilité pour la relecture des documents que je lui ai adressés. J'ai beaucoup apprécié son sérieux et ses qualités humaines tout au long de cette thèse. Qu'il trouve ici l'expression de mon profond respect.

J'adresse également mes sincères remerciement aux rapporteurs de ma thèse, Monsieur **Claude GODART**, Professeur à l'Université de Lorraine et Monsieur **Samir TATA**, Professeur à Télécom SudParis, pour l'honneur qu'ils m'ont accordé en acceptant de rapporter et d'évaluer mon travail et pour leurs commentaires très constructifs et enrichissants.

Je remercie très vivement Madame **Daniela GRIGORI**, Professeur à l'Université Paris Dauphine, et Monsieur **Djamal BENSLIMANE**, Professeur, à l'Université Lyon 1, d'avoir accepté de faire partie de mon jury en tant qu'examinateurs.

Mes remerciement vont également à l'ensemble des membres du laboratoire LSIS, et plus particulièrement les membres de l'équipe DIMAG. Merci aussi à mes amies et amis : Amina, Feda, Meriem, Fadoua, Agus, Anne Marie...

J'exprime toute ma reconnaissance à mes chers parents, **Malika** et **Miloudi** qui ont été les premiers à croire en moi. Merci à vous pour le soutien et les encouragements tout au long de ces années malgré la distance. Merci pour l'amour que vous m'avez apportée. Si j'y suis arrivée, c'est aussi grâce à vous. Je remercie ma mère qui a supporté le déplacement entre le Maroc et la France pour garder mon fils depuis sa naissance. Rien au monde ne vaut les efforts fournis par vous deux pour moi.

Mon cher papa, tu es parti avant d'assister à ma soutenance et voir l'aboutissement de mon travail. Ce doctorat était très important à tes yeux mais rien ne nous prépare à la mort. Je savais que cela arriverait un jour mais pas maintenant en tout cas (à l'âge de 56 ans). Rien ne peut exprimer ce que tu mérites pour tous les sacrifices que tu n'as cessé de nous donner moi, mes soeurs et frères depuis la naissance jusqu'à ton départ. Tu as fait plus qu'un père puisse faire pour que ses enfants suivent le bon chemin dans leur vie et leurs études. Tu étais vraiment un papa exceptionnel !. Nous sommes tous fiers de toi. Tu étais notre ange gardien et notre fidèle compagnant dans les moments les plus délicats. Tu étais d'une force incroyable et d'un courage exemplaire. Tu nous as appris de ne jamais dévier de notre objectif final. Tu me manques beaucoup. Je ne t'oublierai jamais et tu resteras dans mon coeur pour toujours.

Je remercie également mes soeurs et fréres, mes proches ainsi que ma belle famille pour leur soutien. Un merci particulier à ma belle mère **Rkia** pour son soutien.

Je présente aussi mes sincères remerciements à mon cher époux **Mustapha** ; qui était toujours à mes cotés durant toutes ces années. Je le remercie pour son soutien dans les meilleurs comme dans les pires moments, ses multiples conseils, son aide si précieuse et ses encouragements. Cette thèse, nous l'avons en quelque sorte réalisée à deux. Je lui suis pour toujours reconnaissante. Aucun mot ne pourrait suffire pour exprimer tout ce qu'il m'a apporté. Que dieu réunisse nos chemins pour un long commun serein...

Mes remerciement vont aussi à la chair de ma chair, mon fils **Imran**. J'espère que tu seras content quand tu trouveras ton nom le jour où tu apprendras à lire. Tu'es ma joie de vivre !

Résumé

Cette thèse se place dans le cadre de services web en dépassant leur description pour considérer leur structuration en réseaux (i.e. réseaux d'interaction et réseaux de similitude). Nous proposons des méthodes basées sur les motifs, la modélisation probabiliste et l'analyse des concepts formels, pour améliorer la qualité des services découverts. Trois contributions sont alors proposées : (1) découverte de services diversifiés, (2) recommandation de services et (3) cohérence des communautés de services détectées. Nous proposons dans un premier temps une modélisation de l'espace des services web sous forme de réseaux. Afin de découvrir les divers services correspondants à une requête donnée, nous proposons une méthode probabiliste permettant de diversifier les résultats de la découverte. Cette méthode se base à la fois sur la pertinence (basée sur la similarité thématique), la diversité et la densité des services. Dans le cas de requêtes complexes, il est nécessaire de combiner plusieurs services pour satisfaire ce genre de requêtes. Dans ce contexte, nous exploitons le réseau d'interaction de services web construit et la notion de diversité dans les graphes pour identifier les services web qui sont susceptibles d'être composables. Nous proposons également un système de recommandation hybride basé sur le contenu et le filtrage collaboratif. L'originalité de la méthode proposée vient de la combinaison des modèles thématiques probabilistes et les motifs fréquents pour capturer la sémantique commune maximale d'un ensemble de services. Enfin, au lieu de ne traiter que des services individuels, nous considérons aussi un ensemble de services regroupés sous forme de communautés de services pour la recommandation. Nous proposons dans ce contexte, une méthode qui combine la sémantique et la topologie dans les réseaux afin d'évaluer la qualité et la cohérence sémantique des communautés détectées, et classer également les algorithmes de détection de communautés.

Mots-clés

Service web, Réseau de services, Modèles thématiques, Motifs fréquents, Communautés, Analyse de concepts formels, Recommandation.

Abstract

As a part of this thesis, we exceed the description of web services to consider their structure as networks (i.e. similarity and interaction web service networks). We propose methods based on patterns, topic models and formal concept analysis, to improve the quality of discovered services. Three contributions are then proposed : (1) diversified services discovery, (2) services recommendation and (3) consistency of detected communities. Firstly, we propose modeling the space of web services through networks. To discover the diversified services corresponding to a given query, we propose a probabilistic method to diversify the discovery results based on relevancy, diversity and service density. In case of complex requests, it is necessary to combine multiple web services to fulfill this kind of requests. In this regard, we use the interaction web service network and the diversity notion in graphs to identify all possible services compositions. We also propose a new hybrid recommendation system based on both content and collaborative filtering. Its originality comes from the combination of probabilistic topic models and pattern mining to capture the maximal common semantic of a set of services. Finally, instead of processing individual services, we consider a set of services grouped into service communities for the recommendation. We propose in this context, a new method combining both topology and semantics to evaluate the quality and the semantic consistency of detected communities, and also rank the detection communities algorithms.

Keywords

Web service, Service network, Probabilistic topic modeling, Patterns mining, Community, Formal concept analysis, Recommendation.

Table des matières

Table des figures

Liste des tableaux

Listings

Liste des algorithmes

Liste des publications

Ci-dessous la liste des publications realisées dans le cadre de cette thèse.

- **Hafida NAIM**, Mustapha AZNAG, Mohamed QUAFAFOU & Nicolas DURAND : **"Probabilistic Approach for Diversifying Web Services Discovery and Composition"**. In *the 23rd IEEE International Conference on Web Services (ICWS 2016)*, pp. 73-80, San Francisco, CA, USA, June 2016. Acceptance rate : 13%. Best ICWS'2016 Paper Award.

- **Hafida NAIM**, Mustapha AZNAG, Mohamed QUAFAFOU & Nicolas DURAND : **"Semantic Divergence based Evaluation of Web Service Communities"**. In *the 13th IEEE International Conference on Services Computing (SCC 2016)*, pp. 736-743, San Francisco, CA, USA, June 2016. Acceptance rate : 27%

- **Hafida NAIM**, Mustapha AZNAG, Nicolas DURAND & Mohamed QUAFAFOU : **"Semantic Pattern Mining Based Web Service Recommendation"**. In *the 14th International Conference on Service Oriented Computing (ICSOC 2016)*, pp. 417-432, Banff, Alberta, Canada, October 2016. Acceptance rate : 21%

1

Introduction générale

Sommaire

1.1 Contexte et problématique

Les services web sont définis comme des composants logiciels mis à disposition sur Internet et destinés à supporter l'interaction interopérable de machine à machine sur un environnement distribué. L'architecture des services web se fonde sur le modèle SOA (Service Oriented Architecture en anglais) qui permet l'organisation d'un ensemble de logiciels isolés en un ensemble de services interconnectés, accessibles par une interface et des protocoles standard. Un service représente, dans le modèle SOA, une unité discrète qui remplit une collection de tâches répondant aux mêmes objectifs [Jones, 2005]. Les services sont développés par des fournisseurs et publiés dans des annuaires de services. Ils sont accessibles sur le Web pour les clients qui les découvrent, les sélectionnent et les invoquent à l'aide de messages. L'architecture orientée service permet de remédier à des insuffisances des architectures distribuées. L'un des avantages des architectures SOA réside dans le fait que l'utilisation des services web vise essentiellement à assurer un couplage faible entre un ensemble d'applications hétérogènes.

A l'heure actuelle, différents processus ont été largement étudiés dans le but d'assurer une bonne gestion des services web. Dans cette thèse, nous abordons les processus de découverte et sélection, composition et recommandation des services web ainsi que la détection des communautés de services qui représentent des axes de recherche émergeants.

Découverte et sélection : le processus de découverte de services est défini par Keller et al. comme la *"localisation automatique des services répondant à une requête utilisateur"* [Keller et al., 2005]. Le travail présenté dans [Toma et al., 2005] définit également la découverte comme étant *"le processus qui prend en entrée une requête*

utilisateur et retourne une liste de ressources ou services pouvant combler éventuellement le besoin décrit". Comme illustré dans la figure 1.1, l'activité de recherche de services web implique souvent deux processus importants, à savoir la découverte et le classement de services :

- Découverte de services : le processus de découverte de services se base sur l'opérateur d'appariement qui permet de trouver les services répondant aux besoins des utilisateurs (fonctionnels et non-fonctionnels).
- Classement et sélection : le processus de classement et sélection de services permet de classer et sélectionner les meilleurs services web jugés pertinents, découverts lors du processus d'appariement, tout en prenant en considération un ensemble de critères de classement et/ou des préférences utilisateurs.

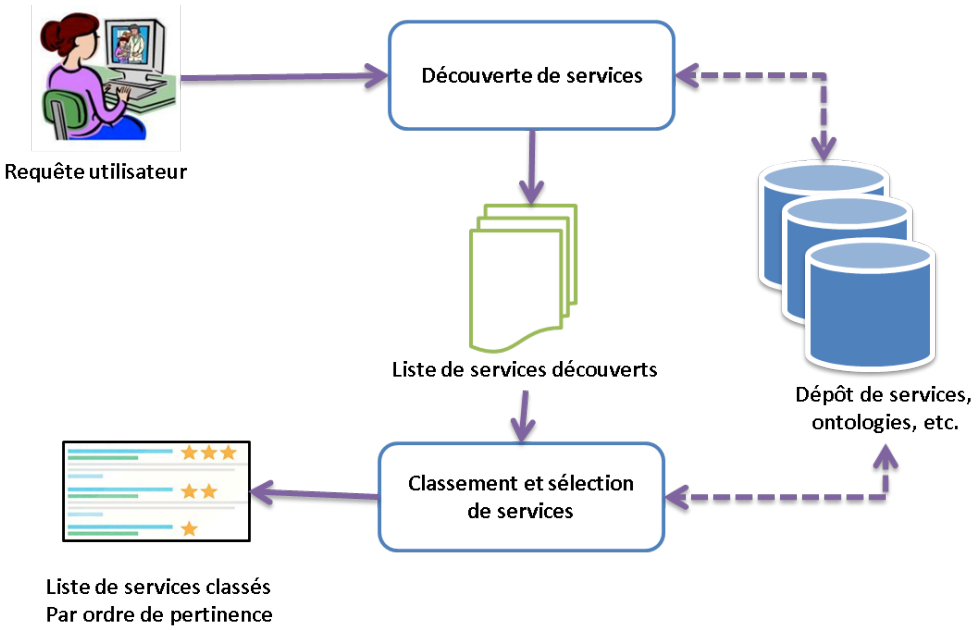


Figure 1.1 – Activités de découverte de services web

Composition de services web : le processus de composition des services web est considéré comme un moyen efficace pour créer, exécuter, et maintenir des services qui dépendent d'autres services [Benatallah et al., 2005]. Il permet donc de satisfaire les requêtes complexes des utilisateurs en combinant plusieurs services web lorsque cette requête ne peut pas être satisfaite par un des services existants.

Recommandation de services web : un système de recommandation a pour but d'aider l'utilisateur à choisir le service le plus adéquat à ses besoins parmi une large liste de service [Werthner et al., 2007]. Les systèmes de recommandations assistent donc l'activité de recherche de l'utilisateur en lui proposant des services qu'ils jugent pertinents par rapport à ses attentes.

Détection de communautés de services web : une communauté de services web peut regrouper un ensemble de services web ayant un même domaine d'intérêt ou offrant des fonctionnalités communes [Maamar et al., 2009]. Dans le cas des réseaux, une communauté peut être définie comme un sous-graphe composé de sommets fortement reliés entre eux et faiblement liés aux autres sommets du graphe [Fortunato, 2010].

Avec l'évolution rapide de la technologie des services web et la grande quantité de services web disponibles sur Internet, les mécanismes de découverte, de composition et de recommandation de services web constituent toujours un défi majeur de notre domaine de recherche. Plusieurs travaux de recherche ont été proposés dans la littérature pour améliorer le processus global de ces principales tâches. L'objectif principal des recherches focalisées sur les services web est la découverte des services pertinents permettant de satisfaire les besoins des utilisateurs. La majorité des systèmes existants se basent sur la recherche par mots-clés qui sont insuffisants pour capturer la sémantique des requêtes des utilisateurs. Certains de ces systèmes découvrent des services web qui sont souvent très similaires, redondants et parfois échouent à apporter une réponse dans le cas des requêtes complexes et/ou ambiguës (qui nécessitent la combinaison de multiple services). Pour faire face à ces problèmes, deux questions importantes doivent être envisagées :

1. Comment l'utilisateur peut choisir le ou les services adéquats à ses besoins ;
2. Comment identifier de possibles compositions de services permettant de satisfaire certaines requêtes.

De plus, l'adoption croissante des services web rend la structure et l'évolution de l'espace des services web de plus en plus complexes. Dans cette optique, différentes méthodes probabilistes ont été proposées afin de réduire la dimensionnalité de l'espace des services et faciliter la découverte des services web. Les travaux reportés dans [Aznag, 2015], utilisent des modèles thématiques probabilistes pour traiter de grands volumes de données en capturant les thématiques et leurs corrélations à partir de descriptions de services web. La notion de thème est utilisée, dans ce contexte, pour découvrir des groupes de données textuelles sur des sujets similaires. Ces groupes sont obtenus en calculant les occurrences des mots dans différents textes indépendants. Dans cette thèse, nous utilisons plus précisément, le modèle thématique probabiliste à thèmes corrélés CTM (Correlated Topic Models) [Blei & Lafferty, 2007] pour extraire un ensemble de thèmes à partir de description de services web. La modélisation thématique est utilisée dans notre contexte comme technique efficace de réduction des dimensions, en capturant des relations sémantiques entre mots-thèmes et thèmes-services, exprimées sous forme de distributions de probabilités.

La plupart des systèmes de découverte existants ignorent complétement la notion de la diversité dans la liste des résultats retournée. Afin d'éviter la redondance tout en maintenant la qualité des services web découverts, la diversité doit être prise en

considération dans les systèmes de découverte et/ou de recommandation. Dans cette thèse, nous proposons une nouvelle méthode permettant la diversification des résultats de la découverte de services web tout en maintenant la qualité des services web découverts. Nos travaux sur la découverte et la sélection de services web étendent les travaux proposés récemment dans [Aznag et al., 2013b, Aznag et al., 2014]. Plus précisément, nous proposons une méthode de diversification des résultats de la découverte en se basant à la fois sur la notion de pertinence (similarité thématique), la diversité et la densité des services [Naim et al., 2016a]. Récemment, le problème de la diversification des résultats de la recherche sur le Web a attiré beaucoup d'attentions et est devenu plus important dans le domaine de la Recherche d'Information [Bache et al., 2013]. En effet, la diversification des résultats de recherche est utilisée comme un moyen efficace pour satisfaire diverses préférences et besoins des utilisateurs sur le Web.

Dans le cas de requêtes complexes, il est nécessaire de renvoyer à l'utilisateur une séquence de services qui peuvent être composables afin de satisfaire ce genre de requêtes. Plusieurs approches traitant la composition de services ont été proposées dans la littérature [Wu & Khoury, 2012, Cheng et al., 2015]. Dans le cadre de cette thèse, nous ne proposons pas une méthode pour la composition de services web mais nous identifions un ensemble de compositions possibles qui peuvent exister entre les services web découverts et/ou avec d'autres services. Dans notre approche, nous traitons le problème de la composition de services web comme un problème de recherche dans un réseau d'interaction de services web [Naim et al., 2016a].

Récemment, plusieurs systèmes de recommandation ont été proposés dont l'objectif est d'aider l'utilisateur à effectuer le choix du meilleur service parmi un ensemble de services disponibles ayant une même fonctionnalité. Ainsi, l'utilisateur peut réduire son temps de recherche et recevoir également des propositions et suggestions auxquelles il n'aurait pas spontanément prêtées attention. C'est dans ce contexte que nous proposons un nouveau système de recommandation hybride basé sur le contenu et le filtrage collaboratif [Naïm et al., 2016]. Au lieu de ne traiter que des services individuels, nous considérons un ensemble de services regroupés sous forme de communautés de services pour faciliter la recommandation. Nous proposons dans ce contexte, une méthode qui combine la sémantique et la topologie dans les réseaux afin d'évaluer la qualité et la cohérence sémantique des communautés détectées, et classer également les algorithmes de détection de communautés [Naim et al., 2016b].

1.2 Contributions de la thèse

Les contributions de cette thèse se résument comme suit :

1. **Construction de réseaux de services :** nous proposons dans un premier temps, une méthode pour structurer des services web sous forme de réseaux (i.e. réseaux

d'interaction et réseaux de similitude). Nous considérons trois niveaux de granularité (paramètres, opérations et services) dans le cas du réseau d'interaction. Nous construisons dans un premier temps le réseau d'interaction d'opérations en se basant sur le formalisme de l'analyse de concepts formels (ACF) [Birkhoff, 1967] pour déduire ensuite le réseau d'interaction de services. Dans notre approche, nous organisons un ensemble d'opérations de services web et leurs paramètres d'entrée/sortie sous forme de treillis de concepts formels. Les réseaux d'interaction d'opérations se distinguent en fonction du mode d'invocation (totale ou partielle) [Naim et al., 2016a]. Nous proposons également deux modèles de réseaux de similitude de services web. Ces deux modèles se basent respectivement sur les descriptions syntaxiques et sémantiques des services web. Afin d'identifier la similarité entre deux services, nous proposons d'utiliser une mesure de proximité qui utilise le cosinus de l'angle entre deux vecteurs décrivant ces services. La similarité entre les services est calculée en se basant sur les représentations vectorielles (basées sur les vecteurs de compte TF-IDF) dans le cas de réseaux de similitude syntaxiques. Dans le cas des réseaux de similitude sémantiques, la distribution de probabilités sur les thèmes est utilisée comme critère de base pour calculer la similarité sémantique entre les services.

2. **Diversification des résultats de la découverte des services web et découverte de leurs possibles compositions :** nous proposons ensuite un système de découverte et de classement de services web permettant de diversifier les résultats de la découverte tout en maintenant la qualité des services découverts [Naim et al., 2016a]. Nous représentons, dans un premiers temps, la requête de l'utilisateur dans l'espace des thèmes afin de découvrir l'ensemble des premiers thèmes implicites qui sont sémantiquement associés à la requête. En se basant sur les thèmes sélectionnés, nous calculons la similarité entre la requête et les services web liés à ces thèmes. Ensuite, nous utilisons les scores de la similarité pour classer et sélectionner les k premiers services web. Afin de minimiser la redondance dans la liste des services sélectionnés, nous proposons de re-classer pour une seconde fois les services sélectionnés précédemment en tenant compte de la pertinence ou similarité sémantique, la diversité et la densité des services. Par conséquent, notre système de découverte retourne un ensemble de services pertinents divers classés par ordre de pertinence décroissant. Afin d'enrichir la liste de services renvoyée à l'utilisateur et maximiser la chance de répondre à certaines requêtes ambiguës, qui nécessitent généralement la combinaison de multiple services, notre système découvre également les services web qui sont susceptibles d'être composables. Dans notre approche, nous nous basons sur le modèle de réseau d'interaction de services pour trouver les liens de compositions possibles entre les services web découverts et/ou avec d'autres services disponibles dans le référentiel de services. Dans un réseau d'interaction

de services, un lien entre deux services représente la possibilité de les composer. Notre méthode permet également l'optimisation et la diversification des liens de compositions identifiés afin de réduire la liste des liens identifés et sélectionner des compositions pertinentes. Nous précisons que nous ne produisons pas un service composite offrant une nouvelle fonctionnalité mais nous découvrons tout simplement un ensemble de compositions possibles de services web. Notre système prend comme entrée la liste des services pertinents divers et effectue une recherche dans le réseau d'interaction de services afin d'extraire un sous-graphe contenant l'ensemble des compositions possibles. Enfin, le système optimise le sous-graphe généré en exploitant les notion de pertinence, de diversité, de densité des services et la notion de diversité dans les graphes.

3. **Recommandation de services web basée sur l'extraction des motifs :** nous proposons également dans le cadre de cette thèse un système de recommandation hybride qui combine les techniques de recommandation basées sur le contenu et le filtrage collaboratif [Naïm et al., 2016]. Les approches de filtrage collaboratif permettent de sélectionner les services pertinents pour l'utilisateur courant en collectant des informations auprès d'autres utilisateurs similaires. Les approches basées sur le contenu recommandent les services web en se basant sur la similarité entre les requêtes de l'utilisateur et la description des services web (i.e. fonctionnalités du service). L'originalité de notre approche vient de la combinaison des modèles thématiques probabilistes et les motifs fréquents pour capturer la sémantique commune maximale d'un ensemble de services. Dans notre approche, l'utilisation des thèmes produits par le modèle thématique probabiliste permet essentiellement de (1) réduire la dimensionnalité de l'espace du contexte d'extraction de motifs à partir d'une grande collection de services web, et (2) capturer des relations sémantiques entre thèmes-mots et services-thèmes. Par conséquent, capturer la sémantique commune maximale d'un ensemble de services. L'objectif principal de notre méthode est d'identifier un ensemble de services web qui sont très sémantiquement liés par rapport à un service donné et ayant de meilleures qualités de services. Pour cela, nous avons introduit la notion de motifs sémantiques. Ces motifs sémantiques correspondent à des motifs fréquents maximaux des thèmes qui sont utilisés ensuite pour extraire un ensemble de motifs de services. Les services d'un motif sémantique sont sémantiquement liés et maximaux. Pour calculer les motifs sémantiques et les ensembles correspondants de services, nous construisons un treillis de concepts fréquents [Zaki & Hsiao, 2005]. Les ensembles de services résultats sont ensuite stockés dans une structure spéciale, appelée MFI-tree [Grahne & Zhu, 2005], afin d'effectuer des recherches rapides par le système de recommandation en se basant sur des index. A partir d'un service spécifié, le système de recommandation utilise cet arbre pour trouver les services qui lui sont sémantiquement similaires. Les

services obtenus sont ensuite classés et recommandés à l'utilisateur. Afin de classer les services web concernés, nous avons introduit une nouvelle mesure de classement permettant de mesurer le degré de pertinence de ces services par rapport au service demandé. Cette mesure combine la similarité sémantique (basée sur la distribution de probabilité sur les thèmes) et un ensemble de propriétés non-fonctionnelles (la qualité de services, la réputation, le temps de réponse, la disponibilité, ...). La QoS joue un rôle important dans les systèmes de recommandation, dans lesquels un ensemble de services similaires peuvent être classés et sélectionnés pour les utilisateurs.

4. **Évaluation de la cohérence sémantique des communautés de services web et classement des algorithmes de détection de communautés :** Plusieurs méthodes et algorithmes de détection de communautés ont été proposés dans la littérature. La majorité des algorithmes de détection des communautés, dans un réseau complexe, ignorent complètement la sémantique des noeuds et considèrent uniquement la nature topologique des communautés. Dans cette thèse, nous proposons une nouvelle méthode permettant : (1) d'évaluer la qualité et la cohérence sémantique des communautés détectées et (2) d'évaluer et classer les algorithmes de détection de communautés [Naim et al., 2016b]. Afin d'évaluer la qualité des communautés détectées, nous utilisons des mesures classiques, à savoir la pureté et l'entropie [Zhao & Karypis, 2001]. Ces mesures sont largement utilisées pour évaluer les performances des techniques d'apprentissage supervisé et non supervisé, en particulier, le clustering et les algorithmes de détection de communautés. Ensuite, pour évaluer la cohérence sémantique des communautés, nous introduisons une nouvelle mesure appelée la divergence sémantique des communautés détectées basée sur la divergence de *Kullback Leibler (KL)* [Steyvers & Griffiths, 2007]. Dans notre approche, nous utilisons les thèmes et les distributions de probabilités produites par le modèle probabiliste à thèmes corrélés CTM pour calculer la divergence sémantique de chaque communauté. Enfin, pour évaluer et classer les algorithmes de détection de communautés, nous introduisons une nouvelle mesure en combinant le score de la cohérence sémantique (basé sur la divergence sémantique) avec celui basé sur les mesures classiques (pureté et entropie).

Différentes expérimentations et évaluations ont été réalisées pour valider nos propositions. Ces expérimentations ont été réalisées en se basant sur des services web réels collectés à partir d'Internet et des collections standard de services web.

1.3 Organisation de la thèse

Le présent rapport est organisé comme suit.

Chapitre 2 : ce chapitre est consacré à une brève présentation des notions de base

permettant de comprendre les travaux proposés dans cette thèse. Nous introduisons dans un premier temps les concepts de base sur les services web et les différentes techniques utilisées pour l'extraction de toutes les informations décrivant les services web à partir des documents WSDL. Ensuite, nous présentons le modèle thématique probabiliste (CTM) utilisé pour extraire les thèmes à partir des descriptions de services web. Enfin, nous introduisons la théorie et des concepts relatifs à l'analyse de concepts formels (ACF) et l'extraction de motifs fréquents.

Chapitre 3 : nous décrivons dans ce chapitre, un état de l'art pour délimiter le périmètre de nos travaux dans le cadre des services web. Nous commençons par présenter différents modèles de description syntaxique et sémantique de services web. Nous présentons ensuite un ensemble de travaux traitant la découverte de services web. Puis, nous dressons quelques approches proposées dans la littérature dédiées aux systèmes de recommandation. Nous étudions également quelques travaux basés sur les réseaux pour modéliser les services web et leurs interactions. Dans ce contexte, nous discutons un ensemble de travaux proposés dans le cadre de la composition et la détection de communautés de services web. Enfin, nous présentons quelques algorithmes d'extraction de motifs fréquents.

Chapitre 4 : dans ce chapitre, nous introduisons deux modèles de réseaux de services web ; un modèle d'interaction et un modèle de similitude. Nous présentons dans un premier temps quelques concepts et propriétés fondamentales relative à la topologie des réseaux. Ensuite, nous décrivons les modèles de réseaux de services proposés et comment nous les construisons.

Chapitre 5 : ce chapitre a pour objectif de présenter la méthode proposée pour diversifier les résultats de la découverte de services et la découverte de leur possibles compositions. Nous présentons d'abord la motivation de ce travail en l'illustrant par un exemple et en formulant le problème traité. Ensuite, nous présentons un aperçu global du système proposé dans ce chapitre. Puis, nous décrivons plus en détail notre méthode de diversification des résultats de la découverte de services web. Enfin, nous décrivons la méthode d'identification et découverte de compositions de services web avant de présenter les différentes expérimentations et évaluations réalisées dans ce chapitre.

Chapitre 6 : nous proposons dans ce chapitre un système de recommandation hybride basé sur le contenu et le filtrage collaboratif. La méthode proposée combine les modèles thématiques probabilistes et les motifs fréquents pour capturer la sémantique commune maximale d'un ensemble de services. Nous présentons dans un premier temps un aperçu général du système et les différentes étapes de notre approche qui repose sur les notions des thèmes et l'extraction de motifs sémantiques. Nous détaillons ensuite la méthode d'extraction des motifs sémantiques et la méthode de recommandation de services web proposée. Enfin, nous présentons en détail les

différentes expérimentations et évaluations réalisées pour évaluer notre méthode.

Chapitre 7 : ce chapitre vise à présenter une approche qui combine la sémantique et la topologie afin d'évaluer la qualité et la cohérence sémantique des communautés détectées. Tout d'abord, nous introduisons brièvement le contexte de notre travail tout en décrivant la problématique traitée dans ce chapitre. Ensuite, nous présentons la liste des algorithmes de détection des communautés que nous avons sélectionné pour évaluer notre méthode. Puis, nous décrivons plus en détail la méthode proposée pour l'évaluation et le classement de ces algorithmes. Enfin nous présentons les différentes expérimentations et évaluations réalisées dans ce chapitre.

Chapitre 8 : ce chapitre conclut cette thèse par une synthèse générale de nos contributions. Nous présentons également un ensemble de perspectives de recherche envisagées pour la poursuite de nos travaux.

2

Préliminaires et concepts de base

Sommaire

2.1 Introduction

Nous décrivons dans ce chapitre les concepts et notions de base nécessaires à la compréhension des approches proposées dans cette thèse. Nous introduisons dans un premier temps les généralités et concepts sur les services web (voir section 2.2). Ensuite, nous présentons dans la section 2.3 les différentes techniques utilisées pour extraire, à

partir des documents WSDL, toutes les informations décrivant les services web. Ces informations seront utilisées pour produire des représentations vectorielles de services. Nous décrivons également dans cette section le modèle thématique probabiliste utilisé pour extraire les thèmes à partir des descriptions de services. Enfin, nous introduisons l'analyse de concepts formels et l'extraction de motifs fréquents, respectivement dans les sections 2.4 et 2.5.

2.2 Généralités sur les services web

Un service web [1] est défini comme un composant logiciel mis à disposition sur Internet par un fournisseur de services et destiné à supporter l'interaction interopérable de machine à machine dans un environnement distribué. Les services web représentent un moyen efficace pour la mise en œuvre des architectures orientés services (SOA - Service Oriented Architecture). Dans cette section, nous présentons les définitions et l'architecture de base des services web. Nous introduisons également la description (i.e. les propriétés fonctionnelles et non-fonctionnelles), le mécanisme de découverte et l'invocation des services web.

2.2.1 Définitions et architecture de base des services web

Dans la littérature, il est difficile de trouver une définition exacte du mot "service" parce que de nombreuses définitions des services web ont été proposées. Nous en citons quelques-unes dans ce qui suit :

- *"Un service web est un composant logiciel granulaire qui peut être utilisé comme un module pour des applications réparties ou pour l'ensemble des processus métiers."*[2]
- *"Un service web est un systéme logiciel conçu pour supporter l'interaction interopérable de machine à machine sur un réseau. Il dispose d'une interface décrite dans un format exploitable automatiquement par la machine, i.e. décrite en WSDL (Web Services Description Language). Les services web peuvent être composés d'une manière faiblement couplée pour réaliser des tâches complexes. Les programmes offrant des services simples, peuvent interagir ensemble afin de mettre en place des services sophistiqués avec des valeurs ajoutées."*[3]
- *" Un service web est considéré comme une façon standardisée d'intégration des applications basées sur le Web en utilisant les standards ouverts XML, SOAP, WSDL, UDDI et les protocoles de transport de l'Internet. XML est utilisé pour représenter les données, SOAP pour transporter les données, WSDL pour décrire les*

1. http ://www.w3.org/standards/webofservices/
2. Ibm, http://www.ibm.com
3. W3C (World Wide Web Consortium), http://www.w3c.org/

services disponibles, et UDDI pour lister les fournisseurs de services et les services disponibles."[4]

A travers les différentes définitions présentées ci-dessus, plusieurs technologies sont mises en évidence pour mettre en œuvre un service web à savoir :

- SOAP (Simple Object Access Protocol) est un protocole de communication standard permettant l'échange de données et l'appel à distance des opérations offertes par un service web. SOAP est basé sur XML et plus simple pour être adapté aux besoins d'un contexte.
- WSDL (Web Service Definition Language) est un langage définissant le mécanisme de description d'un service web sous forme d'un fichier de description en XML.
- UDDI (Universal Description Discovery and Integration) est un protocole fournissant le registre (privé ou public) permettant de publier et découvrir un service web.

L'architecture des services web se base sur le modèle SOA qui fait intervenir trois acteurs : un client (consommateur de services), un fournisseur de services ainsi qu'un intermédiaire jouant le rôle d'annuaire de services.

1. Le fournisseur de services met à disposition son service web et le rend accessible sur le web.
2. Le service client (consommateur du service web) utilise un service web existant en ouvrant une connexion avec l'annuaire de services et en envoyant une demande sous forme d'une requête SOAP, XML-RPC[5], ou tout simplement une requête HTTP pour les services web REST[6].
3. L'annuaire offre des facilités de publication de services aux fournisseurs et facilite la recherche des services aux clients.

La figure 2.1 montre un aperçu global de l'architecture de base des services web basée sur les protocoles de base. Le modèle d'interaction entre ces trois acteurs suivent plusieurs étapes successives :

1. **Déploiement** : le fournisseur déploie son service web sur un serveur et génère une description du service (le document WSDL). Cette description précise les fonctions disponibles et comment les invoquer.
2. **Publication** : le fournisseur publie des services web avec leurs descriptions dans l'annuaire de services.
3. **Découverte** : le client cherche un service particulier publié dans l'annuaire qui va lui fournir les descriptions et les URL des services demandés. Cette étape de découverte

4. Webopedia, http://www.Webopedia.com
5. XML-RPC est un protocole RPC (Remote Procedure Call), une spécification simple et un ensemble de codes qui permettent à des processus s'exécutant dans des environnements différents de faire des appels de méthodes à travers un réseau.
6. Representational State Transfer. REST a été décrit par Roy Thomas Fielding dans sa thèse "Architectural Styles and the Design of Network-based Software Architectures"

permet de connaître les différents fournisseurs disponibles et de sélectionner celui qui répond le mieux aux besoins du client.

4. **Invocation** : le client utilise l'URL et la description du service fourni par l'annuaire pour l'invoquer auprès du fournisseur de services.

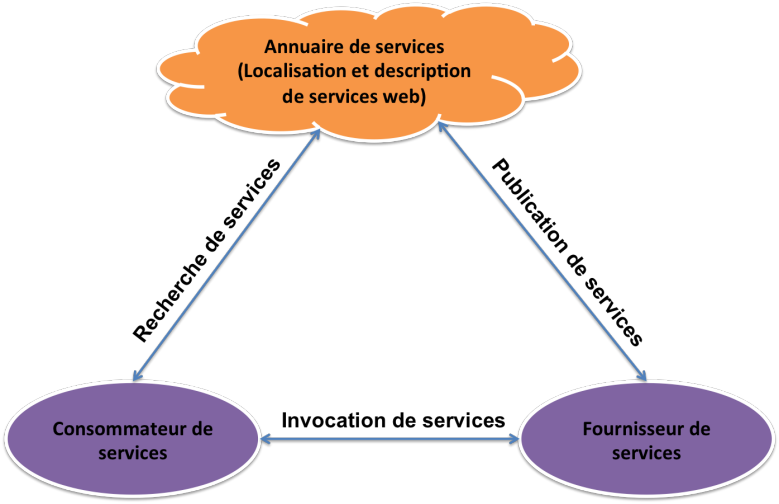


Figure 2.1 – Architecture de base des services web.

2.2.2 Description et propriétés fonctionnelles

La description d'un service web permet de définir son comportement et indique aux clients potentiels comment interagir avec ce service. Une fonctionnalité abstraite de ce service est représentée au monde extérieur via une interface abstraite décrite en utilisant le langage de description des services web WSDL (Web Service Description Language). Le standard WSDL se base sur une grammaire XML et permet de décrire et publier le format et les protocoles d'un service web de manière homogène. Depuis 2007, la version 2.0 du WSDL est considérée comme une recommandation officielle du W3C. La structure d'un document WSDL est composée d'un ensemble de *définitions*. Une *définition* constitue l'élément englobant contenant le nom et les éléments d'un service. En général, un document WSDL comprend sept éléments principaux divisés en deux parties. Une première partie de description abstraite et une deuxième partie de description concrète. La description abstraite décrit les fonctionnalités fournies par un service et peut être instanciée par plusieurs implémentations concrètes. Les *types*, les *messages* et les *interfaces* (opérations fournies par un service) constituent la partie abstraite d'un document WSDL, les *liaisons* (bindings) et les *services* en constituent la partie concrète. Par exemple, le listing 2.1 montre le format du document WSDL décrivant le service

ZipCity_Service[7]. Nous remarquons que l'élément <wsdl :definitions> englobe tous les éléments du document WSDL.

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
 <wsdl:documentation>Cocoma City Lookup Web Services.</wsdl:documentation>
 <wsdl:types>...</wsdl:types>
 <wsdl:message name="GetCitySoapIn">...</wsdl:message>
 <wsdl:portType name="ZipCity_ServiceSoap">...</wsdl:portType>
 <wsdl:binding name="ZipCity_ServiceSoap" type="tns:ZipCity_ServiceSoap">
 ...</wsdl:binding>
 <wsdl:service name="ZipCity_Service">...</wsdl:service>
</wsdl:definitions>
```

Listing 2.1 – Le document WSDL définissant le service ZipCity_Service

Service

L'élément <wsdl :service> agrège plusieurs éléments <Ports>, contenant chacun un nom, une URL de point d'accès et une référence à un binding. Dans le listing 2.2, le service "ZipCity_Service" possède deux interfaces "ZipCity_ServiceSoap" et "ZipCity_ServiceSoap12" pour lesquelles on spécifie deux adresses différentes.

```
<wsdl:service name="ZipCity_Service">
 <wsdl:documentation>Cocoma City Lookup Web Services.</wsdl:documentation>
<wsdl:port name="ZipCity_ServiceSoap" binding="tns:ZipCity_ServiceSoap">
   <soap:address location="http://service.ecocoma.com/geo/zipcity.asmx"/>
 </wsdl:port>
 <wsdl:port name="ZipCity_ServiceSoap12" binding="tns:ZipCity_ServiceSoap12
     ">
  <soap12:address location="http://service.ecocoma.com/geo/zipcity.asmx"/>
 </wsdl:port>
</wsdl:service>
```

Listing 2.2 – Définition de l'élément <wsdl :service> du service ZipCity_Service

Binding

L'élément <wsdl :binding> définit le lien entre les <portTypes> et spécifie les protocoles de transmission de données utilisés (tels que SOAP). L'élément <wsdl :operation> représente la description abstraite d'une action dans le port. Il peut y avoir plusieurs opérations dans un document WSDL. Notons que, par exemple, le service "ZipCity_Service" possède deux binding "ZipCity_ServiceSoap" et "ZipCity_ServiceSoap12" correspondant

7. Le document WSDL associé au service ZipCity_Service est extrait de la collection de test SAWSDL-TC3 décrite dans la section 4.5.1, page 85

aux interfaces qu'il fournit (voir le listing 2.3). Pour l'interface "ZipCity_ServiceSoap", le binding spécifie que le protocole utilisé est SOAP <soap :binding>.

```
<wsdl:binding name="ZipCity_ServiceSoap" type="tns:ZipCity_ServiceSoap">
 <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
   <wsdl:operation name="GetCity">
    <soap:operation soapAction="http://service.ecocoma.com/geo/zipcity/
        GetCity"
    style="document"/> <wsdl:input> <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
<wsdl:binding name="ZipCity_ServiceSoap12" type="tns:ZipCity_ServiceSoap">
    ...
   </wsdl:binding>
```

Listing 2.3 – Définition de l'élément <wsdl :binding> du service ZipCity_Service

PortType

L'élément <wsdl :portType> peut comprendre un ensemble d'opérations abstraites mais en général une seule opération est associée à un portType. Chacune des opérations fait référence à un message en entrée et/ou des messages de sortie. A titre d'exemple, le service "ZipCity_Service" offre un <portType> : "ZipCity_ServiceSoap" qui possède une opération appelée "GetCity". Cette opération prend en entrée un message appelé "GetCitySoapIn" et retourne en sortie le message "GetCitySoapOut" (voir le listing 2.4).

```
<wsdl:portType name="ZipCity_ServiceSoap">
   <wsdl:operation name="GetCity">
      <wsdl:documentation>Get accurate city and state information when you
          only have the ZIP Code. (United States)</wsdl:documentation>
      <wsdl:input message="tns:GetCitySoapIn"/>
      <wsdl:output message="tns:GetCitySoapOut"/>
   </wsdl:operation>
</wsdl:portType>
```

Listing 2.4 – Définition de l'élément <wsdl :portType> du service ZipCity_Service

Message

L'élément <wsdl :message> définit la structure de données et leur contenu qui vont être échangés lors de l'utilisation du service. Les paramètres d'entrée et de sortie sont spécifiés dans l'élément *<wsdl :message>*. Dans le listing 2.5, nous avons les messages

d'entrée et de sortie qui sont spécifiés pour l'opération "GetCity". Le message d'entrée est "GetCitySoapIn" ayant un seul paramètre de type "GetCity". Le message de sortie "GetCitySoapOut" a également un seul paramètre de type "GetCityResponse".

```
<wsdl:message name="GetCitySoapIn">
    <wsdl:part name="parameters" element="tns:GetCity"/>
</wsdl:message>
<wsdl:message name="GetCitySoapOut">
    <wsdl:part name="parameters" element="tns:GetCityResponse"/>
</wsdl:message>
```

Listing 2.5 – Définition de l'élément <wsdl :message> du service ZipCity_Service

Types

L'élément <wsdl :types> définit les types de données échangées dans les messages sous la forme d'un schéma XML. Si le type (ou format) de données est complexe, les développeurs peuvent définir leurs propres types de données complexes en utilisant des types primitifs définis par XML (integer, string, float, long, boolean, short). Le service "ZipCity_Service" définit un schéma composé de deux éléments de types "GetCity" et "GetCityResponse", comme mentionné dans le listing 2.6. Ils représentent les types de paramètres des messages "GetCitySoapIn" et "GetCitySoapOut" respectivement. L'élément "GetCity" se compose d'une séquence de trois éléments de type string, tandis que l'élément "GetCityResponse" contient seulement un seul élément complexe de type "ZipCity". Ce dernier se compose de quatre éléments de type string.

```
<wsdl:types>
      <s:schema elementFormDefault="qualified" targetNamespace="http://
          service.ecocoma.com/geo/zipcity">
          <s:element name="GetCity">
              <s:complexType>
                  <s:sequence>
                      <s:element minOccurs="0" maxOccurs="1" name="KeyID" type=
                          "s:string" sawsdl:modelReference="http://127.0.0.1/
                          ontology/geographydataset.owl#UniqueIdentifier"/>
                      <s:element minOccurs="0" maxOccurs="1" name="DomainID"
                          type="s:string" sawsdl:modelReference="http://127
                          .0.0.1/ontology/geographydataset.owl#UniqueIdentifier
                          "/>
                      <s:element minOccurs="0" maxOccurs="1" name="ZipCode"
                          type="s:string" sawsdl:modelReference="http://127
                          .0.0.1/ontology/geographydataset.owl#PostalCode"/>
                  </s:sequence>
              </s:complexType>
          </s:element>
          <s:element name="GetCityResponse">
```

```
            <s:complexType>
               <s:sequence>
                  <s:element minOccurs="0" maxOccurs="1" name="
                      GetCityResult" type="tns:ZipCity"/>
               </s:sequence>
            </s:complexType>
         </s:element>
         <s:complexType name="ZipCity">
            <s:sequence>
               <s:element minOccurs="0" maxOccurs="1" name="City" type="s:
                   string" sawsdl:modelReference="http://127.0.0.1/ontology
                   /protonu.owl#City"/>
               <s:element minOccurs="0" maxOccurs="1" name="State" type="s:
                   string" sawsdl:modelReference="http://127.0.0.1/ontology
                   /geographydataset.owl#StateCode"/>
               <s:element minOccurs="0" maxOccurs="1" name="Longitude" type
                   ="s:string" sawsdl:modelReference="http://127.0.0.1/
                   ontology/protont.owl#longitude"/>
               <s:element minOccurs="0" maxOccurs="1" name="Latitude" type=
                   "s:string" sawsdl:modelReference="http://127.0.0.1/
                   ontology/protont.owl#latitude"/>
            </s:sequence>
         </s:complexType>
      </s:schema>
   </wsdl:types>
```

Listing 2.6 – Les types de WSDL définis pour le service "ZipCity_Service"

2.2.3 Découverte et invocation de services web

Le processus de découverte d'un service web consiste à localiser ou rechercher un service parmi ceux qui ont été publiés par les fournisseurs de services web. Une fois les services web décrits via le langage WSDL, ils doivent être publiés dans un annuaire de services web. Ainsi, les clients d'un service web donné peuvent lancer une recherche auprès de l'annuaire pour retrouver sa description et les différents fournisseurs disponibles pour choisir le service qui correspond le mieux à leurs besoins. Les annuaires ou portails publics proposent une interface entre les fournisseurs et les clients de services web permettant la publication et la découverte des services web. Les services sont collectés à partir d'Internet ou en se basant sur la publication manuelle. En général, il existe deux types d'annuaires de services qui sont les plus utilisés pour gérer et manipuler les services web à savoir : les *annuaires* et les *moteurs de recherche* de services web.

Les *annuaires* peuvent être définis comme des structures organisées de gestion de services. Toutes les informations nécessaires pour l'identification et la découverte d'un service peuvent être stockées dans ce type de structures. Ainsi, les fournisseurs de services peuvent publier leurs services et les consommateurs peuvent trouver les services satisfaisant

leurs demandes. De manière générale, les annuaires sont basés sur des standards comme la norme *UDDI* permettant la publication et la découverte de services web.

UDDI (Universal Description, Discovery and Integration) [8] [OASIS UDDI Spec, 2004] est un standard normalisé par OASIS (Organization for the Advancement of Structured Information Standards) [9]. Il décrit la structure d'un annuaire de services web permettant la publication et la recherche des services web. Les services publiés dans l'annuaire UDDI sont accessibles par l'intermédiaire du protocole de communication SOAP. Les fournisseurs doivent spécifier en XML toutes les informations les concernant, concernant les services qu'ils offrent ainsi que des détails techniques sur chaque service afin que la recherche soit faite de manière dynamique et automatique. Un annuaire ou registre UDDI peut être public ou privé et lui aussi est exposé sous forme de service Web.

Le registre UDDI offre trois composants principaux sous forme de documents XML permettant la publication d'un service web :

- Le fournisseur (Business Entity) : ce document XML contient des informations concernant le fournisseur d'un service web et l'entreprise qui l'héberge.
- Le service (Business Service) : cette entité comprend la description des services web et toutes les informations concernant le nom d'un service, et son objectif.
- les accès au service (Binding Template) : ce composant spécifie la modalité de liaison avec le service en offrant les points d'accès aux services web (URL) et les protocoles à utiliser pour les invoquer.

Pour rechercher et sélectionner un service web publié, le registre UDDI structure les informations en trois parties spécifiées en XML. Ces parties sont décrites comme suit :

- Pages Blanches : cette partie est utilisée pour trouver un service par le contact, nom et adresse du fournisseur de services.
- Pages Jaunes : sont utilisées pour trouver un service par catégorisations industrielles fondées sur des normes de taxonomies.
- Pages Vertes : ces pages servent à chercher un service par les caractéristiques techniques publiées par les entreprises.

Actuellement, la découverte de services web n'est plus limitée sur les registres UDDI. Le processus de la découverte peut donc se réaliser en utilisant d'autres registres ou solutions qui ont été créées et sont disponibles sur le Web telles que les moteurs de recherche ou les portails spécifiques aux services web. Dans la catégorie des annuaires, nous pouvons citer par exemple les annuaires suivants : Xmethods [10], RemoteMethods [11] et StrikeIron [12]. Les moteurs de recherche ou portails de services sont définis comme des annuaires web publics

8. UDDI:http://xml.coverpages.org/uddi.html
9. OASIS:http://www.oasis-open.org/
10. http://www.xmethods.com
11. http://www.remotemethods.com
12. http://www.strikeiron.com/

qui offrent de nombreuses fonctionnalités avancées via des interfaces web pour faciliter la gestion des services web publiés. Ils permettent la publication manuelle de services par les fournisseurs de services ou par les utilisateurs du portail. La plupart des moteurs de recherche ou portails de services offrent une fonctionnalité de navigation dans les résultats retournés. Nous pouvons donner ainsi l'exemple de Seekda [13], Service-Finder [14], BioCatalogue [15] et WS-Portal [16].

L'invocation d'un service web par un client s'effectue via le protocole de communication standard SOAP. Le client connaît au préalable le service web qu'il veut invoquer auprès de son fournisseur en utilisant l'URL et la description de ce service qui est fourni par l'annuaire.

SOAP (Simple Object Access Protocol) est un protocole basé sur XML assurant l'échange d'informations dans un environnement distribué et décentralisé de façon indépendante de toute plate-forme et de tout langage de programmation. Il fait partie des technologies les plus importantes des services web. Il utilise principalement deux standards HTTP et XML. XML est utilisé pour structurer les données (requêtes et réponses), tandis que HTTP constitue un moyen de transport des messages SOAP. La structure d'un message SOAP se compose de trois éléments essentiels qui sont : l'*enveloppe*, l'*en-tête du message* (Header) et le *corps du message* (Body). L'enveloppe définit le cadre général pour exprimer le contenu d'un message. Sa présence est obligatoire dans un message SOAP. L'en-tête du message est un élément optionnel, appelé aussi « règles d'encodage ». Il constitue un mécanisme qui permet de passer les informations qui ne sont pas prises en considération par les applications. Le corps du message permet la transmission des informations contenues dans les requêtes et les réponses échangées. La présence de cet élément dans un message SOAP est obligatoire.

2.2.4 Qualité de services et propriétés non-fonctionnelles

L'évolution rapide des services web, rend la tâche de sélection et découverte des services adéquats très difficile. Ainsi, les consommateurs finaux de ces services peuvent se trouver avec des services très similaires ayant les mêmes fonctionnalités. Il est donc nécessaire que ces services soient décrits le plus précisément possible. Comme les technologies actuelles basées sur SOAP, WSDL et UDDI ne permettent qu'une description syntaxique de l'interface des services web, plusieurs éléments non-fonctionnels capturant des aspects de la qualité de service (QoS - Quality Of Service) sont utilisés afin de satisfaire au mieux les demandes des consommateurs. Dans la littérature, plusieurs définitions et travaux ont été proposés sur la qualité de service QoS [Blum, 2004, Xu et al., 2007, Lee et al., 2006,

13. http://www.seekda.com/
14. http://www.service-finder.eu
15. https://www.biocatalogue.org/
16. http://wvm.esil.univ-mrs.fr/wsportal

Zheng et al., 2011b]. La QoS peut être définit comme un ensemble d'attributs non-fonctionnels qui influencent la qualité du service offert par un service web [Lee et al., 2006]. Certains attributs de la QoS peuvent être calculés par le client d'un service web (comme par exemple, le temps de réponse), tandis que d'autres attributs doivent être calculés et indiqués par le fournisseur du service web. Afin de modéliser les attributs de la QoS, plusieurs propriétés non-fonctionnelles peuvent être utilisées. Nous pouvons citer :

- **La disponibilité** est la probabilité que le service soit disponible pour répondre aux demandes des consommateurs.
- **La capacité** est la limite de requêtes simultanées qu'un service peut traiter. Lorsque le nombre de demandes simultanées dépasse la capacité d'un service, sa disponibilité et sa fiabilité diminuent.
- **La fiabilité** est la capacité d'un service de s'acquitter de ses fonctions requises dans des conditions déterminées pour une période de temps spécifique.
- **La performance** est la mesure de la vitesse pour effectuer une demande de service. Elle est mesurée par la latence (le délai entre l'arrivée et l'achèvement d'une demande de service), le débit (le nombre de demandes traitées sur une période de temps) et temps de réponse (le délai entre la demande et l'obtention d'une réponse auprès du service).
- **La sûreté** de fonctionnement de services web représente une propriété qui intègre plusieurs attributs dont les plus importants sont : la fiabilité, la disponibilité et la sécurité.
- **Le prix** des services.
- etc.

2.3 Représentation de services web et extraction de thèmes

Dans cette section, nous présentons tout d'abord les différentes techniques nécessaires pour extraire, à partir des documents WSDL, toutes les informations décrivant un service web. Ces informations seront utilisées dans nos approches pour produire des représentations vectorielles des services web. Ensuite, nous décrivons la tâche d'extraction des thèmes à partir des descriptions de services tout en introduisant brièvement les modèles thématiques probabilistes. Enfin, nous décrivons le modèle probabiliste que nous avons utilisé dans cette thèse pour extraire les thèmes à partir des descriptions de services.

2.3.1 Extraction d'information et représentation de services

Comme mentionné précédemment dans la section 2.2.2, les services web sont en général décrits par le langage de description standard WSDL. Le document WSDL permet de

décrire, par le biais de plusieurs balises XML (<*service*>, <*binding*>, <*operation*>, <*message*>, etc.), la localisation du service associé, le protocole de communication et de transport utilisé et le format des messages nécessaires pour invoquer les différentes opérations qu'offre ce service. La première tâche d'extraction d'information consiste donc à extraire toutes les descriptions textuelles et tous les éléments (i.e. messages, opérations, interfaces, types, etc.) qui décrivent un service web à partir de son document WSDL. Nous notons que les types WSDL (simples et complexes) sont utilisés par les messages pour transmettre des informations permettant d'invoquer les services. Par conséquent, ces types représentent des informations pertinentes pour décrire les fonctionnalités principales d'un service web. Nous utilisons ensuite toutes ces informations extraites pour construire des représentations de services web.

Nous présentons dans cette section, quelques traitements textuels classiques et différentes techniques que nous utilisons pour traiter le contenu des documents WSDL. Nous décrivons ensuite comment nous construisons la représentation des services web. Nous détaillons tout d'abord les différents traitements de l'étape d'analyse et d'extraction d'information. Cette étape consiste à analyser tous les documents WSDL du corpus de services web étudiés afin d'extraire l'ensemble des mots-clés potentiels pouvant décrire les fonctionnalités qu'offrent ces services. L'ensemble des traitements textuels utilisés est décrit ci-dessous :

Tokénisation : constitue une première étape appliquée dans n'importe quelle tâche de traitement automatique du langage. La tokénisation consiste à générer, à partir d'un texte, un ensemble de mots simples ou un ensemble d'éléments significatifs individuels. Ce mécanisme se base sur une analyse lexicale permettant d'identifier les éléments en reconnaissant des caractères spéciaux, des espaces de séparation des mots, des chiffres, des ponctuations, etc. Certains mots sont composés de plusieurs mots séparés par une lettre majuscule. Lors de cette étape, nous utilisons des expressions régulières pour extraire des termes simples. Prenons par exemple le document WSDL décrit dans la section 2.2.2, le traitement du nom de la première interface *'ZipCity_ServiceSoap'* produit l'ensemble des mots suivants : *'Zip'*, *'City'*, *'Service'*et *'Soap'*.

Suppression de mots vides : l'objectif de cette étape est d'extraire un ensemble de mots pertinents à partir des descriptions textuelles traités lors de l'étape précédente. Pour cela, nous supprimons tous les éléments jugés inutiles et les mots vides (symboles de ponctuation, pronoms personnels, prépositions, etc.). Le contenu des documents WSDL contient généralement des mots tels que *url, http, host, type, soap, operation, endpoint, binding, post, get, set, request, service, response, etc.* Ces mots peuvent être considérés comme des mots vides. En effet, tous les documents WSDL peuvent contenir ce genre de mots fonctionnels qui ne peuvent donc pas être utilisés pour distinguer les services web étudiés. Par exemple, les mots potentiels

de l'exemple précédent sont ainsi *'Zip'* et *'City'*. Durant cette étape, nous utilisons dans un premier temps une liste de mots vides pour éliminer certains mots. Ensuite, nous utilisons l'outil Standford Log-Linear POS-Tagger [17] pour supprimer toutes autres balises XML/HTML et d'autres mots vides et seuls les mots reconnus comme noms, verbes et adjectifs sont conservés.

Lemmatisation : les descriptions de services contiennent généralement plusieurs mots similaires avec des formes différentes, mais leur origine reste la même. Le processus de lemmatisation permet de regrouper ainsi les différentes variantes d'un mot tout en réduisant des mots dérivés à leurs origines. En général, ces mots avec la même racine (i.e. *lemme*) ont la même signification. Dans cette étape, nous utilisons l'algorithme *Stemmer* de Martin Porter [Porter, 1980] pour identifier des mots qui ont le même lemme et ne garder que la racine. Nous éliminons ainsi les terminaisons de ces mots, et nous gardons seulement la racine pour avoir une forme identique. Par exemple, les mots '*connect*', '*connected*', '*connecting*', '*connection*' et '*connections*' ont le même lemme '*connect*'. Avoir la totalité de ces mots ne fera pas une différence en termes de variations de mots dans la sémantique d'un service web. Toutefois, les mots qui apparaissent le plus souvent sont plus importants que d'autres. Comme nous allons voir dans ce qui suit, nous considérons la fréquence et le nombre d'occurrence des mots pour construire la représentation vectorielle de chaque service web.

La représentation de services est centrée sur les descriptions textuelles et les éléments extraits des documents WSDL. Nous nous basons sur l'ensemble des termes ou mots produits par le processus décrit précédemment pour construire des représentations vectorielles de services. Ces représentations seront utilisées dans nos approches pour extraire les thèmes à partir des descriptions de services web. En recherche d'information, la relation entre les termes/mots et les documents dans un corpus de données peut être représentée avec une matrice dite transactionnelle, où chaque ligne correspond à un document et chaque colonne représente un mot dans le vocabulaire global. Cette représentation facilite l'analyse statistique des données. Nous utilisons donc cette représentation pour décrire les services web étudiés sous forme d'une matrice transactionnelle que nous appelons *STM* (Service Transaction Matrix) dans le reste de ce document. La table 2.1 représente un exemple de matrice transactionnelle STM. Dans la matrice transactionnelle STM, chaque ligne représente la description d'un service web comme un *vecteur* à N dimensions où chaque dimension représente l'apparition d'un mot dans la description de ce service [Aznag, 2015].

La fréquence d'apparition d'un mot dans la description d'un service est calculée en utilisant la technique de *pondération des termes*. Cette technique est très utilisée en recherche d'information. Le poids d'un mot dans une description de service repré-

17. http://nlp.stanford.edu/software/tagger.shtml

Services × mots	m_1	m_2	m_3	...	m_N
s_1	1	2	...	0	0
s_2	0	3	...	5	6
s_3	0	2	...	1	3
...	...	...	...	...	...
s_M	2	1	...	0	1

Table 2.1 – Exemple de matrice transactionnelle STM.

sente l'importance de ce mot dans les documents WSDL. La plupart des techniques de pondération sont fondées sur la combinaison de deux facteurs :

- Facteur *TF (Term Frequency)* de *pondération locale*, quantifiant la représentativité locale d'un terme dans le document. TF a été introduit pour tenir compte de la fréquence d'un terme dans un document. Plus un mot est fréquent dans un document, plus il est important dans sa description.
- Facteur *IDF (Inverse Document Frequency)*, de pondération globale, mesurant la représentativité globale du terme vis-à-vis de la collection des documents. Ce facteur mesure la fréquence d'un terme dans l'ensemble des documents concernés. Il représente une *pondération globale*. En effet, un terme fréquent dans un corpus de données, a plus d'importance qu'un terme moins fréquent.

La combinaison de ces deux mesures TF et IDF donne une bonne approximation de l'importance d'un mot dans un document. Dans notre cas, la fréquence d'apparition d'un mot dans la description d'un service est représentée par le poids *TF-IDF* de ce mot. Nous utilisons l'algorithme TF-IDF [Salton, 1989] pour calculer le poids w_{ij} de chaque mot j pour chaque service i en utilisant l'équation 2.1. Nous produisons ensuite pour chaque service, une représentation vectorielle exprimée sous forme d'un vecteur de comptes $\overline{s_i}$ comme indiqué dans l'équation 2.2 (un service est un N-uplet de mots, $\mathbf{w}_s =< w_1, ..., w_N >$). Nous nous basons sur le principe de la technique VSM (Vector Space Model) pour représenter chaque service comme un vecteur de ces termes. L'ensemble des vecteurs de comptes associés à tous les services, est ainsi utilisé pour former la matrice transactionnelle décrite précédemment.

$$w_{ij} = tf_{ij}.\log(\frac{M}{N_j}) \tag{2.1}$$

Où tf_{ij} est la fréquence du terme j dans le document WSDL i, M est le nombre total des documents WSDL dans l'ensemble de données, et N_j est le nombre de documents WSDL contenant le terme j.

$$\overline{s_i} =< o_1, o_2, ..., o_M >, \qquad \forall \quad o \in \mathbb{R}^+ \tag{2.2}$$

2.3.2 Modèles thématiques probabilistes et extraction de thèmes

Les modèles thématiques probabilistes ont été développés initialement pour la modélisation des documents et plus particulièrement pour l'extraction de thèmes à partir de descriptions textuelles. Ce sont des modèles génératifs probabilistes qui se basent sur l'hypothèse que les documents sont générés par un mélange de thèmes exprimés sous forme de distributions de probabilité sur des mots. Différents modèles thématiques probabilistes ont été efficacement appliqués dans différents domaines d'applications telles que le regroupement (clustering) et la recherche d'information [Blei et al., 2003, Steyvers & Griffiths, 2007, Blei & Lafferty, 2007], le filtrage collaboratif [Hofmann, 2003], la visualisation [Iwata et al., 2008] et la modélisation des données annotées [Blei & Jordan, 2003].

Dans [Aznag et al., 2013b, Aznag, 2015], les auteurs ont utilisé trois modèles thématiques probabilistes décrits ci-dessous (i.e. PLSA, LDA et CTM), pour extraire un ensemble de thèmes à partir des descriptions de services. Ces thèmes ont été utilisés pour regrouper, découvrir et classer des services web. Les résultats de l'évaluation montrent que la méthode probabiliste basée sur le modèle CTM obtient de meilleurs résultats par rapport aux autres méthodes.

Analyse sémantique latente probabiliste (PLSA) : PLSA est un modèle statistique génératif utilisé pour analyser la co-occurrence des données [Hofmann, 1999]. C'est une technique d'apprentissage automatique probabiliste non supervisé qui fait correspondre des vecteurs de comptes de grandes dimensions (telles que celles exprimées dans les matrices transactionnelles de services) en une représentation d'une dimension réduite dans l'espace latent factoriel (*Latent Space Factor*) [Hofmann, 1999]. Le modèle PLSA est basé sur le modèle appelé *Aspect Model*, un modèle de variable latente qui associe une variable de classe non observée $z_k = \{z_1, z_2, ..., z_K\}$ avec chaque observation [Hofmann & Puzicha, 1998].

Allocation dirichlet latente (LDA) : LDA est une autre technique d'apprentissage automatique probabiliste non supervisée qui utilise un modèle probabiliste génératif pour des collections de données discrètes. C'est un modèle qui cherche à découvrir des structures thématiques cachées dans de grands volumes de données [Blei et al., 2003]. LDA est une tentative pour améliorer le modèle PLSA en introduisant un a priori de Dirichlet sur la distribution $P(z|s)$ (i.e. distribution de probabilité des services sur les thèmes) afin de simplifier le problème de l'inférence statistique.

Modèle à thèmes corrélés (CTM) : CTM est un autre modèle thématique génératif probabiliste qui améliore le modèle LDA de base par la modélisation des corrélations entre les thèmes [Blei & Lafferty, 2007] (voir section 2.3.3).

Dans cette thèse, nous utilisons le modèle à thèmes corrélés (CTM) pour extraire les thèmes à partir des descriptions de services web et calculer les corrélations entre

les thèmes extraits. Comme nous allons voir dans la section 2.3.3, le modèle CTM suppose que les mots de chaque document proviennent d'un mélange de thèmes dont chacun est une distribution sur les mots. Dans notre approche, les thèmes sont utilisés comme des techniques efficaces de réduction des dimensions en capturant des relations sémantiques entre mots-thèmes et thèmes-services, exprimées sous forme de distributions de probabilités. Un thème est associé à un groupe de mots ou termes qui peuvent apparaître dans des descriptions de service. Il peut être exprimé sous forme d'une distribution de probabilité sur les mots (i.e. $P(w|z)$ représente la probabilité qu'un mot w appartienne à un thème donné z).

Après avoir identifier tous les mots décrivant les services web, nous calculons la fréquence d'occurrence de ces mots comme décrit dans la section précédente. La technique VSM et l'algorithme TF-IDF ont été utilisés pour représenter chaque service web sous forme d'un vecteur de comptes. L'ensemble de services est donc représenté par une matrice transactionnelle regroupant l'ensemble de vecteurs de comptes de mots qu'il s'agit de modéliser. Les mots représentant un service sont des variables observables et leur occurrence dans une description de service peut être décrite dans un vecteur de comptes comme défini par l'équation 2.2 (un service est un N-uplet de mots, $\mathbf{w}_s =< w_1, ..., w_N >$). Dans notre travail, nous utilisons la matrice transactionnelle produite comme donnée d'entraînement (données observées) pour construire notre modèle thématique probabiliste afin d'identifier un ensemble de thèmes. Une fois le modèle probabiliste construit, nous avons deux distributions de probabilités :

1. $\theta^{(s)} = P(z_k|s)$ la distribution multinomiale sur les thèmes pour un service s.
2. $\phi^{(k)} = P(w|z_k)$ la distribution multinomiale sur les mots pour un thème k.

Á partir d'un modèle de thème, la description d'un service est produite en suivant un certain nombre de règles d'optimisation probabilistes. Ces règles décrivent comment les mots dans la description d'un service sont produits par un mélange de thèmes. Une distribution des thèmes pour un service est choisie dans un premier temps. Puis un thème de la distribution choisie est sélectionné aléatoirement et un mot est tiré de ce thème. Le mot choisi est mis dans la description du service et le processus est répété pour chaque mot dans la description du service. Ce genre de processus génératif est fondé sur l'hypothèse de *sac-de-mots* [Steyvers & Griffiths, 2007] et suppose que les mots dans une description de service apparaissent d'une manière aléatoire (i.e. aucune hypothèse n'est faite à propos de l'ordre des mots au moment où ils apparaissent). La seule question de pertinence de ce modèle est le nombre de fois où un mot est produit dans une description de service. Par conséquent, la description du service peut être décrite par un vecteur $\overline{S}$ (voir l'équation 2.3), où chaque dimension z_k est un nombre entier compris entre 0 et 1 décrivant la probabilité du service s d'être classé dans le thème k. Ce vecteur est la distribution de probabilité sur les thèmes pour le service. Cela permet de réduire la dimensionnalité de l'espace de services et permet à ces services décrits par différents

modèles de description hétérogènes d'être représentés sur le même plan homogène (i.e. l'espace de thèmes) [Aznag, 2015]. Nous décrivons plus en détail, dans la section suivante, le modèle probabiliste à thèmes corrélés.

$$\overline{S} = (z_1, z_2, ..., z_K), \quad z_k \in [0; 1] \tag{2.3}$$

2.3.3 Modèle probabiliste à thèmes corrélés - CTM

Nous présentons dans cette section, le modèle probabiliste thématique que nous utilisons pour extraire un ensemble de thèmes à partir des descriptions de services web. Le modèle à thèmes corrélés, appelé *Correlated Topic Model* (CTM), est un modèle thématique génératif probabiliste qui améliore le modèle LDA [Blei et al., 2003] de base par la modélisation des corrélations entre les thèmes [Blei & Lafferty, 2007]. Le modèle CTM est un modèle hiérarchique qui permet de décrire des collections de données de texte ou d'autres types de données discrètes. CTM utilise une distribution alternative plus souple pour les proportions de thèmes qui tient compte de la structure de covariance entre les composantes. On obtient ainsi un modèle plus réaliste de la structure des thèmes, où la présence d'un thème peut être corrélée avec la présence de l'autre. Contrairement au modèle LDA, où les proportions de thèmes d'un service spécifique sont tirées de Dirichlet et donc où la corrélation entre différents thèmes est ignorée, le modèle CTM propose d'utiliser un autre a priori sur les thèmes que Dirichlet. Une loi logistique normale permet de modéliser les co-apparitions des différents thèmes.

Nous supposons que les mots sont des variables observables et leur occurrence dans une description de service peut être décrite dans un vecteur de comptes comme défini par l'équation 2.2 (voir section 2.3.1, un service est un N-uplet de mots, $\mathbf{w}_s =< w_1, ..., w_N >$). Nous utilisons la notation et la terminologie présentées ci-dessous pour décrire les données, les thèmes et les paramètres dans le modèle CTM :

Mots et services : les seules variables aléatoires observables que nous considérons sont des mots utilisés dans la description des services. Soit w_n désigne le n-ième mot dans le s-ième service, qui est un élément d'un vocabulaire de taille V. Soit $\mathbf{w}_s$ le vecteur de N mots associés au service s.

Thèmes : β est la distribution des thèmes sur les mots. Le modèle contiendra K thèmes $\beta_{1:K}$.

Affectation des thèmes : chaque mot est supposé être choisi de l'un des K thèmes. Le thème assigné z est associé au n-ième mot du s-ième service.

Proportions de thèmes : chaque service est associé à un ensemble de thèmes θ_s, chacun avec une proportion spécifique. θ_s représente une distribution sur les thèmes et reflète les probabilités que les mots soient tirés de chaque thème dans la collection. Nous allons généralement considérer une paramétrisation naturelle de cette multinomiale $\eta = log(\theta_i/\theta_K)$.

La représentation du modèle graphique de CTM est illustrée par la figure 2.2. Le modèle à thèmes corrélés suppose que N mot d'un service résulte du processus génératif suivant. Étant donné les thèmes $\beta_{1:K}$, un vecteur moyen μ de dimension K et la matrice de covariance Σ de K lignes et K colonnes :

1. Pour chaque service s, tirer un vecteur η_s d'une distribution gaussienne multivariée en utilisant le vecteur μ et la matrice de covariance Σ : $\eta_s \sim \mathcal{N}ormal(\mu, \Sigma)$.
2. Pour chaque $n \in \{1, \ldots, N\}$
   - (a) Choisir un thème $z_n|\eta_s$ de $Multinomial(f(\eta_s))$.
   - (b) Choisir un mot $w_n|z_n, \beta_{1:K}$ de $Multinomial(f(\beta_z))$.

Nous considérons que $f(\eta)$ (voir équation 2.4) est la correspondance entre la paramétrisation moyenne et la paramétrisation naturelle des proportions des thèmes.

$$\theta = f(\eta_i) = \frac{\exp \eta}{\sum_i \exp \eta_i} \tag{2.4}$$

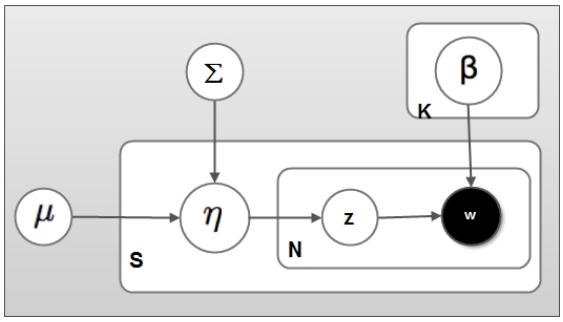


Figure 2.2 – La représentation graphique du modèle CTM.

Le problème principal est alors de calculer la distribution a posteriori $P(\eta, z_{1:N}, w_{1:N})$ des variables latentes données. Puisque cette quantité est intraitable, nous utilisons des techniques approximatives. Dans ce cas, nous choisissons des méthodes variationnelles [Blei & Lafferty, 2007] plutôt que l'échantillonnage de Gibbs [Steyvers & Griffiths, 2007] en raison de la non-conjugaison entre la logistique normale et multinomiale. L'inférence variationnelle utilise l'optimisation plutôt que l'échantillonnage et permet de résoudre un problème d'optimisation. L'idée est de commencer par établir une distribution sur les variables cachées avec des paramètres libres (les paramètres variationnels), puis d'optimiser les paramètres variationnels pour que la distribution converge vers la distribution à posteriori souhaitée. Le problème est donc de borner la log-probabilité d'un service web :

$$\begin{array}{rcc} \log P(w_{1:N}|\mu, \Sigma, \beta) & \geq & E_Q[\log P(\eta|\mu, \Sigma)] + \sum_{n=1}^{N} E_Q[\log P(z_n|\eta)] \\ & + & \sum_{n=1}^{N} E_Q[\log P(w_n|z_n, \beta)] \\ & + & H(Q) \end{array} \tag{2.5}$$

Où l'espérance est calculée par rapport à une distribution variationnelle Q (voir équation 2.6) des variables latentes et $H(Q)$ désigne l'entropie de cette distribution. Le lecteur intéressé peut se reporter à [Blei & Lafferty, 2007] pour plus de détails.

$$Q(\eta, z|\lambda, \nu^2, \phi) = \prod_{i=1}^{K} Q(\eta_i|\lambda_i, \nu_i^2) \prod_{n=1}^{N} Q(z_n|\phi_n) \tag{2.6}$$

Comme cité ci-dessus, l'estimation des paramètres est conduite par l'inférence variationnelle. Les variables et les paramètres du modèle ne sont pas connus initialement, et il faut essayer de les apprendre à partir des données observables, c'est à dire les mots des documents. Dans la représentation graphique présentée dans la figure 2.2, nous pouvons voir que les seules variables observées sont les mots $\mathbf{w}_s =< w_1, ..., w_N >$, alors que toutes les autres variables sont cachées. Étant donné les paramètres du modèle $\{\beta_{1:K}, \mu, \Sigma\}$ et un service web représenté par ces mots $w_{1:N}$, l'algorithme d'inférence variationnelle optimise la borne inférieure (voir équation 2.5) en respectant les paramètres variationnelles et en utilisant l'algorithme *Espérance-Maximisation* [Dempster et al., 1977]) appelé *variational EM*. Dans l'étape "E", nous maximisons la borne par rapport aux paramètres variationnelles en exécutant l'inférence variationnelle pour chaque service. Dans l'étape "M", nous maximisons la borne par rapport aux paramètres du modèle. Les étapes "E" et "M" sont répétées jusqu'à la convergence.

2.4 Analyse de concepts formels

Dans cette section, nous décrivons les concepts et notions de base relatives à la théorie des treillis et à l'Analyse de Concepts Formels (ACF). L'Analyse de concepts formels [Wille, 1982, Ganter & Wille, 1999] est un formalisme mathématique qui permet de restructurer la théorie des treillis [Birkhoff, 1967] (voir section 2.4.1) afin de faciliter son utilisation dans des applications du monde réel et de permettre l'interprétation de ses notions en dehors du cadre théorique. Il s'agit d'une méthode de regroupement conceptuel qui fournit une approche menant à la découverte et la structuration de connaissances. L'ACF a été centrée autour de la notion de concept. De manière informelle, un concept peut être défini comme un groupement d'individus et leurs propriétés communes. L'ACF extrait les concepts à partir de contextes formels (voir section 2.4.2). L'analyse de concepts formels permet ainsi l'identification des groupes d'objets ayant des attributs communs.

2.4.1 Concepts de base et théorie des treillis

Nous décrivons brièvement la théorie des treillis nécessaire à la compréhension de l'ACF. Nous définissons tout d'abord la notion de relation d'ordre et les ensembles partiellement ordonnés qui constituent un cadre théorique pour la structure des treillis.

Définition 2.1 *Soit R une relation binaire définie sur E. R est une relation d'ordre lorsque :*

1. *R est réflexive : $\forall x \in E, xRx$*
2. *R est transitive : $\forall x, y, z \in E, (xRy \wedge yRz \Rightarrow xRz)$*
3. *R est antisymétrique : $\forall x, y \in E, (xRy \wedge yRx \Rightarrow x = y)$*

Une relation d'ordre est souvent notée $\preceq$: $\forall x, y \in E$, si $x \preceq y$, alors on dit que y majore x ou que x minore y.

Nous présentons ci-dessous deux exemples de relations d'ordre :

1. L'inclusion sur $\mathcal{P}(E)$ [18] est définie de la manière suivante : $R = \{(X, Y) \in \mathcal{P}(E)^2 \mid X \subseteq Y\}$.
2. L'ordre sur $\mathbb{R}$ est définie de la manière suivante : $R = \{(x, y) \in \mathbb{R}^2 \mid x \leq y\}$.

Définition 2.2 *Soit $\preceq$ une relation d'ordre sur un ensemble non vide E, on dit que $\preceq$ définit un ordre total sur E si tous ses éléments sont comparables deux à deux par $\preceq$: $\forall x, y \in E^2, x \neq y \Rightarrow (x \preceq y) \vee (y \preceq x)$.*

Un ordre qui n'est pas total est dit partiel.

Définition 2.3 *Un ensemble partiellement ordonné est un couple $(E, \preceq)$ où E est un ensemble non vide d'éléments et $\preceq$ une relation d'ordre partiel.*

Définition 2.4 *Soient $(E, \preceq)$ un ensemble ordonné et S un sous ensemble de E. Un élément $a \in E$ est dit majorant de S lorsque $\forall s \in S, s \preceq a$. De façon duale, $a \in E$ est dit minorant de S lorsque $\forall s \in S, a \preceq s$. Le plus petit majorant (respectivement le plus grand minorant) de S, s'il existe, est unique et appelé supremum ou borne supérieure (respectivement infimum ou borne inférieure) de S et noté $\bigvee S$ (respectivement $\bigwedge S$).*

Définition 2.5 *Un treillis est un ensemble partiellement ordonné $(E, \preceq)$ dans lequel chaque couple d'éléments admet une borne supérieure et une borne inférieure. Un treillis est dit complet si et seulement si toute partie non vide $S \subseteq E$ admet une borne supérieure $\bigvee S$ et une borne inférieure $\bigwedge S$. En particulier, un treillis complet admet un élément maximal (top) noté par $\top$ et un élément minimal (bottom) noté par $\bot$.*

Chaque treillis complet possède au moins un idéal et un filtre, qui sont définis de la manière suivante :

- L'idéal d'un treillis E est un sous-treillis E_i qui est stable par l'opération $\vee$: $\forall x \in E, (y \in E_i) \Rightarrow x \vee y \in E_i$

18. $\mathcal{P}(E)$ est l'ensemble des parties de l'ensemble E, noté aussi 2^E. Il désigne l'ensemble des sous-ensembles de E.

- Le filtre d'un treillis E est un sous-treillis E_f qui est stable par l'opération $\wedge$: $\forall x \in E, (y \in E_f) \Rightarrow x \wedge y \in E_f$.

Tout ensemble partiellement ordonné $(E, \preceq)$ peut être graphiquement représenté par un diagramme appelé **diagramme de Hasse** (ou diagramme de couverture).

Définition 2.6 *Soient $(E, \preceq)$ un ensemble ordonné et x, $y \in E$, on dit que x est couvert par y (ou que y couvre x) si $x \preceq y$ et s'il n'existe pas de $z \in E$ tel que $x \preceq z$ et $z \preceq y$. Dans ce cas, x est appelé prédécesseur de y (resp. y est appelé successeur de x).*

Le diagramme Hasse est obtenu comme suit :
- Tout élément E est représenté par un petit cercle dans le plan.
- Si x, $y \in E$ et $x \preceq y$ alors le cercle correspondant à y doit être au-dessus de celui correspondant à x et les deux cercles sont reliés par un segment.

A partir d'un tel diagramme on peut lire la relation d'ordre comme suit : $x \preceq y$ si et seulement s'il existe un chemin ascendant qui relie le cercle correspondant à x à celui correspondant à y.

Exemple 2.1 *Soit E l'ensemble de tous les diviseurs de 60, E = { 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, 60 } est partiellement ordonné par la relation de divisibilité R. La figure 2.3 montre le diagramme de Hasse correspondant à (E, R).*

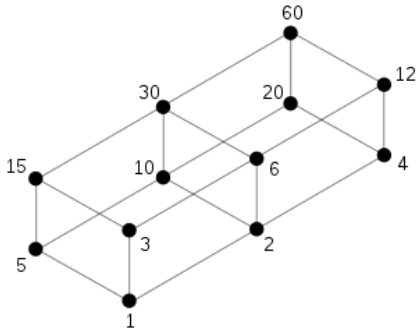


Figure 2.3 – Diagramme de Hasse de (E, R)

Nous définissons ci-dessous la *fermeture et correspondance de Galois* respectivement dans les définitions 2.7 et 2.8. La notion de fermeture est fortement liée à celle de treillis puisque l'ensemble des fermés dans un ensemble E, étant donné un opérateur de fermeture sur E, forme un treillis complet.

Définition 2.7 *On appelle opérateur de fermeture sur un ensemble ordonné $(E, \preceq)$, toute application $\phi : E \to E$ qui vérifie les propriétés suivantes pour tout $x, y \in E$:*

1. *ϕ est extensive : $x \preceq \phi(x)$*
2. *ϕ est monotone croissante : si $x \preceq y$ alors $\phi(x) \preceq \phi(y)$*
3. *ϕ est idempotente : $\phi(x) = \phi(\phi(x))$*

*Un élément $x \in E$ est dit **fermé** pour ϕ si et seulement si $x = \phi(x)$*

Définition 2.8 *Soient $\alpha : O \to A$ et $\beta : A \to O$ deux applications entre deux ensembles partiellement ordonnés $(O, \preceq_O)$ et $(A, \preceq_A)$. On dit que α et β forment une **correspondance de Galois** si elles vérifient les conditions suivantes pour tous $o, o_1, o_2 \in O$ et $a, a_1, a_2 \in A$:*

1. *si $o_1 \preceq_O o_2$ alors $\alpha(o_2) \preceq_A \alpha(o_1)$,*
2. *si $a_1 \preceq_A a_2$ alors $\beta(a_2) \preceq_O \beta(a_1)$,*
3. *$o \preceq_O \beta(\alpha(o))$ et $a \preceq_A \alpha(\beta(a))$.*

Les conditions données dans la définition 2.8 sont équivalentes à la formule suivante : $o \preceq_O \beta(a) \Leftrightarrow a \preceq_A \alpha(o)$

2.4.2 Contexte et concept formel

L'ACF fournit une approche de structuration de connaissances à partir de données. En effet, elle permet de construire sur la base d'une relation binaire une hiérarchie de concepts traduisant une association forte entre un sous-ensemble d'objets et un sous-ensemble d'attributs. L'ACF modélise un ensemble de données grâce à l'utilisation des objets et des attributs [Cole & Eklund, 2001]. L'ensemble des objets, d'attributs et les relations entre un objet et un attribut dans un ensemble de données, forment un *contexte formel* (i.e. définition 2.9) [Ganter & Wille, 1999, van der Merwe et al., 2004].

Définition 2.9 *Un contexte formel est un triplet $\mathcal{C} = (O, A, R)$, où O et A sont respectivement l'ensemble des objets et celui des attributs, et $R \subseteq O \times A$ est une relation binaire reliant O et A et exprimant que $\forall (o, a) \in R$, a est un attribut de l'objet o.*

Un contexte formel peut être représenté sous la forme d'un tableau où les lignes correspondent aux objets et les colonnes correspondent aux attributs. Les cases du tableau sont remplies comme suit : si le $i$ème objet o est en relation R avec le $j$ème attribut a alors la case intersection de la ligne i et la colonne j contient '$\times$' sinon la case est vide.

Exemple 2.2 *Le tableau 2.2 présente un exemple de contexte formel $\mathcal{C}_1 = (O, A, R)$ avec $O = \{o_1, o_2, o_3, o_4, o_5\}$ et $A = \{a_1, a_2, a_3, a_4, a_5\}$.*

En se basant sur la relation binaire R, on associe un ensemble d'attributs à un objet. Un objet $o \in O$ possède l'ensemble des attributs suivant : $oR = \{a \in A \mid oRa\}$. De même, on associe un ensemble d'objets à un attribut. Un attribut $a \in A$ est possédé par l'ensemble des objets suivant : $Ra = \{o \in O \mid oRa\}$. En étendant ces notations à des ensembles, on établit des relations entre des ensembles d'objets et des ensembles d'attributs. D'où la définition de deux opérateurs : le premier allant de $\mathcal{P}(O)$ vers $\mathcal{P}(A)$ et le deuxième de $\mathcal{P}(A)$ vers $\mathcal{P}(O)$.

$O \times A$	a_1	a_2	a_3	a_4	a_5
o_1	×	×	×		
o_2		×	×	×	
o_3	×			×	×
o_4			×	×	×
o_5	×				

Table 2.2 – Un exemple de contexte formel

Les application φ et ψ définies dans la définition 2.10 sont appellées *applications caractéristiques d'une relation binaire.*

Définition 2.10 *Soit $\mathcal{C} = (O, A, R)$ un contexte formel. Pour tout $X \subseteq O$ et $Y \subseteq A$, on définit :*

- $\varphi(X) = \{y \in A \mid \forall x \in X, xRy\} = \{y \in A \mid X \subseteq Ry\} = \bigcap_{x \in X} xR$.
- $\psi(Y) = \{x \in O \mid \forall y \in Y, xRy\} = \{x \in O \mid Y \subseteq Ry\} = \bigcap_{y \in Y} Ry$.

La composition des applications φ et ψ d'un contexte $\mathcal{C} = (O, A, R)$ produit deux opérateurs :

1. $h = \varphi \circ \psi : \mathcal{P}(A) \to \mathcal{P}(A)$
2. $h' = \psi \circ \varphi : \mathcal{P}(O) \to \mathcal{P}(O)$

L'opérateur h' associe à un ensemble d'objets X l'ensemble maximal d'objets dans O partageant les attributs communs avec les objets de X. Le second opérateur h permet d'associer à un ensemble d'attributs Y l'ensemble maximal d'attributs dans A communs aux objets ayant les attributs dans Y.

Les opérateurs h et h' définissent deux fermetures définies respectivement sur $\mathcal{P}(A)$ et $\mathcal{P}(O)$, appelées *fermetures de Galois*. Un ensemble $X \subseteq O$ (resp. $Y \subseteq A$) est dit fermé dans C ssi $X = h'(X)$ (resp. $Y = h(Y)$). Chacun des ensembles des fermés de $(\mathcal{P}(O), \subseteq)$ et $(\mathcal{P}(A), \subseteq)$ est un treillis complet. Les applications φ et ψ forment alors une bijection entre les ensembles des fermés de $\mathcal{P}(O)$ et $\mathcal{P}(A)$. En d'autres termes, à chaque fermé X de $\mathcal{P}(O)$ correspond un unique fermé Y dans $\mathcal{P}(A)$ et inversement. Ainsi, le couple (φ, ψ) forme une correspondance de Galois entre les ensembles partiellement ordonnés $(\mathcal{P}(O), \subseteq)$ et $(\mathcal{P}(A), \subseteq)$ du contexte C et les couples de fermés reliés par cette correspondance constituent les concepts formels. L'ACF est centrée autour de la notion de concept qui est considérée, par les philosophes, comme l'unité de base de la pensée humaine. De manière informelle, un concept peut être vu comme un groupement d'objets et de leurs attributs communs [Messai, 2009]. Par ailleurs, l'ACF a pour principal objectif de déterminer des paires d'ensembles d'objets et d'ensembles d'attributs qui se définissent mutuellement et de manière unique, appelées *concepts formels* (voir définition 2.11).

Définition 2.11 *Soit $\mathcal{C} = (O, A, R)$ un contexte formel. Un concept formel est un couple (X,Y) tel que $X \subseteq O$, $Y \subseteq A$, $X = \psi(Y)$ et $Y = \varphi(X)$. L'ensemble des concepts formels associés au contexte formel $\mathcal{C} = (O, A, R)$ est noté par $\mathcal{CF} = (O, A, R)$.*

Un concept formel associe un ensemble maximal d'objets à l'ensemble maximal d'attributs qu'ils partagent. On appelle l'ensemble $X = \{o \in O \mid \forall y \in Y, (o, y) \in R\}$ l'*extension* du concept. Il représente les objets couverts. Alors que l'ensemble $Y = \{a \in A \mid \forall x \in X, (x, a) \in R\}$ est appellé l'*intention* du concept et représente les attributs partagés. Ces ensembles maximaux d'objets et d'attributs correspondent à des fermés dans $\mathcal{P}(O)$ et $\mathcal{P}(A)$ respectivement. D'où une propriété importante de la correspondance de Galois qui énonce que chaque fermé de l'un des ensembles $\mathcal{P}(O)$ et $\mathcal{P}(A)$ est associé à un unique fermé de l'autre ensemble :

Propriété 2.12 *Si (X, Y) est un concept formel alors X est un fermé de h' $(h'(X) = \psi \circ \varphi(X) = \psi(Y) = X)$ et Y est un fermé de h $(h(Y) = \varphi \circ \psi(Y) = \varphi(X) = Y)$.*

Définition 2.13 *Soit $\mathcal{C} = (O, A, R)$ un contexte formel. Une paire d'ensembles d'objets et d'ensembles d'attributs (X,Y) telle que $X \subseteq O$ et $Y \subseteq A$ est un 1-rectangle dans C si et seulement si $\forall x \in X$ et $\forall y \in Y$, $(x, y) \in R$.*

Dans un contexte formel $\mathcal{C} = (O, A, R)$, un concept formel correspond à un 1-rectangle maximal de la table formée par la relation binaire R. La maximalité du 1-rectangle provient du fait que tous les objets de l'extension doivent avoir tous les attributs de l'intension. Certains concepts formels particuliers peuvent être identifiés dans un contexte formel $\mathcal{C} = (O, A, R)$. Il s'agit des concepts objets et attributs qui sont définis dans la définition 2.14 :

Définition 2.14 *Pour un objet $o \in O$, la paire (h'({o}),φ({o})) est un concept formel appelé concept objet. Pour un attribut $a \in A$, la paire (ψ({a}),h({a})) est un concept formel appelé concept attribut.*

2.4.3 Treillis de concepts

Les concepts formels de $\mathcal{CF}(O, A, R)$ sont ordonnés par une relation de subsomption, notée $\preceq$ (voir définition 2.15).

Définition 2.15 *Etant donné deux concepts formels (X_1, Y_1) et (X_2, Y_2) de $\mathcal{CF}(O, A, R)$, on dit que (X_1, Y_1) est un sous-concept de (X_2, Y_2) et (X_2, Y_2) est un sur-concept de (X_1, Y_1), noté $(X_1, Y_1) \preceq (X_2, Y_2)$, si et seulement si $X_1 \subseteq X_2$, ou de manière équivalente $Y_2 \subseteq Y_1$. La relation $\preceq$ est dite relation de subsomption.*

La relation d'ordre hiérarchique $\preceq$ définie entre les concepts formels s'appuie sur deux inclusions duales entre les ensembles d'objets et d'attributs. Un concept plus général est caractérisé par un plus large ensemble d'objets qui partagent un ensemble plus restreint d'attribut. De façon duale, un concept plus spécifique est caractérisé par un plus petit ensemble d'objets partageant un ensemble plus grand d'attributs. Le treillis de concepts

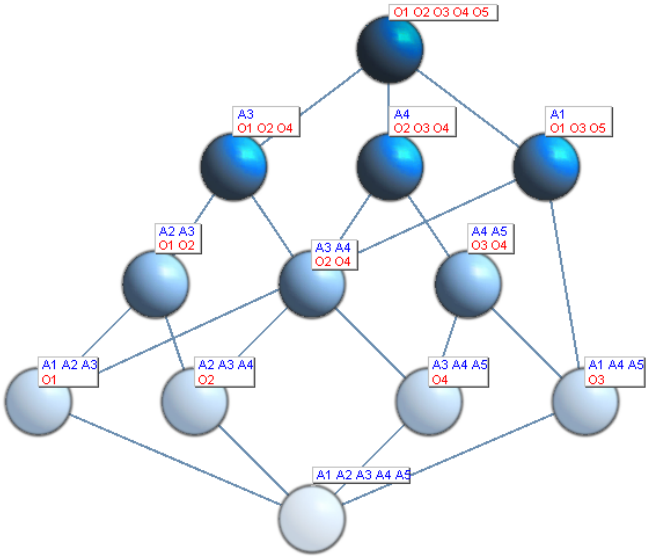


Figure 2.4 – Diagramme de Hasse du treillis $\tau(\mathcal{C}_1)$

(voir définition 2.16) est une représentation des données d'un contexte formel qui met en évidence les groupements possibles entre objets et attributs ainsi que les relations d'inclusion entre ces groupements.

Définition 2.16 *L'ensemble de tous les concepts de* $\mathcal{CF}(O, A, R)$ *muni de l'ordre* $\preceq$ *forme un treillis complet appelé treillis de concepts [Birkhoff, 1967], et noté* $\tau(\mathcal{C})$.

La figure 2.4 montre le diagramme de hasse du treillis de concepts correspondant au contexte $\mathcal{C}_1$ donné dans la table 2.2. Un cercle représente un concept et les arcs entre les cercles matérialisent la relation d'ordre du plus général (en haut) vers le plus spécifique (en bas).

La représentation des concepts est réduite, elle est fondée sur l'héritage des attributs et des objets entre les concepts du treillis. Les attributs sont placés en haut du treillis, tous les descendants d'un noeud étiqueté par un attribut héritent de cet attribut. En ce qui concerne les objets, ils sont placés en bas du treillis et tous les ancêtres d'un noeud étiqueté par un objet le partagent. Ainsi, l'extension d'un concept est obtenue en considérant tous les objets qui apparaissent dans ses descendants dans le treillis et son intension est obtenue en considérant tous les attributs qui apparaissent dans ses

ancêtres dans le treillis. La représentation graphique du treillis de concepts, sous la forme d'un diagramme de Hasse facilite l'interprétation de la relation entre les objets et les attributs. D'ailleurs, il est toujours possible de retrouver le contexte formel correspondant et inversement. L'infimum et le supremum du treillis de concepts $\tau(\mathcal{C})$ sont caractérisés par le théorème de base de l'ACF (i.e. théorème 2.1).

Thérème 2.1 *Le treillis de concepts formels $\tau(\mathcal{C})$ est un treillis complet dont l'infimum et le supremum sont donnés par :*

$\bigwedge_{t\in T}(A_t, B_t) = (\bigcap_{t\in T} A_t, h(\bigcup_{t\in T} B_t))$

$\bigvee_{t\in T}(A_t, B_t) = (h'(\bigcup_{t\in T} A_t), \bigcap_{t\in T} B_t)$

où T est un ensemble d'index et $\forall t \in T$, (A_t, B_t) est un concept formel.

2.5 Découverte de motifs fréquents

Dans cette section, nous introduisons essentiellement des définitions et notions de bases sur l'extraction de motifs qui seront utiles pour comprendre le travail proposé dans le chapitre 6. Nous décrivons également la tâche d'extraction de motifs fréquents tout en introduisant quelques représentations condensées des motifs fréquents.

2.5.1 Définitions et notions de base

Pour simplifier, nous considérons ici des sources de données comme des tables simples contenant l'information à étudier. La définition 2.17 introduit le contexte d'extraction décrivant un ensemble d'objets et un ensemble d'attributs. Le contexte d'extraction est sous la forme de contexte formel décrit précédemment dans le cadre de l'analyse de concepts formels (voir section 2.4).

La table 2.3 montre un exemple de contexte d'extraction.

Définition 2.17 (Contexte d'extraction) *Un contexte d'extraction est un triplet $\mathcal{D} = (\mathcal{O}, \mathcal{A}, R)$, décrivant un ensemble fini $\mathcal{O}$ d'objets (ou transactions), un ensemble fini $\mathcal{A}$ d'attributs (ou items) et une relation binaire R entre $\mathcal{O}$ et $\mathcal{A}$ vérifiant $R \subseteq \mathcal{O} \times \mathcal{A}$. Un couple $(o, a) \in R$ (noté aussi oRa), signifie que l'objet $o \in \mathcal{O}$ possède l'attribut $a \in \mathcal{A}$.*

Nous allons maintenant introduire la notion de motifs fréquents. Dans la littérature, un motif d'attributs est aussi appelé un *itemset*.

Définition 2.18 (Motif) *Soient $\mathcal{A}$ un ensemble d'attributs. Un motif (d'attributs) est un sous-ensemble de $\mathcal{A}$.*

Dans ce qui suit, les motifs seront notés tout simplement sous forme de chaîne plutôt que sous forme d'ensembles bien que l'ordre soit sans importance (i.e., a_1a_2 au lieu de

	Attributs							
Objets	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
o_1	×		×		×		×	
o_2		×	×		×		×	
o_3	×		×		×			×
o_4	×			×		×		×
o_5		×	×			×		×
o_6		×	×		×	×		×

Table 2.3 – Exemple d'un contexte d'extraction $\mathcal{D}$.

$\{a_1, a_2\}$). Un motif d'attributs est trié dans un ordre lexicographique et appelé également ***motif***. Les notions de support et de fréquence d'un motif sont introduites comme suit :

Définition 2.19 (Support, fréquence) *Un objet $o \in \mathcal{O}$ supporte le motif d'attributs X si $\forall a \in X, (o, a) \in R$ (ou par abus d'écriture, $X \subseteq o$). Le support $supp(X)$ d'un motif X d'attributs est l'ensemble des objets qui le supportent. Sa fréquence $\mathcal{F}(X)$ est le cardinal du support.*

Dans la littérature, le terme *support* est parfois utilisé pour désigner le nombre d'occurrences d'un motif (ce que nous appelons ici *fréquence*), tandis que la *fréquence* est utilisée pour exprimer en pourcentage la fraction d'occurrences relativement au nombre total de transactions.

Définition 2.20 (Motif fréquent) *Un motif d'attribut est fréquent (ou minsup-fréquent) si sa fréquence dépasse un seuil minsup fixé par l'utilisateur.*

Sur notre exemple (table 2.3), les objets o_1 et o_2 contiennent le motif $a_3a_5a_7$, son support est o_1o_2. Sa fréquence vaut 2, et si le seuil de fréquence est fixé à 1 ou 2, ce motif est fréquent.

2.5.2 Extraction de motifs fréquents

La découverte de motifs fréquents est devenue une tâche trés importante dans la fouille de données et dans d'autres domaines d'applications. Elle a été initiée par Agrawal et al. [Agrawal et al., 1993] et elle correspond à trouver les ensembles d'attributs (ou items) qui apparaissent simultanément dans au moins un certain nombre d'objets (ou transactions) définis dans un contexte d'extraction (voir définition 2.17). Ces ensembles d'items sont appelés des motifs fréquents. Dans cette section, nous décrivons l'extraction des motifs fréquents à l'aide du formalisme unificateur développé par Mannila et Toivonen [Mannila & Toivonen, 1997]. En effet, les algorithmes d'extraction utilisent les

mécanismes de ce formalisme dans leur grande majorité : il s'agit d'un probléme de recherche, dont l'espace à parcourir est progressivement élagué grâce aux propriétés du probléme.

L'extraction de motifs fréquents est caractérisée par un espace de recherche à parcourir, pour découvrir des éléments satisfaisant la contrainte de fréquence. Cet espace de recherche des motifs est défini comme le langage $\mathcal{L}_{\mathcal{A}}$ construit sur l'alphabet $\mathcal{A} = \{a_1, a_2, \ldots, a_{|\mathcal{A}|}\}$ constitué de tous les attributs d'un contexte d'extraction $\mathcal{D}$. C'est aussi l'ensemble des parties de $\mathcal{A}$. L'extraction de motifs devient une tâche algorithmiquement difficile quand l'espace de recherche est très grand : si $|\mathcal{A}| = n$, alors la taille de l'espace de recherche est 2^n. L'espace $\mathcal{L}_{\mathcal{A}}$ dans lequel les motifs sont recherchés correspond à un treillis (cf. figure 2.5). En plaçant l'ensemble vide en haut, la deuxiéme ligne contient les singletons, la troisiéme les paires, etc... L'inclusion entre deux motifs définit une relation de *spécialisation* [Mitchell, 1982]. Ainsi, si un motif X est inclus dans un motif Y, on dit que Y est plus *spécifique* que X ou que X est plus *général* que Y.

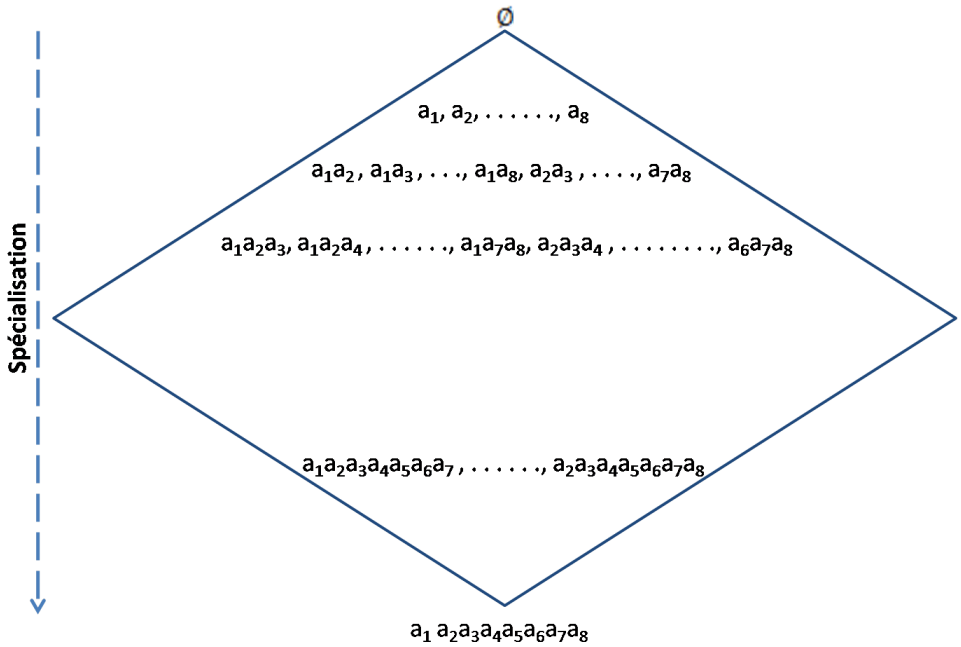


Figure 2.5 – Treillis représentant le langage $\mathcal{L}_{\mathcal{A}}$ avec $\mathcal{A} = \{a_1, a_2, \ldots, a_8\}$.

Pour parcourir l'espace de recherche des motifs, il faut utiliser une stratégie, dérivée des propriétés de la contrainte de fréquence. Cette contrainte de fréquence est *anti-monotone* (voir définition 2.21), c'est-à-dire que si un motif est fréquent, ses sous-ensembles le sont également. Réciproquement, si un motif n'est pas fréquent, tous ses sur-ensembles ne

le sont pas non plus. Une stratégie couramment employée consiste donc à parcourir l'espace de recherche par niveaux, en examinant les motifs potentiellement fréquents par ordre croissant de leurs tailles. On commence par calculer la fréquence des singletons, puis celle des paires. Les paires infréquentes sont éliminées et tous leurs sur-ensembles élagués. Ensuite, les motifs de longueur trois sont obtenus par fusion des paires fréquentes, etc. L'ensemble des motifs du langage $\mathcal{L}_A$ qui vérifient une certaine contrainte q dans $\mathcal{D}$, est appelé *théorie* [Mannila & Toivonen, 1997] et est noté $Th(\mathcal{L}_\mathcal{A}, \mathcal{D}, q)$. Certaines contraintes vérifient des propriétés qui facilitent le calcul de $Th(\mathcal{L}_\mathcal{A}, \mathcal{D}, q)$ comme les contraintes anti-monotones.

Définition 2.21 (Contrainte monotone ou anti-monotone) *Une contrainte q est monotone (resp. anti-monotone) suivant la relation de spécialisation si et seulement si pour tout motif satisfaisant q, ses spécialisations (resp. généralisations) satisfont également la contrainte q.*

La contrainte de fréquence, qui consiste à sélectionner les motifs dont le nombre d'occurrences dans $\mathcal{D}$ dépasse un seuil donné, est la plus populaire des contraintes anti-monotones. Une contrainte monotone (ou anti-monotone) par rapport à la spécialisation est en fait une simple fonction croissante (ou décroissante) par rapport à la spécialisation. Les contraintes monotones/anti-monotones sont bien adaptées à l'extraction de motifs. Tout d'abord, elles peuvent être aisément combinées par conjonction ou disjonction. La négation d'une contrainte monotone (resp. anti-monotone) est une contrainte anti-monotone (resp. monotone). La conjonction d'une contrainte monotone et d'une contrainte anti-monotone n'est ni monotone, ni anti-monotone. La négation d'une contrainte monotone ou anti-monotone donne directement sa condition d'élagage :

Propriété 2.22 (Condition d'élagage) *Si un motif X ne satisfait pas la contrainte monotone (resp. anti-monotone) q, alors toutes les généralisations (resp. spécialisations) de X ne satisfont pas la contrainte q.*

Cette condition d'élagage bénéficie de la variation de la contrainte pour garantir que soit toutes les généralisations, soit toutes les spécialisations ne vérifieront plus la contrainte.

2.5.3 Représentation condensée de motifs

Dans cette section, nous introduisons quelques représentations condensées de motifs fréquents. Les représentations condensées de motifs ont été introduites pour réduire l'espace de recherche de façon à améliorer les temps de calcul et limiter le nombre de motifs produits en s'appuyant sur les propriétés de classes d'équivalence. L'objectif est d'obtenir un ensemble plus restreint de motifs tout en conservant les informations

nécessaires pour en déduire tous les motifs fréquents. Il existe deux types de représentations condensées : des représentations exactes et approximatives. Les représentations condensées exactes permettent de déduire avec exactitude toutes les fréquences des motifs fréquents. Il existe plusieurs représentations condensées de motifs telles que les représentations par motifs fermés [Pasquier et al., 1999], les représentations par motifs maximaux [Burdick et al., 2005], les représentations par motifs libres [Boulicaut & Bykowski, 2000], les représentations par itemsets non-dérivables [Calders & Goethals, 2007], les représentations par intervalles selon l'opérateur de fermeture préfixée [Soulet, 2006], etc.

Dans cette thèse, nous étudions plus en détail les représentations condensées par motifs fréquents fermés et par motifs fréquents maximaux que nous utilisons pour capturer la sémantique commune maximale d'un ensemble de services web (voir chapitre 6). La figure 2.6 montre la relation entre les motifs fréquents, les motifs fermés et les motifs fréquents maximaux.

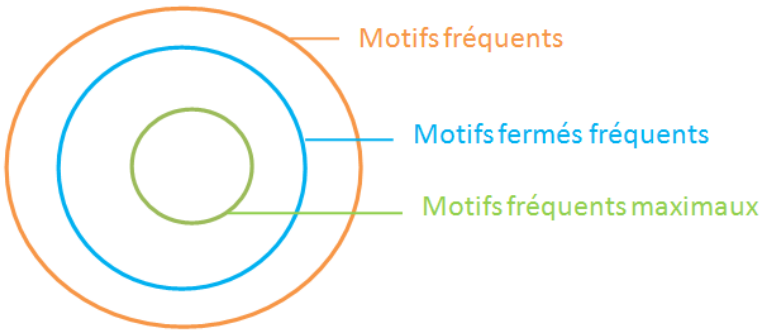


Figure 2.6 – Relation entre les motifs fréquents, les motifs fermés fréquents et les motifs fréquents maximaux.

La représentation par motifs maximaux [Burdick et al., 2005] est composée de l'ensemble de motifs formant la bordure positive (voir définition 2.23). C'est une repésentation approximative puisque elle ne permet pas de calculer précisément la fréquence des motifs mais de dériver une borne inférieure de celle-ci. La recherche des éléments, qui vérifient l'anti-monotonie de la contrainte, peut s'assimiler à la découverte de la *frontiére* qui sépare le treillis en deux parties : l'ensemble des motifs valides d'un côté (noté S), l'ensemble des motifs invalides de l'autre. Cette séparation en deux parties est possible car la contrainte est préservée par généralisation. Nous utilisons le terme *bordure* qui désigne l'union des bordures positive et négative. La bordure positive contient les motifs valides qui jouxtent la frontiére : ce sont les maximaux valides au sens de l'inclusion, c'est-à-dire les plus longs. La bordure négative contient les motifs invalides qui sont de l'autre côté de la frontiére : ce sont les minimaux invalides, ou les plus courts. La figure 2.7 schématise ces

différentes notions.

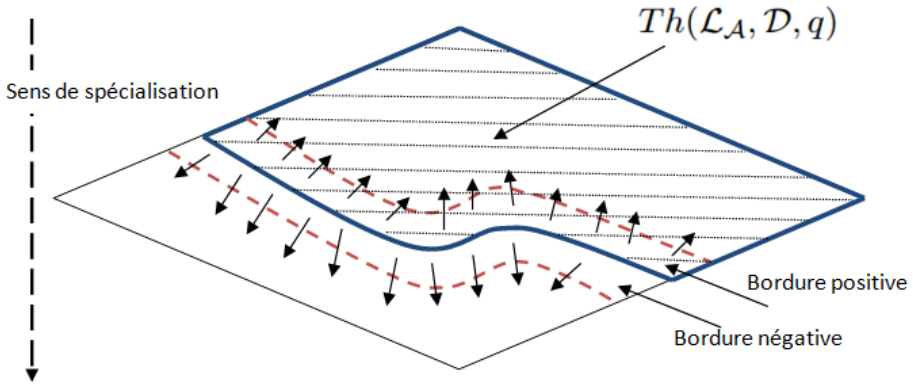


Figure 2.7 – Bordures positive et négative.

Les définitions 2.23 et 2.24 présentent respectivement la bordure positive et négative d'une théorie. Prenons par exemple, le treillis de motifs contenus dans le contexte d'extraction présentée dans la table 2.3. En considérant la contrainte q "avoir une fréquence au moins égale à 2", nous avons la frontiére entre les motifs vérifiant q et ceux qui ne la vérifient pas. La bordure positive est : $\{a_1a_3a_5, a_1a_8, a_2a_3a_5, a_3a_5a_7, a_2a_3a_6a_8, a_3a_5a_8\}$. La bordure négative est : $\{a_1a_3a_8, a_1a_6, a_1a_7, a_1a_5a_8, a_4, a_2a_7, a_5a_6, a_2a_5a_8\}$.

Définition 2.23 (Bordure positive) *L'ensemble des motifs maximaux (au sens de l'inclusion) qui satisfont la contrainte q dans $\mathcal{D}$, est noté $Bd^+(Th(\mathcal{L}_\mathcal{A}, \mathcal{D}, q))$ et constitue la bordure positive de $Th(\mathcal{L}_\mathcal{A}, \mathcal{D}, q)$.*
$Bd^+(Th(\mathcal{L}_\mathcal{A}, \mathcal{D}, q)) = \{X \in Th(\mathcal{L}_\mathcal{A}, \mathcal{D}, q) \mid \forall Y \; tq \; X \subset Y, \; \neg q(Y)\}$

Définition 2.24 (Bordure négative) *L'ensemble des motifs minimaux (au sens de l'inclusion) qui ne satisfont pas la contrainte q dans $\mathcal{D}$ est noté $Bd^-(Th(\mathcal{L}_\mathcal{A}, \mathcal{D}, q))$ et constitue la bordure négative de $Th(\mathcal{L}_\mathcal{A}, \mathcal{D}, q)$.*
$Bd^-(Th(\mathcal{L}_\mathcal{A}, \mathcal{D}, q)) = \{X \in \mathcal{L}_\mathcal{A} \setminus Th(\mathcal{L}_\mathcal{A}, \mathcal{D}, q) \mid \forall Y \; tq \; Y \subset X, \; q(Y)\}$

La représentation fondée sur les motifs fermés (voir définition 2.25) est une représentation exacte [Pasquier et al., 1999, Boulicaut & Bykowski, 2000]. Elle tire son origine de la théorie des treillis et plus précisément des travaux autour de l'analyse de concepts formels [Ganter & Wille, 1999]. Formellement, les motifs maximaux des classes d'équivalence sont appelés des motifs *fermés* en référence à l'opérateur de fermeture de Galois. Comme décrit dans la section 2.4 (voir définitions 2.7 et 2.8, page 34), une connexion de Galois (ou correspondance de Galois) [Birkhoff, 1967] permet d'associer deux ensembles partiellement ordonnés. Dans notre contexte, ce sont les motifs d'attributs et les motifs d'objets

d'un contexte d'extraction qui sont reliés par une connexion de Galois particuliére. En se basant sur la définition des opérateurs de *fermetures de Galois* h et h' (respectivement sur $2^{\mathcal{A}}$ et $2^{\mathcal{O}}$) (voir page 35), nous définissons la notion de motifs fermés comme suit :

Définition 2.25 (Motif fermé, concept formel) *Soit $\mathcal{D} = (\mathcal{O}, \mathcal{A}, R)$ un contexte formel. Un motif A d'attributs est fermé dans $\mathcal{D}$ ssi $h(A) = A$. Un motif O d'objets est fermé dans $\mathcal{D}$ si et seulement si $h'(O) = O$. Un concept formel [Wille, 1982] (O, A) associe deux motifs fermés, O d'objets et A d'attributs, tels que $O = \psi(A)$ (ou $A = \varphi(O)$). O et A sont respectivement appelés extension et intention du concept formel.*

Couramment, un *motif fermé* désigne un motif fermé d'attributs (l'intension d'un concept formel). Dans notre exemple présenté dans la table 2.3, le motif a_3a_7 n'est pas fermé car les objets de son extension o_1 et o_2 ont aussi tous deux en commun l'attribut a_5. Cela signifie que l'attribut a_5 est toujours présent avec le motif a_3a_7. La fermeture de a_3a_7 est donc $a_3a_5a_7$. La définition 2.26 introduit la notion de *fermeture d'un motif.*

Définition 2.26 (Fermeture d'un motif (d'attributs)) *La fermeture d'un motif (d'attributs) X dans $\mathcal{D}$ est $h(X, \mathcal{D}) = \bigcap\{o \in \mathcal{O} \mid X \subseteq o\}$.*

Chacun des ensembles de fermés de $(2^{\mathcal{A}}, \subseteq)$ et de $(2^{\mathcal{O}}, \subseteq)$ est un treillis complet. L'ensemble de tous les concepts formels de $\mathcal{D}$ est noté $\mathcal{CF} = (O, A, R)$. Dans chacun de ces concepts formels, nous trouvons un motif fermé 1-fréquent. La représentation graphique du treillis de concept, sous la forme d'un diagramme de Hasse (voir section 2.4.1), facilite l'interprétation de la relation entre les objets et les attributs. Remarquons qu'à partir de ce treillis, il est toujours possible de retrouver le contexte formel correspondant et inversement. La figure 2.8 présente le treillis de Galois obtenu sur l'exemple de la table 2.3. Chaque sphére correspond à un concept formel.

Pour tous $X, X' \in 2^{\mathcal{A}}$, $X \sim X'$ si et seulement si $\psi(X) = \psi(X')$ ($supp(X) = supp(X')$). On peut alors déduire une structuration du treillis des parties de $\mathcal{A}$ (voir figure 2.5) sous forme de classes d'équivalence comme indiqué dans la définition 2.27 et la définition 2.28.

Définition 2.27 (Classe d'équivalence 1) *La classe d'équivalence $\mathcal{R}_{supp}(X)$ d'un motif X de $2^{\mathcal{A}}$ est définie par : $\mathcal{R}_{supp}(X) = \{X' \in 2^{\mathcal{A}} \mid supp(X') = supp(X)\}$.*

Définition 2.28 (Classe d'équivalence 2) *La classe d'équivalence d'un motif X de $2^{\mathcal{A}}$ est définie par : $\mathcal{R}_{supp}(X) = \{X' \subseteq X \mid X \sim X'\}$*
$X \sim X' \Rightarrow (\mathcal{F}(X) = \mathcal{F}(X'))$
$X \subseteq X' \wedge (\mathcal{F}(X) = \mathcal{F}(X')) \Rightarrow X \sim X'$.

Les classes ainsi définies sont habituellement appelées classes d'équivalence de fréquence. La classe d'équivalence d'un motif X d'attributs est notée $\mathcal{R}_{supp}(X)$ ou bien

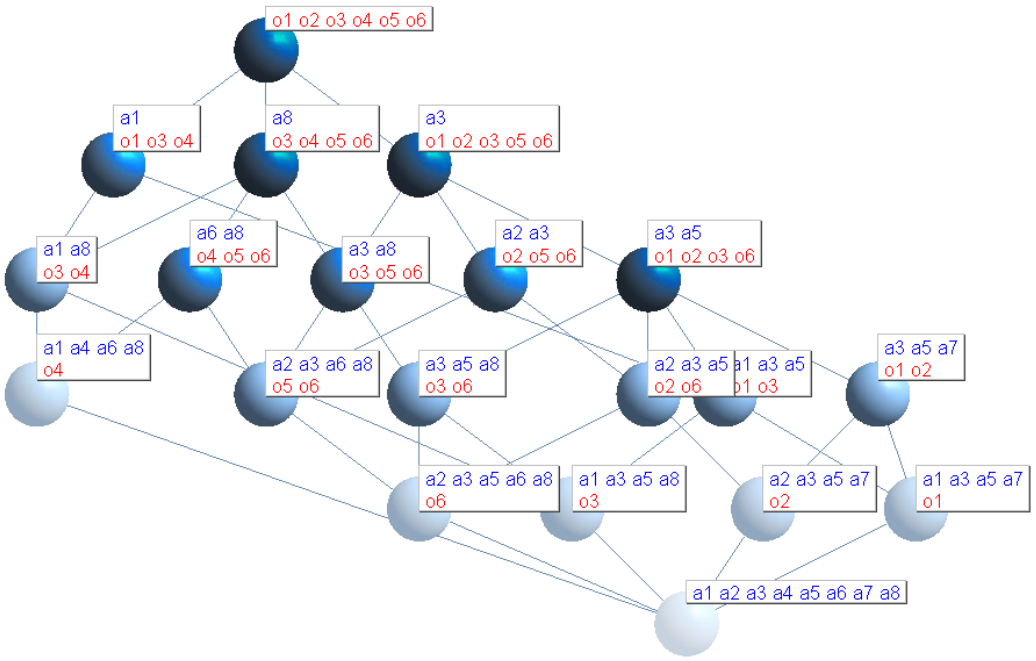


Figure 2.8 – Treillis de Galois de l'exemple de la table 2.3.

$[X]_{supp}$. Reprenons la table 2.3, les motifs a_7, a_3a_7, a_5a_7 et $a_3a_5a_7$ ont tous la même extension o_1o_2. Ils forment donc une classe d'équivalence.

Cette partition du treillis en classes d'équivalence est important pour l'extraction de motifs. Puisque les motifs d'une classes d'équivalence partagent la même fréquence, il est possible de ne considérer que les motifs minimaux ou le motif maximal (i.e le motif fermé) de chaque classe d'équivalence au lieu de parcourir tout l'espace de recherche. Dans notre exemple, seul a_7 (qui est minimal) ou $a_3a_5a_7$ (qui est maximal) sera considéré.

Définition 2.29 (Motif fermé 2) *Soit X un motif fermé. X est le plus grand élément de $\mathcal{R}_{supp}(X)$. On ne peut pas lui ajouter d'attribut sans faire baisser sa fréquence : $\mathcal{F}(X \cup \{a_i\}) < \mathcal{F}(X)$.*

Les motifs fermés sont donc les éléments maximaux des classes d'équivalence. Une classe d'équivalence a un unique motif fermé. Les motifs dit libres [Boulicaut et al., 2003] sont les éléments minimaux. Une classe d'équivalence a un ou plusieurs motifs libres. Remarquons que les motifs libres peuvent être utilisés comme *générateurs* d'ensembles fermés.

En fixant dans notre exemple le seuil minimal *minsup* de fréquence à 2, on dénombre 13 classes d'équivalence. Il y a en effet 13 motifs fermés 2-fréquents (cf. figure 2.8) : $\{a_1$, a_3, a_8, a_1a_8, a_2a_3, a_3a_5, a_3a_8, a_6a_8, $a_1a_3a_5$, $a_2a_3a_5$, $a_3a_5a_7$, $a_3a_5a_8$, $a_2a_3a_6a_8\}$. Les motifs 2-fréquents sont au nombre de 31.

2.6 Conclusion

Dans ce chapitre, nous avons présenté les généralités et les concepts de base sur les services web. Ensuite, nous avons décrit les différentes techniques permettant l'extraction de toutes les informations décrivant les services web à partir des documents WSDL. Ces informations seront utilisées dans nos approches pour produire des représentations vectorielles de services. Nous avons également introduit les modèles thématiques probabilistes. Dans le cadre de cette thèse, nous avons utilisé plus spécifiquement le modèle à thèmes corrélés CTM pour extraire les thèmes à partir des descriptions de services. Enfin, nous avons introduit les concepts de base et le cadre théorique de l'analyse de concepts formels et l'extraction de motifs fréquents. Dans cette thèse, l'analyse de concepts formels est utilisée pour construire le réseaux d'intéraction de services web dans le chapitre 4 et pour construire un treillis de concepts fréquents dans le chapitre 6. Les motifs fréquents seront utilisés par notre système de recommandation combinés avec les modèles thématiques probabilistes pour capturer la sémantique commune maximale d'un ensemble de services (voir chapitre 6).

Nous présentons dans le chapitre suivant, une synthèse de l'état de l'art des approches de la littérature proposées dans le cadre de la découverte, composition, recommandation des services web, détection de communautés ainsi qu'un ensemble d'algorithmes d'extraction de motifs fréquents, motifs fermés fréquents et motifs fréquents maximaux.

3

État de l'art

Sommaire

3.1 Introduction

Avec le nombre croissant de services web disponibles en ligne, plusieurs problèmes sont apparus. Ces derniers sont notamment liés à l'organisation de ces services, la découverte, la disponibilité, la diversité, etc. Dans ce chapitre, nous présentons un état de l'art permettant de situer notre travail. Dans un premier temps, nous présentons dans la section 3.2, différents modèles de description syntaxique et sémantique de services web. Nous discutons ensuite quelques approches proposées dans la littérature traitant la découverte de services web. La section 3.3 est consacrée aux systèmes de recommandation et aux approches proposées dans la littérature dans ce contexte. Dans la section 3.4.1, nous étudions quelques approches basées sur les réseaux pour modéliser les services web et leurs interactions. Nous présentons également dans la section 3.4.2, un ensemble de

travaux antérieurs dans le cadre de la composition de services web. Dans la section 3.4.3, nous introduisons la notion de communauté et différents travaux de recherche proposés pour la détection de communautés de services web. Enfin, nous présentons dans la section 3.5, quelques algorithmes d'extraction de motifs fréquents.

3.2 Modèles de description et découverte de services web

La découverte de services web consiste à découvrir un ensemble de services web satisfaisant une requête utilisateur. Les systèmes de découverte de services web, proposées dans la littérature, se différencient essentiellement par trois points importants, à savoir :

- Les langages ou modèles de description des services web utilisés,
- L'organisation de la recherche,
- Les techniques utilisées dans le processus d'appariement et de sélection de services.

Dans cette section, nous présentons d'abord un aperçu global des différents modèles de description syntaxique et sémantique de services. Ensuite, nous discutons quelques approches de découverte et de classement de services web proposées dans la littérature.

3.2.1 Modèles de description de services web

Pour que deux systèmes (ou plus) communiquent, il doit y avoir un accord ou un contrat sur les interfaces utilisées et les formats de données. Dans le modèle RPC (Remote Procedure Call) traditionnel, où toutes les différences entre l'informatique locale et l'informatique distribuée sont masquées, un langage de description d'interface IDL (Interface Description Language) est utilisé pour spécifier ces interfaces. Les types de données qu'un tel IDL offre, sont des abstractions des types de données trouvés dans les langages de programmation réels pour permettre l'interopérabilité entre les différentes plates-formes. De cette façon, la génération automatique de code sur les deux, le client et le côté serveur, sont possibles. Le protocole SOAP, successeur de XML-RPC, est basé exactement sur le même modèle. Les interfaces de service SOAP sont généralement définies par un document Web Service Description Langage (WSDL) pour décrire les méthodes avec leurs entrées et sorties attendues. XML Schema est utilisé pour décrire les schémas pour ces entrées et sorties. Le standard WSDL [Christensen et al., 2001] se base sur une grammaire XML et permet de décrire et publier le format et les protocoles d'un service web de manière homogène. La description d'un service web consiste à définir une interface exposant une ou plusieurs opérations ayant des entrées/sorties encodés en XML. D'autres services peuvent interagir avec ces opérations en utilisant des messages SOAP. SOAP sert de protocole de transmission entre l'utilisateur et le fournisseur du service (voir la section 2.2 pour plus de détails). Étant donné que les documents WSDL sont lisibles par les machines, il est possible de générer automatiquement des "stubs" de code qui visent à améliorer la productivité des développeurs d'applications.

Les échanges entre les services web ne se restreignent pas à des messages XML-SOAP. En effet, les services web de type REST sont basés sur l'architecture du Web et ses standards de base : HTTP et URI [Fielding, 2000]. Les requêtes sont simples et sous forme URI avec des verbes HTTP, du type GET/POST et des entêtes de requête pour décrire les informations envoyées. L'appel d'un service REST avec les valeurs d'entrée données peut renvoyer des valeurs de sortie en XML, JSON, RSS, etc. Notons qu'il est possible de décrire les services REST avec WSDL 2.0 [1] [Haas, 2005]. Mais cela rend évidemment impossible la génération automatique de code ou la validation automatique des requêtes et des réponses. A ce jour, il n'y a pas de norme acceptée pour décrire les descriptions de services REST qui pourraient former la base de descriptions sémantiques. Certaines APIs de services REST sont décrites par le langage WADL (Web Application Description Language) dans des fichiers XML ou des micro-format HTML appelés hRESTS [Kopecky, 2012]. Le WADL est un format de fichier basé sur XML qui permet de décrire des services web REST. L'adresse d'une ressource est spécifiée comme un template URI, avec différents types de paramètres qui peuvent être remplis dans l'URI et également dans les en-têtes de la requête HTTP. hRESTS (HTML pour les services RESTful) est un modèle simple permettant de capturer des détails pertinents des APIs Web afin de faciliter la tâche d'invocation.

La plupart du temps, la description syntaxique de l'interface d'un service n'est pas suffisante. En effet, deux services peuvent avoir la même définition syntaxique mais effectuer des fonctions significativement différentes. Ainsi, la sémantique des données et le comportement du service doivent être documentés et compris. Cela se fait normalement sous la forme d'une description textuelle qui est facilement compréhensible par un être humain. Cependant, les machines ont d'énormes problèmes pour comprendre un tel document et ne peuvent pas extraire suffisamment d'informations pour utiliser correctement et automatiquement un tel service d'une manière sémantique. Pour résoudre ce problème, les services doivent être annotés sémantiquement ; le service résultant est appelé un service Web sémantique (SWS) ou un service sémantique RESTful (SRS). Ces descriptions sémantiques supplémentaires des propriétés du service peuvent par conséquent conduire à un niveau plus élevé d'automatisation pour des tâches telles que la découverte, l'invocation, la composition, etc. L'annotation sémantique de ces services consiste à établir des correspondances entre des éléments de la description et des concepts d'un ensemble d'ontologies de référence. La sémantique des éléments de l'interface d'un service est décrite par des références à des concepts appropriés et des règles. Ces règles sont formellement définies dans une ontologie partagée dans un langage d'ontologie standard du W3C comme RDFS ou OWL2. Les modèles actuels de description de services sémantiques regroupent différents langages à base d'ontologie, à savoir, OWL-S [Martin et al., 2004a], WSML [de Bruijn & Lausen, 2005], le standard W3C SAWSDL

1. https ://www.w3.org/TR/wsdl20/

[Farrell & Lausen, 2007], et Link USDL [Pedrinaci & Leidig, 2011] qui est simplement USDL (Unified Service Description Language) [Bansal et al., 2005] modélisé en RDFS. Ces différents langages de description se distinguent principalement par leur sémantique formelle basée sur la logique. OWL-S (Ontology Web Language for Services) est une ontologie de description des services web, dont les objectifs sont de résoudre les ambiguités et de rendre la description d'un service compréhensible par une machine. Il est compatible avec des formats de description syntaxiques tels que WSDL. Un service peut être caractérisé par des conditions préalables spécifiées dans le langage Semantic Web Rule Language (SWRL) [Ian Horrocks, 2004]. WSMO (Web Service Modeling Ontology) est une ontologie basée sur le Web Service Modeling Framework WSMF proposé par Fensel et Bussler [Fensel & Bussler, 2002]. Les ontologies WSMO et OWL-S partagent le même but d'automatisation des tâches liées aux services. Dans le modèle WSMO, la description des services web sémantiques est indépendante du langage utilisé pour décrire ces services.

SAWSDL (Semantic Annotations for Web Services Description Language) est un langage sémantique de description de service Web [Farrell & Lausen, 2007]. Il permet d'annoter les descriptions WSDL 2.0 tout en supportant WSDL 1.1. En effet, l'objectif de SAWSDL est de définir comment une annotation doit être réalisée, tout en laissant le choix du langage utilisé pour la description sémantique. SAWSDL fournit les mécanismes permettant d'attacher des concepts décrits dans des ontologies aux annotations des descriptions WSDL. Il augmente l'expressivité du langage WSDL avec la sémantique en utilisant des concepts analogues à ceux utilisés dans OWL-S. Aussi, les services REST décrits par hRESTS peuvent être sémantiquement annotés par des propositions telles que SA-REST [Sheth et al., 2007] et MicroWSMO [Kopecky et al., 2008]. Les services décrits par WADL avec, par exemple, les services SBWS [Battle & Benson, 2008]. Les services REST définis par WSDL 2.0 peuvent être annotés directement avec SAWSDL. Le micro-format MicroWSMO/SA-REST ainsi que l'approche SBWS ajoute des annotations de type SAWSDL aux descriptions de service basées respectivement sur hRESTS et WADL [GOMADAM et al., 2010]. La figure 3.1 illustre une comparaison des langages de description basés respectivement sur le protocole SOAP et REST.

Actuellement, la plupart des services web publics disponibles sur Internet sont décrits par le langage de description standard WSDL ou selon le paradigme REST. A l'heure actuelle, il n'y a pas de statistiques publiques sur les services web sémantiques disponibles sur Internet. La plupart des services sémantiques décrits par OWL-S, WSML, WSDL-S, et SAWSDL sont disponibles uniquement dans des collections de test spécifiques. De plus, la création et la maintenance des ontologies peuvent être très difficiles et impliquent un très grand effort humain [Atkinson et al., 2007, Lausen & Haselwanter, 2007]. Dans cette thèse nous étudions plus en détails les services web décrits soit par le langage syntaxique WSDL ou bien le langage sémantique SAWSDL. Comme nous allons voir

dans le reste de ce document, les approches proposées dans cette thèse sont qualifiées non-logiques et utilisent des techniques de recherche d'information. Ainsi, nos approches peuvent être appliquées à n'importe quel modèle de description de services. Cependant, nous supposons que les services web utilisés pour la méthode de découverte de services susceptibles d'être composable, doivent être décrits par des documents WSDL et/ou SAWSDL.

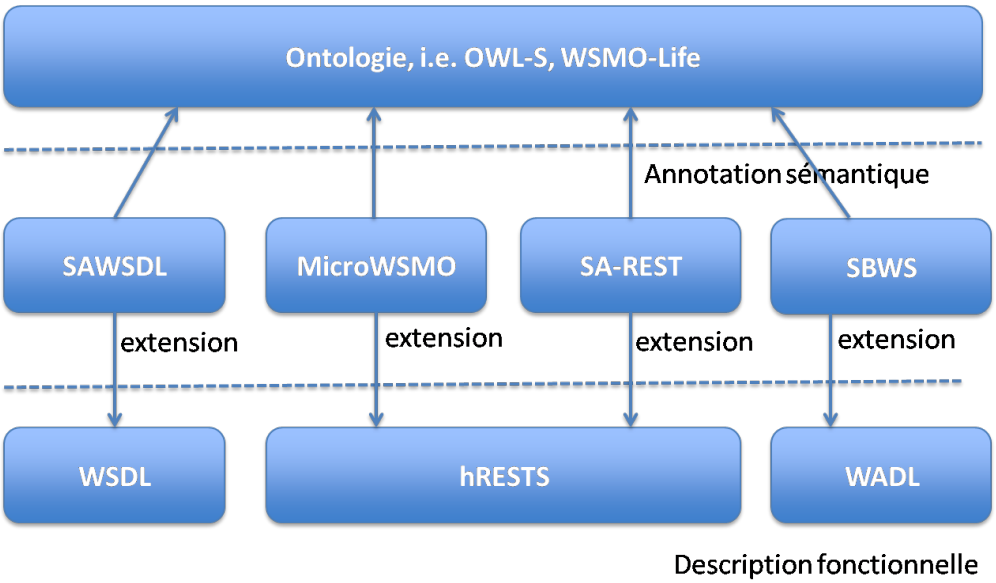


Figure 3.1 – Comparaison de quelques modèles de description de services web [GOMADAM et al., 2010]

3.2.2 Découverte et sélection de services web

Une fois les services web décrits par l'un des modèles de description sont publiés dans un annuaire ou portails de services en ligne, les utilisateurs peuvent chercher et découvrir les services qui répondent à leurs besoins. Comme mentionné précédemment, la découverte de services web est généralement définie comme un processus de localisation de services existants répondant aux besoins des utilisateurs.

Nous discutons dans cette section quelques approches proposées dans la littérature pour améliorer le processus de découverte de services. Ces approches utilisent généralement différentes techniques et mécanismes afin de trouver les services web qui répondent aux exigences des utilisateurs. Elles peuvent être classées en trois catégories en fonction de la nature du mécanisme utilisé pour l'appariement de services : les approches logiques, les approches non-logiques et les approches hybrides [Klusch, 2014]. Les approches classées non-logiques se basent essentiellement sur des techniques de recherche d'information et de fouille de données (data mining) telles :

- Mesures de similarité [Fethallah et al., 2010, A. Abdullah, 2016],
- Regroupement et classification [Elgazzar et al., 2010, Zhang et al., 2016],
- Modèles thématiques probabilistes [Cassar et al., 2013, Aznag et al., 2013b],
- Analyse de concept formels et classification conceptuelle [Chollet et al., 2011, Aznag et al., 2014],
- ...

Tandis que les approches classées logiques se basent essentiellement sur des approches déductives et utilisent un moteur de raisonnement permettant de déduire de nouvelles connaissances à partir des relations définies dans des descriptions de services sémantiques [Fang et al., 2006, Parsa & Fakhr, 2009]. Les approches qui combinent les mécanismes logiques et non-logiques sont qualifiées d'approches hybrides [Klusch & Kaufer, 2009, Klusch et al., 2010, Schulte et al., 2010, Klusch & Kapahnke, 2012]. Les approches logiques offrent une amélioration sur les insuffisances des méthodes syntaxiques mais elles sont très complexes. Les approches de découverte non-logiques permettent de réduire la complexité des matchmackers (apparieurs) sémantiques en analysant la fréquence d'occurrence de certains termes dans des descriptions de services et déterminant la sémantique implicite dans les descriptions de services. Les approches hybrides combinent les mécanismes logiques et non-logiques [Klusch & Kaufer, 2009, Klusch et al., 2010, Schulte et al., 2010, Klusch & Kapahnke, 2012]. La majorité de ces approches supportent soit le modèle de description OWL-S ou SAWSDL, mais seulement quelques-unes supportent WSML, ou d'autres formats de description. Les approches hybrides souffrent de problèmes d'interopérabilité semblables à ceux des méthodes basées sur la logique. Aussi, la majorité de ces approches se basent généralement sur des ontologies pour calculer des similarités sémantiques entre les services web étudiés [Fethallah et al., 2010]. Cependant, la création et la maintenance des ontologies peuvent être difficiles et impliquent un très grand effort humain [Lausen & Haselwanter, 2007]. Dans cette thèse, nous nous intéressons seulement aux approches qualifiées non-logiques que nous discutons dans ce qui suit.

Diverses techniques d'apprentissage telles que la classification et les algorithmes de regroupement ont été largement utilisés pour améliorer le processus de découverte de services web. Par exemple, Elgazzar et al. [Elgazzar et al., 2010] et Liu et Wong [Liu & Wong, 2009] appliquent des techniques d'extraction d'information pour extraire un ensemble de caractéristiques à partir des documents WSDL(i.e. le contenu, les types, les messages, les ports et le nom de service). Ces caractéristiques sont utilisées pour regrouper les services ayant des fonctionnalités similaires dans le même groupe. En organisant les services web en clusters, le regroupement de services peut améliorer l'efficacité de découverte de services [Wu et al., 2014, Zhang et al., 2016]. Plusieurs travaux de recherche se basent sur les modèles thématiques probabilistes pour extraire les thèmes et modéliser les documents. Dans le contexte des services web, le travail présenté dans [Ma et al., 2007], utilise le modèle PLSA (Analyse sémantique latente probabiliste)

[Hofmann, 1999] pour capturer des concepts sémantiques de la requête et les descriptions de services afin que le mécanisme d'appariement de services s'effectue au niveau du concept. L'extension de ce travail a été reporté dans [Ma et al., 2008] en proposant un mécanisme de découverte de services appelé CPLSA. Une approche de découverte de services proposée dans [Cassar et al., 2013], utilise les modèles thématiques probabilistes PLSA et LDA pour extraire des thèmes à partir des descriptions de services écrites dans le langage OWL-S. Dans [Chen et al., 2013, Wu et al., 2014], les auteurs proposent une approche de regroupement de services intégrant des tags de service et le contenu des documents WSDLs en utilisant le modèle probabiliste LDA (Allocation Dirichlet Latente). Néanmoins, le modèle LDA est incapable de modéliser la corrélation entre les thèmes extraits à partir de descriptions de services web [Blei & Lafferty, 2007]. Pour pallier cette limite, Aznag et al. [Aznag et al., 2013b, Aznag et al., 2013a] proposent une méthode probabiliste pour découvrir les services web en utilisant le modèle probabiliste à thèmes corrélés CTM (Correlated Topic Models) [Blei & Lafferty, 2007]. Ce dernier permet de capturer les thématiques et leurs corrélations à partir de descriptions de services web. Les thèmes sont utilisés comme techniques efficaces de réduction des dimensions, en capturant des relations sémantiques entre mots-thèmes et thèmes-services, exprimées sous forme de distributions de probabilités. Dans [Wang et al., 2015], les auteurs proposent une nouvelle approche de découverte de services permettant d'extraire et faisant correspondre des groupes de thèmes communs. Ces derniers sont utilisés pour faire correspondre les requêtes des utilisateurs aux services web appropriés.

La découverte de services web présente un axe de recherche émergeant. De nombreuses approches utilisant l'analyse de concepts formels (voir chapitre 2, section 2.4) ont été également proposées dans la littérature. Dans [Chollet et al., 2011], les auteurs proposent une approche de découverte basée sur les treillis de concepts afin d'organiser les services. Le registre de service est considéré comme un contexte formel où les services sont les objets et les types WSDL, les caractéristiques fonctionnelles et non-fonctionnelles (i.e. caractéristiques de sécurité) sont des attributs. Dans [Driss et al., 2010], les treillis de concepts sont utilisés pour trouver les services pertinents de haute qualité (i.e. services qui correspondent aux besoins fonctionnels). Les objets représentent les services et les attributs sont les caractéristiques de qualité de service. Une approche similaire basée sur l'ACF a été proposée dans [Azmeh et al., 2010] pour faciliter la classification et la sélection des services web. Un treillis de concepts est construit pour organiser les services web à partir des contextes formels où les objets sont les services web et les attributs représentent les caractéristiques de la qualité de service associés à ces services. Dans [Aznag et al., 2014], les auteurs proposent une approche basée sur le modèle probabiliste à thèmes corrélés CTM pour extraire des thèmes à partir des descriptions de service et construire un regroupement hiérarchiques de services en exploitant la corrélation entre les thèmes. Les auteurs utilisent l'analyse de concepts formels pour organiser les groupes

hiérarchiques construits en treillis de concepts en fonction de leurs thèmes associés. Ainsi, la découverte de services peut être réalisée plus facilement en utilisant le treillis de concepts construit.

La majorité des systèmes existants découvrent des services web qui sont très similaires et/ou redondants. Afin d'éviter la redondance, la diversité doit être prise en considération dans les systèmes de découverte et/ou de recommandation. Dans cette thèse, nous proposons une nouvelle méthode de classement de services permettant la diversification des résultats de la découverte de services web tout en maintenant la qualité des services web découverts (voir chapitre 5, page 95). Nos travaux sur la découverte et la sélection de services web étendent les travaux proposés récemment par Aznag et al. [Aznag et al., 2013b, Aznag et al., 2013a, Aznag et al., 2014]. Plus précisément, nous proposons une méthode de diversification des résultats de la découverte en se basant à la fois sur la notion de pertinence (ou similarité), la diversité et la densité des services. Notre système de découverte permet de découvrir des services divers correspondant à une requête donnée et minimiser la redondance dans la liste renvoyée à l'utilisateur. Récemment, le problème de diversification des résultats de la recherche sur le Web a attiré beaucoup d'attentions et est devenu plus important dans le domaine de la Recherche d'Information. En effet, la diversification des résultats de recherche est utilisée comme un moyen efficace pour satisfaire diverses préférences et besoins des utilisateurs sur le Web. Le problème est généralement formulé comme l'optimisation d'une fonction objective qui spécifie un compromis entre la pertinence des documents retournés par rapport à une requête donnée et la dissimilarité entre ces documents [Gollapudi & Sharma, 2009]. Cela est essentiellement semblable à l'idée de l'algorithme MMR (i.e. Maximal Marginal Relevance) qui a été proposé dans [Carbonell & Goldstein, 1998] pour re-classer les résultats de la requête. Il est également appliqué souvent pour la recherche de documents sur Internet. L'algorithme MMR combine la pertinence de la requête et la nouveauté de l'information, en mesurant la dissimilarité d'un document candidat par rapport aux autres documents dans la liste des résultats classée. Nous allons étudier plus en détail dans le chapitre 5, la problématique de la diversité des résultats de recherche dans le cadre de la découverte des services web. Nous allons décrire ainsi notre proposition pour diversifier la liste des services renvoyés à l'utilisateur, tout en prenant en compte leur pertinence.

3.3 Systèmes de recommandation de services web

Les systèmes de recommandation visent à assister l'activité de recherche de l'utilisateur en lui proposant des ressources pertinentes répondant à ses besoins. Ces systèmes sont utilisés dans divers domaines tels que la recherche documentaire, le commerce électronique, etc. Les systèmes de recommandation s'inscrivent dans le cadre de la personnalisation

de l'accès à l'information. En d'autre terme, un système de recommandation a pour but d'aider l'utilisateur à choisir un produit ou un service le plus adéquat à ses besoins parmi une large liste d'éléments [Werthner et al., 2007]. La recommandation et la sélection des services web constituent un domaine de recherche fondamentale depuis l'apparition des technologies de services web. Les moteurs de recherche de services web disponibles sur Internet exploitent largement les techniques de recherche par mots-clés. Certains de ces moteurs de recherche ne tiennent pas compte de la sémantique des services ainsi que des caractéristiques non-fonctionnelles (tels que la qualité de service). Les performances d'un tel système de recommandation de services sont donc limitées. Au cours des dernières années, la recommandation de service web a été un domaine de recherche actif et de nombreuses techniques ont été proposées. Il existe plusieurs techniques de recommandation, issues notamment du domaine de l'apprentissage automatique et du data mining. Ces techniques peuvent être classées en trois catégories : le filtrage collaboratif, les approches basées sur le contenu et les approches hybrides. Dans cette section, nous présentons ces techniques de recommandation tout en décrivant quelques approches proposées dans la littérature dans le contexte des services web. Le lecteur intéressé peut se référer à [Kantor, 2009] pour plus d'informations sur les systèmes de recommandation.

3.3.1 Méthodes de filtrage collaboratif

Les méthodes de filtrage collaboratif sont largement utilisées dans les systèmes de recommandation qui recommandent des éléments (les services web dans notre contexte) en se basant sur la similitude des différents utilisateurs [Zheng et al., 2011b, Zheng et al., 2009]. Ces systèmes de recommandation génèrent une ou des recommandations susceptibles d'intéresser un utilisateur à travers l'analyse à la fois (1) des opinions de cet utilisateur sur les ressources qu'il a consultées ainsi que (2) celles des autres utilisateurs sur les ressources qu'ils ont consultées [Goldberg et al., 1992]. Cependant, la majorité des approches de recommandations basées sur le filtrage collaboratif, le contenu ou les approches hybrides se reposent sur l'hypothèse que les services sont indépendants. Par conséquent, elles renvoient une liste de services web à recommander dont de nombreux services sont très similaires ou redondants. Pour pallier ce problème, plusieurs approches ont été proposées en incorporant les préférences QoS des utilisateurs et leurs divers intérêts sur les services web [Zheng et al., 2011b, Kang et al., 2016]. Kang et al. [Kang et al., 2016] proposent une approche de recommandation de services en intégrant les préférences de QoS potentielles d'un utilisateur et la diversité des intérêts des utilisateurs sur les services Web. Les intérêts et les préférences QoS de l'utilisateur concernant les services web sont d'abord extraits en explorant l'historique d'usages des services web. Ensuite, les auteurs calculent des scores pour les services candidats, en mesurant leur pertinence via les intérêts historiques et potentiels de l'utilisateur, et leur utilité QoS. Les auteurs ont proposé un algorithme permettant de classer les services web candidats en se basant

sur la notion de diversité dans les graphes et les scores de pertinences calculés. Dans [Zheng et al., 2011b], les auteurs proposent une approche de prédiction de services web basée sur la QoS pour prédire les valeurs de la QoS manquantes en se basant sur les informations QoS provenant des services web et d'utilisateurs similaires.

La sélection des services web basée sur la QoS prend en charge la sélection optimisée des services en considérant les attributs QoS des services avec des fonctionnalités similaires, ainsi que les préférences des utilisateurs du service [Zheng et al., 2011a]. La qualité de la recommandation à partir de ces approches dépend de la qualité des informations QoS disponibles pour les services web. Dans un autre contexte, Mehta et al. [Mehta et al., 2004], proposent une architecture pour la recommandation basée sur la médiation de service dans laquelle les auteurs tiennent compte de deux autres dimensions de la description de service : la qualité et l'utilisation de motifs. L'utilisation de motifs permet de trouver des applications avec un mode d'utilisation similaire à l'application qui fait la demande et renvoie une liste de recommandations contenant les services utilisés par ce type d'applications.

3.3.2 Méthodes basées sur le contenu

Les systèmes de recommandation basés sur le contenu recommandent des ressources ou services semblables à ceux que l'utilisateur apprécie en fonction des caractéristiques de ces ressources (i.e. les fonctionnalités). Ces systèmes s'appuient également sur des évaluations effectuées par un utilisateur sur un ensemble de services et compare le contenu sémantique des ressources aux besoins exprimés par l'utilisateur [Pazzani & Billsus, 2007]. Les systèmes de recommandation de services web basés sur le contenu recommandent à un utilisateur donné des services similaires à ceux précédemment sélectionnés par cet utilisateur. En général, ces systèmes se basent sur l'analyse des similarités du contenu (par exemple, documents WSDL et descriptions courtes) des services web. Les techniques de recommandation basées sur le contenu peuvent être classées en deux catégories : les approches syntaxiques [Blake & Nowlan, 2007] et sémantiques [Lécué & Delteil, 2007]. Les méthodes basées sur la syntaxe utilisent généralement des techniques de recherche d'informations, d'extraction de connaissances telles que la recherche et l'analyse linguistique [Fethallah et al., 2010]. En revanche, les méthodes basées sur la sémantique recommandent des services web en exploitant la description sémantique de leurs fonctionnalités via des descriptions ontologiques [Sheth et al., 2007, Segev & Sheng, 2012]. Les aspects sémantiques peuvent être également introduits dans les approches basées sur le contenu en utilisant des modèles thématiques probabilistes [Xu et al., 2008, Qinjiao Mao et al., 2013], ou la découverte des motifs [Maccatrozzo et al., 2014]. Qinjiao Mao et al. utilisent le modèle probabiliste LDA pour modéliser les intérêts des utilisateurs. En utilisant un tel modèle probabiliste sur la génération de profils d'utilisateurs, les relations entre les utilisateurs, les ressources et les intérêts sont construites [Qinjiao Mao et al., 2013]. Les motifs sont aussi utilisés

dans des systèmes de filtrage collaboratif. Ces systèmes sont basés par exemple sur les motifs fréquents, motifs fréquents maximaux, clustering, l'analyse de concepts formels (i.e. treillis de concepts) ou le modèle de Markov [Suguna & Sharmila, 2013].

3.3.3 Méthodes hybrides

En combinant les méthodes de filtrage collaboratif et les méthodes basées sur le contenu, les méthodes hybrides peuvent incorporer les avantages des deux méthodes tout en éliminant les faiblesses de chaque approche. Lécué [Lécué, 2010] a proposé un système de recommandation de services web sémantiques. L'auteur considère les voisins d'un utilisateur actif en calculant les similarités entre les différentes informations personnelles des utilisateurs. Ensuite, les services web manipulés par des utilisateurs similaires, à l'exception des services déjà utilisés par l'utilisateur actif, sont classés en fonction de leur similarité sémantique avec les services utilisés par l'utilisateur final actif. Yao et al. [Yao et al., 2013] ont proposé une approche de recommandation de services hybride qui exploite à la fois les données de notation et les données de contenu des services via un modèle d'aspect. Dans leur approche, les intérêts des utilisateurs sont représentés par un ensemble de variables latentes. Cependant, les préférences QoS des utilisateurs ne sont pas prises en compte dans cette approche. Dans [Yao et al., 2015], les auteurs considèrent à la fois les données de notation (QoS) et le contenu sémantique des services web en utilisant un modèle génératif probabiliste. Dans [Karim, 2014], l'auteur propose un système de recommandation évolutif et indépendant du domaine. Le système proposé se base sur le filtrage collaboratif pour adapter les recommandations aux préférences des utilisateurs et assurer un degré de diversité et de nouveauté dans les éléments proposés.

Dans cette thèse, nous proposons un système de recommandation hybride qui combine les deux approches décrites ci-dessus (le filtrage collaboratif et l'approche basée sur le contenu). Notre approche exploite les avantages de ces deux techniques et permet des recommandations plus précises. L'originalité du système proposé vient de la combinaison des modèles thématiques probabilistes et les motifs fréquents (plus précisément, l'extraction de motifs fréquents maximaux) pour capturer la sémantique commune maximale d'un ensemble de services web [Naïm et al., 2016]. Dans notre approche, nous combinons également la similarité sémantique et un ensemble d'attributs de la qualité de services web pour classer et sélectionner les différents services candidats (voir chapitre 6).

3.4 Réseaux de services web et applications

Les systèmes naturels ou artificiels peuvent être représentés par des réseaux, c'est-à-dire des noeuds reliés par des liens. Dans le cadre des services web, les noeuds peuvent représenter les services, leurs opérations ou leurs paramètres, et les liens peuvent symboliser les relations d'interaction qui exisent entre ces noeuds. Dans cette section, nous allons

dresser un panorama de travaux antérieurs utilisant des approches réseaux afin de modéliser et représenter les services web et leurs interactions.

3.4.1 Topologie des réseaux de services web

Les travaux présentés dans [Chu et al., 2016], proposent une méthode de construction de réseaux de services web dynamiques basés sur la qualité de services pour diminuer la complexité et améliorer l'efficacité des services. Les niveaux de régularisation sont ajustés de façon dynamique en fonction de la variation du degré de connexion avec un impact minimal sur les performances de calcul. En se basant sur les historiques de QoS extraits à partir de fichiers de log de services, les réseaux dynamiques sont construits pour révéler les relations entre les services. La sélection de services est ensuite guidée par les valeurs de QoS observées des services connexes.

Dans [Kil et al., 2009], une étude basée sur les aspects topologiques a été effectuée. Divers réseaux de services web ont été construits en se basant sur les correspondances exactes et approximatives. Les résultats montrent que ces réseaux exhibent les propriétés du petit monde et la distribution sans échelle (voir chapitre 6). Les travaux reportés dans [Liu et al., 2007], concernent la construction d'un réseau d'interaction à partir d'un registre UDDI. Les relations entre les services sont fournies par leurs fournisseurs. Lors de la génération du réseau, les auteurs supposent seulement que le nombre de paramètres en sortie d'un service doit être supérieur ou égal au nombre de paramètres en entrée d'un autre service pour qu'un lien soit créé dans le réseau. Les auteurs ont également proposé une méthode de recherche de compositions en se basant sur le réseau construit. Le réseau d'interaction présenté dans [Dekar & Kheddouci, 2008] est un réseau non orienté dont les noeuds sont les services et un lien d'interaction entre deux services est pondéré par le nombre de fois où les deux services sont composés. Les services qui sont dans une même composition, sont regroupés en cluster à l'aide d'un algorithme de b-coloration. En général, un algorithme de coloration consiste à affecter à tous les noeuds d'un graphe une couleur de telle sorte que deux noeuds adjacents ne doivent pas porter la même couleur. Une coloration est dite b-coloration, si pour chaque couleur C_i, il existe au moins un noeud v_i coloré C_i dont le voisinage est coloré par toutes les autres couleurs. Un second algorithme a été également proposé dans ces travaux permettant de maintenir et de mettre à jour la classification des services web. La classification est opérée sur le réseau d'interaction de services web.

D'autres travaux de recherche utilisent des propriétés topologiques des réseaux dans le but d'accroître l'efficacité de la composition [Gekas & Fasli, 2007, Oh et al., 2008]. Dans [Gekas & Fasli, 2007], les auteurs s'intéressent à la connectivité du réseau pour guider un algorithme de recherche de compositions. La pertinence d'un service pour faire partie d'une composition est liée à son importance en termes de connectivité avec son environnement. Ainsi, un rang traduisant la notoriété d'un noeud donné du réseau lui est

associé. Le service de plus grand rang est évalué en premier et les autres sont placés dans une file d'attente prioritaire par rang décroissant. Ce processus est itéré jusqu'à ce que les compositions satisfaisant la requête soient découvertes. Des expérimentations sont conduites sur un réseau extrait à partir d'un ensemble de 2450 descriptions sémantiques générées automatiquement. Dans le cadre des services web, ce travail est considéré comme un des rares travaux intéressants s'inspirant des grands graphes de terrain. Cependant, les résultats ne sont pas validés sur des données réelles.

Le travail reporté dans [Oh et al., 2008] s'inscrit aussi dans cette optique qui considère la composition de services web comme domaine d'application des réseaux complexes. Les auteurs proposent plusieurs types de réseaux pour représenter des services caractérisés par une description syntaxique. Les propriétés caractéristiques des grands graphes de terrain à savoir la propriété petit monde et la distribution en loi de puissance des degrés ont été étudiées à partir d'une collection de 984 descriptions WSDL réelles provenant de sites spécialisés. Les résultats montrent que tous les réseaux étudiés possèdent bien la propriété petit monde et que la distribution des degrés suit une loi de puissance. Les auteurs ont également proposé dans [Oh & Lee, 2009], un générateur de descriptions synthétiques WDSL appelé WSBen (Web Service Discovery and Composition Benchmark) basé sur la topologie du réseau de paramètres. WSBen est conçu pour étudier et tester les performances des algorithmes de découverte et de composition de services web. L'utilisateur a la possibilité de spécifier différents modèles de réseaux. Ceci permet de générer des descriptions en adéquation avec les modèles de réseaux complexes (Newman-Watts-Strogatz, Barabasi-Albert) ou les modèles de réseaux aléatoires (Erdös-Rényi). Néanmoins, les auteurs n'abordent pas l'aspect sémantique des descriptions de services web et ne considèrent que les graphes.

Les travaux présentés dans [Cherifi et al., 2010, Cherifi et al., 2013a, Cherifi et al., 2013b] s'inscrivent dans la continuité de [Oh et al., 2008]. Ils définissent un ensemble de réseaux extraits à l'aide de l'outil Web Services Network EXTractor (WS-NEXT) [Cherifi et al., 2011] pour la composition sur la base de services. Ces services sont décrits à la fois dans un langage syntaxique (WSDL) et un langage sémantique (SAWSDL). L'exploration expérimentale de ces réseaux permet de mettre en évidence les propriétés caractéristiques des grands graphes de terrain (la propriété petit monde et la distribution sans échelle).

Plusieurs travaux traitant la diversité dans les grands graphes ont été proposés dans [Mei et al., 2010, Tong et al., 2011, Li & Yu, 2013]. Ils proposent des mesures de classement qui capturent la diversité et la pertinence dans les graphes.

Dans le cadre de notre travail, nous proposons une méthode permettant de modéliser et construire une structure de services web sous forme de réseaux (i.e. réseaux d'interaction et réseaux de similitude) (voir chapitre 4). Nous exploitons le réseau d'interaction de services web construit et la notion de diversité dans les graphes pour identifier les services

web qui sont susceptibles d'être composables (voir chapitre 5). Le réseau de similitude est utilisé pour traiter les services sous forme de communautés de services (voir chapitre 7).

3.4.2 Interaction et composition de services web

La composition de services web est abordée dans un grand nombre de travaux de recherche. Elle permet la satisfaction d'une demande complexe de l'utilisateur, par la combinaison de plusieurs services web si cette demande n'est pas satisfaite par un des services existants. Selon [Martin et al., 2004b], la composition de services web est définie comme étant le processus de sélection, de combinaison et d'exécution de services en vue d'accomplir un objectif donné. La définition de la composition de services la plus utilisée est celle présentée dans [Benatallah et al., 2005]. Les auteurs la considèrent comme un moyen efficace pour créer, exécuter, et maintenir des services qui dépendent d'autres services.

En général, les approches permettant la réalisation d'une composition de services web, sont nombreuses. Dans cette section, nous allons décrire deux catégories de ces approches : la composition *dynamique* et la composition *statique* de services.

La composition *dynamique* a lieu au moment de l'exécution et permet de créer de manière autonome des services complexes en combinant des composants à la volée en fonction des demandes des utilisateurs et du contexte [Hassen et al., 2008]. La composition dynamique de services web permet de prendre en compte les services disponibles, leurs fonctionnalités et le but à accomplir que ce soit avant ou pendant l'exécution des services web.

La composition *statique* est définie via un processus métier (workflow). Ce type de composition peut avoir lieu au moment de la conception d'une application et ne tient pas compte des spécificités de la requête initiale. Elle regroupe les techniques d'*orchestration* et les techniques de *chorégraphie* [Hu, 2003]. Selon [Benatallah et al., 2005], l'*orchestration* et la chorégraphie de services constituent des moyens différents pour concevoir la composition. Elles se basent sur les workflows et se différencient par leur point de vue concernant la coordination des services intervenant dans une composition de services web. Dans [Lopez-Velasco et al., 2006], l'*orchestration* de services web exige de définir l'enchaînement des services web selon un canevas prédéfini, et de les exécuter selon un script d'*orchestration*. Le canevas et le script permettent de décrire les interactions entre les services web en identifiant les messages, et en spécifiant la logique et les séquences d'invocation. La *chorégraphie* est dédiée aux processus complexes ayant plusieurs parties qui interagissent et aux systèmes basés sur les évènements et sur les agents. Elle consiste à concevoir une coordination décentralisée d'un ensemble de services web afin d'accomplir un but commun [Benatallah et al., 2005]. Les interactions entre les différents services et les règles de participation doivent être décrites et déterminées. Chaque service web participant dans la chorégraphie doit savoir exactement quand être actif et avec qui interagir.

Contrairement à la composition de type orchestration, les services invoqués ne savent pas qu'ils font partie d'un processus métier. Seul le processus central est conscient de l'objectif général à accomplir. La chorégraphie est aussi appelée composition dynamique [Peltz, 2003] car elle n'est pas régie de façon statique comme dans une orchestration.

Un ensemble de langages a été proposé dans le but de décrire et de modéliser la composition des services web par un processus métier. Dans le cas d'une composition de type orchestration, nous pouvons citer à titre d'exemple *BPEL* (Business Process Execution Language) ou *BPEL4WS* (BPEL for Web Services) [Andrews et al., 2003]. Parmi les langages de composition dédiés à la chorégraphie, nous donnons l'exemple de *WS-CDL* (Web Service Choreography Description Language) [Kavantzas et al., 2004] et *OWL-S* (Semantic Markup for Web Services) [Burstein et al., 2004].

Dans la suite nous décrivons d'autres approches qui ont été proposées dans la littérature traitant la composition dynamique de services web.

Dans [Oh et al., 2008], les auteurs proposent une méthode basée sur les techniques de planification permettant la composition automatique de services web. Plusieurs types de réseaux ont été construits pour représenter des services caractérisés par une description syntaxique. Ils ont développé un générateur de descriptions synthétiques WDSL appelé WSBen (Web Service Discovery and Composition Benchmark) [Oh & Lee, 2009] basé sur la topologie des réseaux pour tester les performances des algorithmes de découverte et de composition de services web. Dans le même contexte, Cherifi et al. [Cherifi et al., 2013a] proposent un outil appelé Web Services Network EXTractor (WSNEXT) [Cherifi et al., 2011] permettant l'extraction d'un ensemble de réseaux pour faire la composition de services web. Contrairement au travail présenté dans [Oh et al., 2008], Cherifi et al. considèrent des réseaux orientés pour la composition et abordent l'aspect sémantique des descriptions et les éventuelles relations entre les représentations syntaxiques et sémantiques des services web.

Les travaux présentés dans [Wu & Khoury, 2012] proposent un algorithme de recherche basé sur un arbre pour la composition de services web dans une plate-forme de cloud computing. Les auteurs créent d'abord un arbre qui représente toutes les compositions possibles répondant aux exigences de l'utilisateur. Ils éliminent ensuite les branches irrégulières pour réduire le temps de réponse et améliorer les performances. Enfin, ils utilisent un algorithme heuristique pour rechercher, évaluer et classer les compositions optimales. Le travail reporté dans [Cheng et al., 2015], propose aussi une méthode de composition automatique de services basée sur les réseaux de Petri. Les exigences d'entrée/sortie de l'utilisateur sont modélisées selon les règles logiques de type Horn. Un problème de recherche de composition de services se transforme en un problème de Horn. Ensuite, la technique FPPN (Fuzzy Predicate Petri Net) est appliquée pour modéliser l'ensemble des clauses de Horn, et la technique de T-invariant est utilisée pour déterminer les services composables satisfaisant les exigences de l'utilisateur. L'inconvénient de cette

méthode et de celle de [Wu & Khoury, 2012] est que le processus d'optimisation ne peut pas être exécuté avant de recevoir les exigences de l'utilisateur.

Dans le cadre de la composition de services web, nous proposons dans le chapitre 5 une méthode permettant d'identifier un ensemble de compositions optimales possibles qui peuvent exister entre les services web découverts et/ou avec d'autres services. Nous exploitons le réseau d'interaction de services web construit dans le chapitre 4 (voir section 4.4.4, page 82) pour identifier les compositions possibles des services web qui sont susceptibles d'être composables (voir section 5.5, chapitre 5). Nous proposons également une méthode pour l'optimisation et la diversification des liens de compositions identifiés (voir section 5.5.3, chapitre 5).

3.4.3 Détection de communautés de services

Dans cette section, nous introduisons tout d'abord la notion de communauté. Ensuite, nous exposons différentes approches et méthodes qui ont été proposées pour la détection de communautés de services web.

L'étude des réseaux complexes, aussi appelés graphes de terrain, a montré qu'ils partageaient un certain nombre de propriétés topologiques non-triviales qui caractérisent la connectivité d'un réseau. La plupart des réseaux complexes exhibent une structure communautaire. Il s'agit de groupes de sommets très densément connectés à l'intérieur mais avec peu de liens vers l'extérieur. De nombreux travaux portant sur la définition des communautés ont été menés. Une communauté est donc une partition ou un sous-graphe composé de sommets densément reliés entre eux et faiblement liés aux autres sommets du graphe [Newman, 2004b, Schaeffer, 2007, Fortunato, 2010]. Ce type de définition ne vaut que pour les graphes statiques tandis que la plupart des graphes de terrain sont dynamiques. Ainsi, les données utilisées sont souvent dynamiques ; sur le web par exemple, les données peuvent être modifiées ou supprimées constamment. Une grande quantité d'information peut donc être ignorée. Pour intégrer la dynamique dans la détection de communautés, plusieurs travaux ont été proposés dans ce contexte [Palla et al., 2007].

La détection de communautés dans un réseau a fait l'objet de nombreuses recherches ayant abouti à plusieurs méthodes et algorithmes. Dans le contexte de services web, la notion de communauté est aussi intéressante. Une communauté peut regrouper un ensemble de services web ayant un même domaine d'intérêt ou offrant des fonctionnalités communes [Medjahed & Bouguettaya, 2005, Maamar et al., 2009]. Un noeud, dans un réseau de services web, peut représenter par exemple un service web ayant généralement une description textuelle. De nombreux travaux ont porté sur la découverte de communautés dans le cadre des services web.

Dans [Bentahar et al., 2007, Bentahar et al., 2008], les auteurs proposent un framework permettant de regrouper les services ayant des fonctionnalités similaires sous forme de communautés en utilisant des agents argumentatifs. Ils associent à chaque service

web un agent argumentatif. Les agents argumentatifs ont pour but de faciliter l'interaction entre les services et de résoudre les conflits qui peuvent exister. Les services web deviennent ainsi plus autonomes et capables de supporter une gestion automatique des communautés de services. Les auteurs de [Liu et al., 2009] utilisent les descriptions des documents WSDL pour construire des communautés de services web homogènes où, une communauté contient un ensemble de services offrant soit des opérations similaires ou des opérations potentiellement composables, par rapport à une requête donnée. La similitude entre les opérations de services web est calculée comme étant la similitude combinée de la description du texte et les entrées/sorties. Un graphe a été aussi construit pour représenter les opérations similaires et composables de services web par rapport à la requête de l'utilisateur. Une opération est représentée par un noeud dans le graphe et les compositions possibles sont représentées par des arêtes orientées. Certaines approches se basent sur les techniques de clustering pour contruire les communautés de services web. L'algorithme de clustering proposé dans [Liu & Wong, 2008] se base sur quatre types de caractéristiques pour déterminer la similitude entre les services à savoir le contenu, le contexte, le nom et la localisation du service. Un mécanisme de pondération est utilisé pour combiner ces caractéristiques et calculer la mesure de connexité entre les services. Le travail présenté dans [Yu & Rege, 2010] propose également un algorithme de clustering similaire permettant de regrouper les services similaires sous forme de communautés homogènes pour faciliter le processus de découverte de services web. Ainsi, la recherche d'un service peut s'effectuer directement dans la communauté concernée. L'algorithme proposé se base sur un schéma de co-clustering en exploitant les relations entre les services et les opérations.

Dans [Yu, 2011], Yu et al. proposent un nouveau framework pour découvrir automatiquement les communautés de services qui regroupent les services similaires. Le framework proposé applique la méthode de factorisation par matrices non négatives (Non-negative Matrix Factorization NMF) sur le corpus WSDL pour découvrir les communautés de services. De cette manière, les auteurs utilisent non seulement les descriptions de services, mais exploitent aussi les relations existantes entre les services et les opérations pour améliorer les performances de découverte des communautés de services. Un autre modèle a été proposé dans [Verma & Bharadwaj, 2015]. Il permet de détecter les communautés dans les réseaux sociaux hétérogènes en utilisant la méthode de factorisation par matrices non négatives NMF et un ensemble d'algorithmes de clustering.

Contrairement aux approches présentées ci-dessus, nous proposons dans le cadre de cette thèse, une nouvelle méthode permettant d'évaluer la qualité et la cohérence sémantique des communautés détectées (voir chapitre 7). Nous utilisons des mesures classiques de la classification, à savoir, la pureté et l'entropie [Zhao & Karypis, 2001] pour évaluer la qualité des communautés détectées. Ensuite, pour évaluer la cohérence sémantique d'une communauté, nous introduisons une nouvelle mesure appelée la divergence

sémantique des communautés détectées basée sur la divergence de *Kullback Leibler (KL)* [Steyvers & Griffiths, 2007] et les modèles thématiques [Blei & Lafferty, 2007]. Dans nos expérimentations, nous avons utilisé plusieurs algorithmes de détection de communautés pour valider notre méthode (voir section 7.3, chapitre 7).

3.5 Algorithmes d'extraction de motifs et de construction de treillis de concepts

L'extraction de motifs permet de répondre à des usages trés divers. Les motifs obtenus peuvent soit être interprétés de maniére brute (motif local), soit être combinés les uns avec les autres (motif global) ou encore être exploités pour créer un modéle prédictif ou descriptif. Dans cette section, nous présentons quelques algorithmes d'extraction de motifs fréquents, motifs fermés fréquents et motifs fréquents maximaux.

Les algorithmes d'extraction de motifs fréquents les plus utilisés sont APRIORI [Agrawal & Srikant, 1994], *ECLAT* [Zaki et al., 1997] et FP-Growth [Han et al., 2000]. L'algorithme APRIORI [Agrawal & Srikant, 1994] est certainement le plus connu et consiste à parcourir en largeur l'espace de recherche en commençant par les attributs singletons puis en visitant les paires d'attributs, puis les triplets, etc. Cet algorithme est une instance de l'algorithme générique GUESS & CORRECT [Mannila & Toivonen, 1997]. Il commence par rechercher les items fréquents. Puis, pour chaque itération k, il génère un ensemble de motifs *candidats* de taille k. Ces motifs candidats sont des motifs potentiellement fréquents. APRIORI élague l'espace de recherche en tirant partie de la propriété d'élagage 2.22 (voir chapitre 2, page 41). L'algorithme s'arrête lorsque l'ensemble des motifs candidats est vide. *ECLAT* [Zaki et al., 1997] est le premier algorithme à utiliser une stratégie de parcours en profondeur pour calculer des motifs fréquents. Il démarre avec la liste des éléments fréquents puis l'algorithme est réitéré en rajoutant les ensembles fréquents à l'ensemble des candidats jusqu'à ce que cet ensemble soit vide. L'algorithme *FP-Growth* (Frequent-Pattern Growth) [Han et al., 2000] fait également un parcours en profondeur et est unanimement considéré comme plus efficace par rapport aux autres algorithmes presque tous basés sur APRIORI. *FP-Growth* n'effectue pas de phases de génération de motifs candidats, suivies de phases de vérification du support. Il génère directement les motifs et leurs supports à partir des transactions. L'algorithme utilise une structure de données compacte appelée $FP-tree$ (*Frequent Pattern tree*). Cette structure est composée d'un arbre préfixe et d'une liste transversale de pointeurs faisant le lien entre les items et les transactions. Les items sont contenus dans la liste et dans l'arbre sont les items fréquents ordonnés par fréquence croissante, ce qui permet d'avoir une représentation des données compacte en mémoire.

De nombreux algorithmes ont été proposés, avec divers langages, pour la découverte et le calcul des motifs fermés fréquents comme par exemple CLOSE [Pasquier et al., 1999],

CHARM [Zaki & Hsiao, 2002] et FP-CLOSE [Grahne & Zhu, 2005]. CLOSE a été proposé par Pasquier et al. [Pasquier et al., 1999], et permet l'extraction de motifs fermés (et leurs supports) de la base de données en réalisant un parcours par niveaux. Tout motif non fermé est inclus dans le même ensemble d'objets et possède donc le même support que sa fermeture (le plus petit motif fermé qui le contient). Tous les motifs fréquents et leur support peuvent donc être déduits des motifs fermés fréquents avec leur support, sans accéder à la base de données. L'algorithme CHARM proposé par Zaki et al. [Zaki & Hsiao, 2002], procède par une approche en profondeur pour découvrir tous les motifs fermés fréquents. Il explore simultanement l'espace des motifs et l'espace des transactions. CHARM exploite la maximalité d'un motif fermé, i.e. un motif fermé couplé avec l'ensemble des objets le vérifiant n'est pas inclus dans aucun autre motif fermé. Il utilise également une représentation verticale, appelée $diffsets$ pour accélérer le calcul des supports. FP-CLOSE est basé sur l'algorithme *FP-Growth* et présenté par Grahne et Zhu en 2003 [Grahne & Zhu, 2005]. Pour trouver tous les motifs fermés fréquents, FP-CLOSE utilise une structure de données appelée $CFI-tree$ *(Close Frequent Pattern tree)*, qui est une variation d'un $FP-tree$. Un nouveau motif fréquent Y qui contient X a seulement besoin d'être comparé avec le $CFI-tree$ de X. Si dans le $CFI-tree$ de X, il n'y a pas de sur-ensemble de Y avec le même support que Y, alors Y est fermé.

Pour extraire les motifs fréquents maximaux dont tous les sur-ensembles sont non fréquents et tous les sous-ensembles sont fréquents, plusieurs algorithmes ont été proposés dans ce contexte. Nous citons à titre d'exemple les algorithmes de recherche en profondeur comme Mafia [Burdick et al., 2001], FPMax* [Grahne & Zhu, 2003] et Genmax [Gouda & Zaki, 2005].

Mafia [Burdick et al., 2001] utilise une représentation verticale pour la base de données de transactions. Il compose deux types de structure de données : un *bitmap* et une liste d'identifiant de transactions, pour chaque item. Dans le *bitmap*, il y a un *bit* pour chaque transaction de la base de données. Si l'item i apparait dans la *t-ième* transaction, alors le bit t du *bitmap* de l'item i est mis à un ; sinon le bit reste à zéro. Cette structure de données permet d'avoir une procédure de comptage du support des motifs très efficace. Toutefois, la structure peut être creuse pour certains types de données. Pour enlever les motifs fréquents qui ne sont pas maximaux, *Mafia* utilise trois stratégies d'élagage : PEP, $FHUT$ et $HUTMFI$. L'algorithme FPMax* [Grahne & Zhu, 2003] a été proposé par Grahne et Zhu. *FPMax** est basé sur l'algorithme *FP-Growth* et comprend plusieurs stratégies pour calculer efficacement les motifs fréquents maximaux tout en élaguant l'espace de recherche. Cet algorithme utilise aussi une variante de la structure $FP-tree$, appelée $MFI-tree$, pour les tests de sous-ensemble, et donne un certain nombre d'optimisations qui réduisent encore le temps de calcul. L'algorithme Genmax est présenté par Gouda et Zaki dans [Gouda & Zaki, 2005]. Il utilise une stratégie de parcours en profondeur pour énumérer tous les motifs fréquents maximaux. Cet algorithme utilise plusieurs

optimisations pour élaguer rapidement une grande partie de l'espace de recherche de sous-ensemble. Il utilise une technique appelée *progressive concentration* pour vérifier la maximalité et *la propagation de diffset* pour effectuer un calcul rapide du support.

Divers algorithmes ont été également proposés pour la construction de treillis de concepts dont les plus connus sont Chein [Chein, 1969], Ganter [Ganter, 1984], Bordat [Bordat, 1986], Carpineto [Carpineto & Romano, 1996], Nourine [Nourine & Raynaud, 1999], etc. Chacun de ces algorithmes se distingue des autres par plusieurs critères dont la stratégie de calcul des concepts, la recherche de l'ordre entre ces concepts, les structures de données utilisées pour le stockage des résultats intermédiaires et le résultat final. Certains algorithmes permettent aussi la visualisation de ces treillis. Chein [Chein, 1969], Ganter[Ganter, 1984] et Bordat [Bordat, 1986] font partie de la première génération des algorithmes de construction de treillis et sont appelés algorithmes batch. Ils prennent en entrée le contexte formel tout entier et calculent les concepts formels et l'ordre entre ces concepts de manière simultanée ou séquentielle. L'algorithme Bordat par exemple construit les concepts en s'appuyant sur une structure d'arbre pour garder les résultats intermédiaires et Chein génère les concepts par niveaux. Carpineto[Carpineto & Romano, 1996] et Nourine[Nourine & Raynaud, 1999] sont des algorithmes incrémentaux qui considèrent le contexte formel ligne par ligne (ou colonne par colonne) et construisent le treillis de concepts par ajouts successifs de ligne ou de colonne tout en conservant sa structure.

Pour le calcul de treillis de concepts fréquents, nous pouvons citer à titre d'exemple les algorithmes TITANIC [Delugach & Stumme, 2001] et CHARM-L [Zaki & Hsiao, 2005]. L'algorithme TITANIC [Delugach & Stumme, 2001] utilise la notion de fréquence pour calculer les concepts sans faire d'intersection entre les ensembles d'attributs. CHARM-L [Zaki & Hsiao, 2005] est un algorithme efficace qui permet de construire le treillis de concepts fréquents en spécifiant un seuil minimum.

3.6 Conclusion

Dans ce chapitre, nous avons présenté un état de l'art des travaux relatifs à la description des services web, la topologie des réseaux, le processus de découverte, la composition et la recommandation, la détection de communautés, l'extraction de motifs et la construction de treillis de concepts.

Dans un premier temps, nous avons introduit les différents modèles de description syntaxique et sémantique de services web. Ensuite, nous avons dressé un ensemble de travaux qui ont été proposés dans la littérature visant à faciliter le processus de découverte de services web. Nous avons décrit les trois catégories de ces approches, à savoir, les approches logiques, les approches non-logiques et les approches hybrides. Dans le cadre de cette thèse, nous proposons une approche de découverte qualifiée non-logique permettant

de diversifier les résultats de la recherche et d'identifier les services qui sont susceptibles d'être composable. Nous avons également souligné que toutes les approches proposées dans cette thèse peuvent être appliquées à n'importe quel modèle de description de services. Néanmoins, les services doivent être décrits par des documents WSDL et/ou SAWSDL pour pouvoir identifier les compositions possibles de ces services (voir chapitre 5). Puis, nous avons décrit les systèmes de recommandation et les approches proposées dans ce contexte. Nous proposons dans cette thèse, un système de recommandation hybride. Le système proposé exploite les avantages des approches basées à la fois sur le filtrage collaboratif et le contenu. L'originalité de notre système vient de la combinaison des modèles thématiques probabilistes et des motifs fréquents (voir chapitre 6).

Ensuite, nous avons présenté diverses approches basées sur les réseaux permettant de représenter les services web et leurs interactions. Nous proposons dans ce contexte, une méthode pour représenter et construire une structure de services web sous forme de réseaux (i.e. réseaux d'interaction et réseaux de similitude) (voir chapitre 4). Nous avons également étudié un ensemble de travaux antérieurs dans le cadre de la composition de services web. Dans notre travail, nous exploitons le réseau d'interaction pour identifier les liens de compositions entre les services web qui sont susceptible d'être composable (voir chapitre 5, section 5.5). Nous avons également décrit plusieurs approches dans le cadre de la détection de communautés de services web. Dans cette thèse, nous proposons une nouvelle méthode permettant d'évaluer la qualité, la cohérence sémantique des communautés détectées et classer les algorithmes de détection de communautés (voir chapitre 7). Enfin, nous avons évoqué plusieurs algorithmes d'extraction de motifs et de construction de treillis de concepts utiles pour la génération de réseaux d'interaction (voir chapitre 4) et pour le système de recommandation proposé (voir chapitre 6).

Dans le chapitre suivant, nous proposons deux modèles de réseaux pour représenter les services web, un réseau d'interaction et un réseau de similitude (voir chapitre 4). Les réseaux d'interaction sont utilisés pour identifier les liens de compositions entre les services web (voir chapitre 5, section 5.5). Les réseaux de similitude sont utilisés pour étudier la structure communautaire de ces réseaux (voir chapitre 7).

4

Réseaux de services web

Sommaire

4.1 Introduction

Les systèmes complexes sont constitués de nombreux éléments en interaction et dont les caractéristiques globales ne peuvent se réduire à celles de leurs composants. Les réseaux complexes sont utilisés dans de nombreux contextes pour modéliser les

interactions complexes de ces systèmes. L'analyse des réseaux complexes a été rendue possible grâce à la grande disponibilité de données massives permettant d'étudier la topologie des réseaux complexes. Un réseau complexe, aussi appelé graphe de terrain, désigne un grand graphe permettant de modéliser un système réel. Ce type de graphes, malgré leurs origines variées, ont des caractéristiques topologiques communes. Dans ce chapitre nous présentons deux modéles de réseaux de services web, un modèle d'interaction et un modèle de similitude [Naim et al., 2016a].

Le modèle de réseau de similitude est conçu sur la base de la similitude entre les services web et est lié généralement à la classification des services web. Nous définissons un réseau de similitude comme un graphe dont les nœuds représentent les services web et les liens indiquent le degré de similitude. Dans notre approche, nous proposons deux types de réseaux de similitude de services web qui se basent respectivement sur les descriptions syntaxiques et sémantiques des services web. Afin d'identifier la similarité entre deux services, nous proposons d'utiliser une mesure de proximité qui utilise le cosinus de l'angle entre deux vecteurs décrivant ces services (voir section 4.3.3). La similarité entre les services est calculée en se basant sur les représentations vectorielles (basées sur les vecteurs de compte TF-IDF) dans le cas de réseau de similitude syntaxique. Dans le cas de réseau de similitude sémantique, la distribution de probabilités sur les thèmes est utilisée comme critère de base pour calculer la similarité sémantique entre les services.

Un réseau d'interaction de services est défini comme un graphe orienté dans lequel les nœuds représentent l'ensemble des services et les liens représentent les interactions et les invocations entre deux opérations quelconques de chacun de ces services. Autrement dit, dans un réseau d'interaction de services, les liens matérialisent un flux d'information entre les opérations des différents services considérés. Un lien entre deux services représente alors la possibilité de les composer. Dans notre approche, nous considérons trois niveaux de granularité (i.e., paramètres, opérations et services) dans le cas du réseau d'interaction. Nous construisons dans un premier temps le réseau d'interaction d'opérations en se basant sur le formalisme de l'analyse de concepts formels (ACF) (voir chapitre 2, section 2.4, page 31) pour déduire ensuite le réseau d'interaction de services. Plus précisément, nous organisons un ensemble d'opérations de services web et leurs paramètres d'entrée/sortie sous forme de treillis de concepts formels. Les réseaux d'interaction d'opérations se distinguent en fonction du mode d'invocation (totale ou partielle).

Le reste de ce chapitre est organisé comme suit. Dans la section 4.2, nous présentons les concepts de la topologie des réseaux ainsi que leurs domaines d'application. Nous décrivons ensuite dans la section 4.3, les deux types de réseaux de similitude proposés. La section 4.4 décrit plus en détail les trois modèles de réseaux d'interaction basés sur les trois niveaux de granularité (i.e., paramètres, opérations et services). Enfin, les expérimentations et les évaluations réalisées dans ce chapitre sont décrites dans la section 4.5 avant de conclure dans la section 4.6.

4.2 Topologie des réseaux et propriétés fondamentales

Dans cette section, nous introduisons les concepts de la théorie des graphes permettant d'étudier et d'analyser les réseaux complexes.

4.2.1 Définitions et notions de base

Un graphe permet de modéliser de nombreux problèmes, de phénomènes et toutes formes de relations entre les objets d'une collection. Géométriquement, les objets sont représentés par des points appelés *sommets* ou *nœuds*. Ils sont reliés entre eux par des flèches appelées *arcs* dans un graphe orienté et *arêtes* dans un graphe non-orienté.

Les sommets sont les objets, au sens général du terme, qui sont en relation dans le graphe. Deux sommets v et v' sont *voisins* ou *adjacents* s'ils sont les extrémités d'une même arête du graphe.

Les *arêtes* décrivent les relations entre les sommets du graphe. Une arête relie deux sommets (éventuellement non confondus) du graphe. Deux arêtes d'un graphe sont *adjacentes* si elles ont au moins un sommet en commun. Une arête est *incidente* à un sommet si le sommet constitue une (ou deux) de ses extrémités. Les arêtes peuvent être valuées, c'est-à-dire qu'une valeur leur est attribuée. Une valuation forte indique alors une relation de forte intensité. On dit que le graphe est valué. Nous donnons ci-après les définitions formelles correspondantes.

Définition 4.1 ***Un graphe*** *$G = (V, E)$ est un ensemble fini et non vide de sommets (ou nœuds) V et un ensemble fini, mais éventuellement vide, de liens (arcs ou arêtes) E.*

Définition 4.2 *Un graphe orienté $G = (V, E)$ est composé d'un ensemble V de sommets et d'un ensemble E de paires de sommets nommées arcs.*

Définition 4.3 *Un graphe non-orienté $G = (V, E)$ est composé d'un ensemble V de sommets et d'un ensemble E de paires de sommets nommées arêtes.*

Définition 4.4 *Un graphe valué orienté (ou non orienté) $G = (V, E, F)$ est composé d'un ensemble V de sommets, un ensemble E d'arcs (ou arêtes) et F une fonction de coût.*

Nous pouvons associer à un graphe donné, une matrice carrée appelée également *Matrice d'adjacence* M. La relation entre les différents sommets du graphe est définie comme suit :

$$M(i,j) = \begin{cases} 1 & \text{Si } (i,j) \in E \\ 0 & \text{Sinon} \end{cases} \tag{4.1}$$

Où i et j représentent deux sommets dans le graphe G.

La matrice d'adjacence M est symétrique dans le cas des graphes non orientés. Un graphe peut être également représenté par une autre matrice dite *Laplacienne* définie par $M_L = M_D - M$ où M_D représente la matrice diagonale dont les éléments correspondent au degré d'un sommet. Comme nous allons voir dans ce qui suit, le degré d'un sommet est défini par le nombre d'arêtes incidentes à ce sommet.

La définition 4.5 introduit la notion de voisinage d'ordre l dans un graphe.

Définition 4.5 ***Voisinage d'ordre*** *l : soit $G = (V, E)$ un graphe orienté composé d'un ensemble de sommets V et d'un ensemble d'arcs E. Deux sommets i et j sont voisins ou voisins d'ordre $l = 1, 2, \ldots$ si et seulement si le plus court chemin entre eux a une longueur l. Nous notons ce lien par $(i, j) \in v(l)$.*

4.2.2 Propriétés topologiques fondamentales

Nous noterons n le nombre de sommets ($n = |V|$) (ou taille du graphe) et m le nombre d'arêtes ou d'arcs ($m = |E|$) dans un graphe G.

- Degré : dans un graphe non orienté, le **degré** d'un sommet est le nombre d'arêtes incidentes à ce sommet. Pour un graphe orienté, on peut distinguer le **degré entrant**, le **degré sortant** et le **degré total** d'un sommet. Le degré entrant est le nombre d'arcs incidents. Le degré sortant correspond au nombre d'arcs émanant du sommet considéré. Le degré total est la somme des degrés entrants et sortants. Nous notons k_i le degré d'un sommet i. Le **degré moyen** est la valeur moyenne des degrés de l'ensemble du graphe. Il peut être calculé par la relation suivante :

$$Deg_{moy} = \frac{1}{n} \sum_{i=1}^{n} k_i \tag{4.2}$$

- Distance : dans un graphe non orienté, une **chaîne** (directed walk) est une suite d'arêtes distinctes reliant deux sommets. La notion correspondante dans les graphes orientés est celle de **chemin**. Une **chaîne élémentaire** est une chaîne ne rencontrant pas deux fois le même sommet, c'est-à-dire dont les sommets sont distincts. Une **chaîne simple** est une chaîne n'utilisant pas deux fois la même arête. La **longueur** d'une chaîne est le nombre d'arêtes qui la composent.

- La **distance** d_{ij} entre deux sommets est la plus courte longueur des chaînes qui la relient.
- Diamètre : le **diamètre D** d'un graphe est la plus grande distance entre deux sommets.

$$D = max(d_{ij}) \tag{4.3}$$

- La **distance moyenne** est la moyenne de toutes les distances entre deux sommets quelconques du graphe. Elle est dite longueur caractéristique **L** et définie par la

formule suivante :

$$L = \frac{1}{n(n-1)} \sum_{i \neq j} d_{ij} \tag{4.4}$$

- **Transitivité :** la transitivité est aussi appelée fraction de triangles dans un graphe ou coefficient de clustering. Elle correspond à la densité de triangles d'un graphe, un triangle étant une structure de trois sommets complètement connectés. Sa formule est la suivante :

$$Transitivity = \frac{3 * NbTriangle}{NbTriplets} \tag{4.5}$$

Où $NbTriangle$ représente le nombre de triangles dans le graphe et $NbTriplets$ représente le nombre de triplets connectés. Le facteur 3 au numérateur contrebalance le fait que chaque triangle contribue à trois triplets et contraint la valeur du coefficient à l'intervalle [0,1].

La valeur de la transitivité globale est une estimée de la probabilité moyenne que deux sommets qui sont voisins d'un troisième soient eux-mêmes voisins. Cette valeur donne une vue d'ensemble de la présence de triades dans un réseau. Notons qu'il existe également une définition locale du coefficient de transitivité.

- **Composante connexe :** un graphe est dit **connexe** si, pour tout couple de sommets, il existe une chaîne allant de l'un à l'autre. Une **composante connexe** d'un graphe non orienté G est un sous-graphe G' de G qui est connexe et maximal (c'est à dire qu'aucun autre sous-graphe connexe de G ne contient G'). Dans le cas d'un graphe orienté, on parle de **graphe fortement connexe** au lieu de **connexe** et de **composante fortement connexe** au lieu de **composante connexe**. Quand on parle de connexité pour un graphe orienté, on considère non pas ce graphe mais le graphe non-orienté correspondant.

- **Densité :** la **densité** est le rapport du nombre d'arêtes m dans le graphe sur le nombre maximum qui peut exister. Elle est définie par la formule suivante :

$$d = \frac{m}{n(n-1)} \tag{4.6}$$

- **Partitionnement :** la grande taille et la complexité naturelle des graphes de terrain constituent des obstacles à la compréhension de leur structure. Ainsi, la répartition de ces graphes en groupes, dans lesquels les sommets sont plus fortement connectés entre eux, facilite leur analyse et la compréhension de la structure de leur fonctionnement. Nous donnons ci-après quelques notions utilisées pour mieux analyser ces graphes.
  - Une **partition** est définie comme un sous-graphe connexe.

  - Une **clique** est un sous-graphe maximal complet (tous les couples de sommets sont reliés par une arête) comprenant au moins 3 sommets. Une **p-clique** désigne une clique contenant p sommets.
  - Une **communauté** ou « cluster » : est une division d'un graphe ou réseau en groupes à l'intérieur desquels la densité des relations est plus forte que la densité de connexions vers l'extérieur.

- Propriétés communes aux grands graphes de terrain : les grands graphes de terrain ou réseaux possèdent des caractéristiques structurelles communes et non triviales. Un graphe **petit monde** (small world en anglais) est caractérisé par une petite distance moyenne. En d'autres termes, la plupart des sommets du graphe peuvent être atteints de tous les autres par un petit nombre de sauts. Elle constitue l'une des caractéristiques communes des graphes de terrain : ils possèdent tous une faible distance moyenne. Dans certains contextes, le terme petit-monde implique aussi un fort coefficient de clustering. Dans le cas d'un graphe **sans échelle**, la distribution des degrés suit une loi de puissance. Cette propriété implique que dans les graphes de terrain, certain sommets ayant un degré très élevé jouent des rôles particuliers par rapport à d'autres sommets du graphe possédant un degré beaucoup plus faible.

4.2.3 Type de réseaux complexes et domaines d'application

De nombreux systèmes réels peuvent être modélisés sous forme réseaux complexes. Un réseau complexe est un grand graphe possédant des propriétés topologiques non triviales, une structure irrégulière, complexe et évoluant de façon dynamique dans le temps. Ce type de graphes est utilisé, dans des contextes divers et variés, afin de modéliser des interactions complexes tels que : les réseaux sociaux, les réseaux d'information, les réseaux technologiques ou encore les réseaux biologiques.

  - Les **réseaux sociaux** décrivent un ensemble de relations entre un ensemble d'acteurs. Les acteurs peuvent être des individus ou des entités sociales comme des associations, des entreprises, etc. Dans un réseau, les sommets peuvent alors représenter les acteurs et les liens représentent les relation. Nous pouvons citer l'exemple du réseau social des étudiants d'une université qui ont déjà été dans la même classe. Les acteurs seront alors tous les étudiants de l'université. La relation entre deux éléments sera alors "ont déjà été dans la même classe".
  - Les **réseaux d'information** sont une généralisation de la notion des réseaux sociaux permettant de définir des liens abstraits pour référencer des supports d'information. En d'autres termes, les réseaux sociaux représentent des entités et les relations qui existent entre elles, tandis que les réseaux d'information intègrent également des attributs pour décrire ces entités. Avec l'émergence du Web 2.0 et des réseaux numériques, les réseaux d'information sont utilisés pour prendre en

considération les caractéristiques décrivant les acteurs des réseaux sociaux et leurs relations. Différentes sources de données peuvent être modèlisées sous la forme de réseaux d'information. Nous pouvons citer par exemple le World Wide Web dont les nœuds représentent les pages Web et les liens sont des liens hypertextes.

– Les **réseaux technologiques**, aussi appelés réseaux d'infrastructure, désignent la distribution de ressources. Ils représentent des connections matérielles entre des objets distribués dans un espace géographique. De nombreux exemples peuvent être cités pour ce type de réseaux. Les réseaux de transport avec les liaisons aériennes ou les voies terrestres. Le réseau de l'Internet fait également partie de cette catégorie. Ce réseau est une capture de la structure de l'Internet au niveau des systèmes autonomes, reconstruit à partir des tables BGP (Border Gateway Protocol) de archive.routeviews.org. Ce snapshot a été créé par Mark Newman à partir des données du 22 juillet 2006. L'analyse d'un tel réseau, dont les nœuds représentent des routeurs et les liens des liaisons physiques entre les routeurs, peut être utile pour identifier par exemple les points de rupture.
– Les **réseaux biologiques** représentent les relations des systèmes vivants. De nombreux modèles ont été etudiés dans cette catégorie de réseaux. Citons par exemple, le réseau transcriptionnel, qui décrit la relation entre les gènes et les protéines, le réseau d'interaction protéine-protéine, qui tient compte des relations entre protéines, ou le réseau métabolique, qui cherche à modéliser les réactions métaboliques d'un organisme. Les réseaux de neurones et les réseaux alimentaires sont d'autres exemples d'origine biologique.

Dans le cadre de cette thèse, nous nous intéressons aux réseaux d'information pour étudier les interactions entre les services web. Nous construisons ainsi des réseaux de services dont les nœuds sont les services web et les liens représentent les interactions et les dépendances entre ces services. Nous utilisons les réseaux d'interaction pour identifier les liens de compositions entre les services web (voir chapitre 5, section 5.5). Nous étudions également, dans le chapitre 7, la structure communautaire de ces réseaux.

4.3 Modèles de réseaux de similitude de services web

4.3.1 Définition

Le modèle de réseau de similitude est conçu sur la base de la similitude entre les services web. Il se considère comme un support de la classification des services web selon le critère de la similitude. Nous définissons un réseau de similitude comme un graphe dont les nœuds représentent les services web et les liens indiquent le degré de similitude. Afin d'identifier la similarité entre deux services, nous utilisons une mesure de proximité, que nous décrivons par la suite dans la section 4.3.3. Cette mesure permet de déterminer les correspondances entre les déscriptions de services. Dans un résau de similitude, un

lien est créé entre deux services $S1$ et $S2$ si et seulement si le degré de similitude entre ces deux services est supérieur à la valeur d'un seuil donné.

4.3.2 Réseaux de similitude syntaxique et sémantique

Dans cette section nous proposons deux types de réseaux de similitude de services web. Ces deux modèles se basent respectivement sur les descriptions syntaxiques et sémantiques des services web.

Comme nous l'avons mentionné précédemment, la plupart des services web disponibles sur Internet sont décrits par des documents WSDL ou selon le paradigme REST. Dans notre travail, nous extrayons toutes les informations pertinentes décrivant les principales fonctionnalités qu'offrent les services web étudiés. Ces informations sont utilisées pour produire des représentations vectorielles basées sur les poids TF-IDF permettant de représenter les services web (voir section 2.3.1, page 23). La technique VSM et l'algorithme TF-IDF ont été utilisés pour représenter chaque service web sous forme d'un vecteur de comptes. En effet, ces représentations vectorielles sont centrées sur des descriptions textuelles syntaxiques et les différents éléments présents dans les documents de description de services. Dans notre approche, nous utilisons les représentations vectorielles basées sur les poids TF-IDF pour construire le réseau de similitude syntaxique des services web.

Pour capturer la sémantique des fonctionnalités offertes par les services web, nous utilisons dans cette thèse la notion de thème comme décrit précédemment dans le chapitre 2. Plus précisément, nous utilisons le modèle probabiliste thématique CTM [Blei & Lafferty, 2007], pour extraire un ensemble de thèmes à partir de descriptions de services web (section 2.3.3, page 29). Les thèmes sont utilisés comme des techniques efficaces de réduction des dimensions en capturant des relations sémantiques entre mots-thèmes et thèmes-services, exprimées sous forme de distributions de probabilités. Une fois le modèle probabiliste CTM formé, la distribution des mots pour chaque thème est connue et tous les services dans le corpus de données peuvent être décrits comme une distribution de thèmes (i.e. $\overline{s} = \{z_1, z_2, ..., z_K\}$) où chaque dimension z_k reflète la probabilité que le service s appartienne au thème k. Chaque thème extrait est associé à un groupe de concepts textuels (i.e. mots) et/ou des concepts sémantiques qui apparaissent dans les descriptions de services web. Les thèmes sont exprimés sous forme de distributions de probabilités sur les mots. Pour construire le réseau de similitude sémantique de services web, nous nous basons sur la distribution multinomiale sur les thèmes $\theta^{(s)}$ pour un service s, afin de déterminer et calculer la similarité sémantique entre les différents services considérés.

La figure 4.1 montre les différentes étapes permettant de construire les deux modèles de réseaux de similitude.

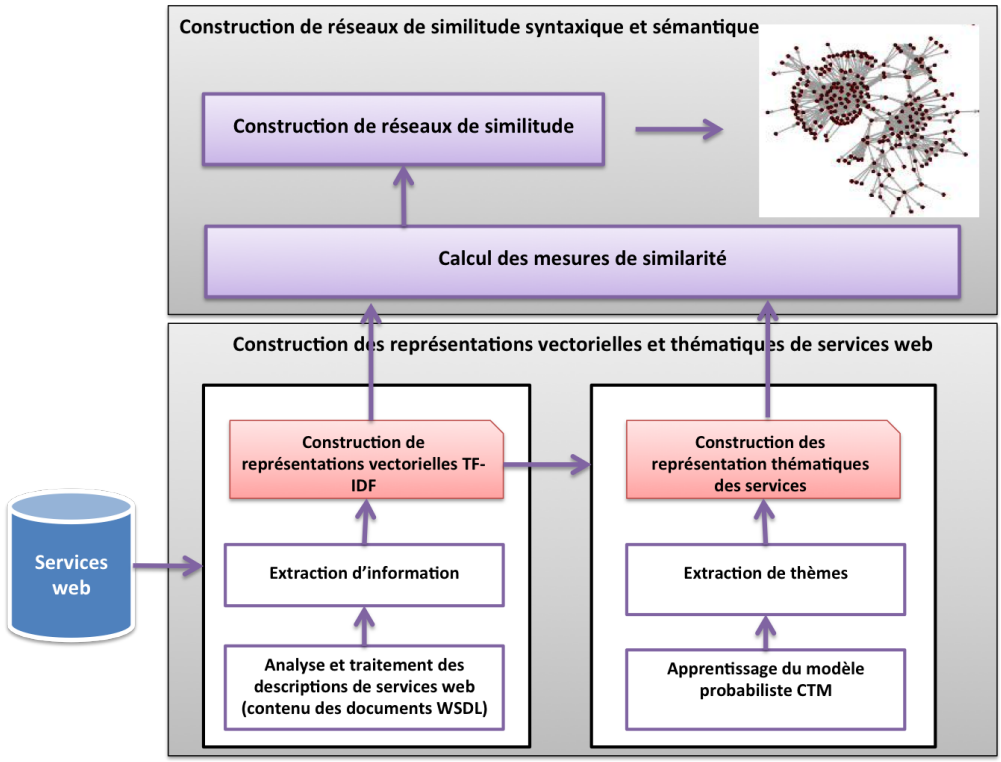


Figure 4.1 – Étapes de construction des réseaux de similitude de services web.

4.3.3 Mesure de similarité

Afin d'identifier la similarité entre deux services, nous proposons d'utiliser une mesure de proximité appelée *Multidimensional Angle* (ou *Similarité Cosinus*) ; une mesure qui utilise le cosinus de l'angle entre deux vecteurs. La similarité Cosinus entre un vecteur p et un vecteur q est calculée en utilisant l'équation 4.7.

$$Cosinus(p,q) = \frac{p.q}{\parallel p \parallel . \parallel q \parallel} = \frac{\sum_{i=1}^{t} p_i q_i}{\sqrt{\sum_{i=1}^{t} p_i^2 \sum_{i=1}^{t} q_i^2}} \tag{4.7}$$

Où $t = |p| = |q|$. Les valeurs de similarité sont dans l'intervalle [0,1] où 0 indique l'absence de similarité entre les vecteurs et 1 indique que les vecteurs sont identiques.

Dans notre approche, nous nous basons sur cette mesure pour construire les réseaux de similitude de services web. Comme décrit dans la section précédente, la similarité entre les services est calculée en se basant sur les représentations vectorielles (basée sur les vecteurs TF-IDF) dans le cas de réseau de similitude syntaxique. Dans le cas de réseau de similitude sémantique, la distribution de probabilités sur les thèmes $\theta^{(s)}$ est utilisée comme critère de base pour calculer la similarité sémantique entre les services. Dans un

réseau de similitude, un lien est créé entre deux services si et seulement si le degré de similitude entre les deux vecteurs décrivant ces services est supérieur à un seuil donné.

Notons également que nous pouvons utiliser d'autres mesures de similarité et/ou de distance pour calculer la similarité entre les services web. Dans [Gibbs & Su, 2002], les auteurs ont examiné plusieurs métriques/distances de probabilité très utilisées pour calculer la similarité et/ou la distance entre deux vecteurs. Certaines de ces mesures se basent sur le calcul de la distance entre les services.

4.4 Modèle de réseaux d'interaction de services web

Afin de modéliser les interactions et les dépendances entre un ensemble de services web sous forme de graphes ou réseaux, les nœuds peuvent être définis à partir de trois niveaux de granularité (paramètres, opérations et services). Nous décrivons dans cette section les trois modèles de réseaux d'interaction.

4.4.1 Réseau de dépendance de paramètres

Un ***réseau de dépendance de paramètres*** est un graphe orienté dont les nœuds sont les paramètres des services web. Les liens représentent les relations de dépendance entre les paramètres des opérations. Afin de construire un réseau d'interaction de paramètres, un lien est créé entre chacun des paramètres en entrée d'une opération et chacun de ses paramètres en sortie. Formellement, dans un réseau de dépendances des paramètres, une opération peut être définie comme un triplet (IN_i, OUT_i, D_i) où IN_i représente l'ensemble de paramètres d'entrées, OUT_i représente l'ensemble de paramètres de sorties et D_i représente l'ensemble des liens de dépendance. Supposons que nous avons quatres services web ($S1$, $S2$, $S3$, et $S4$) représentés dans la figure 4.2. Leurs opérations sont numérotées de 1 à 6. Les paramètres en entrée et sortie sont étiquetés de a à l. Les relations de dépendance entre les paramètres sont représentées dans le tableau 4.1. La figure 4.3 (partie droite) représente un réseau de paramètres traduisant les relations de dépendance entre les paramètres d'entrée et sortie. A titre d'exemple, considérons l'opération 1. Elle a a comme paramètre d'entrée et les paramètres de sortie sont b et c. Deux arcs sont respectivement créés de a vers b et de a vers c. Autrement dit, b et c dépendent tous les deux de a. La présence d'un lien, dans un réseau de dépendance de paramètre, signifie qu'il existe au moins une opération qui utilise l'extrémité initiale en tant que paramètres d'entrée et l'extrémité finale en tant que paramètre de sortie. Un paramètre est représenté par un seul nœud dans le réseau et peut être utilisé, soit comme entrée, soit comme sortie de plusieurs opérations. Par exemple, sur la figure 4.3 (partie gauche), les paramètres $\{c, d, e\}$ apparaissent plus d'une fois, soit comme entrée, soit comme sortie de plusieurs opérations. c est en sortie de 1 et en entrée de 3 et 5, d est en entrée de 4 et en sortie de 2 et e est en entrée de 6 et en sortie de 3.

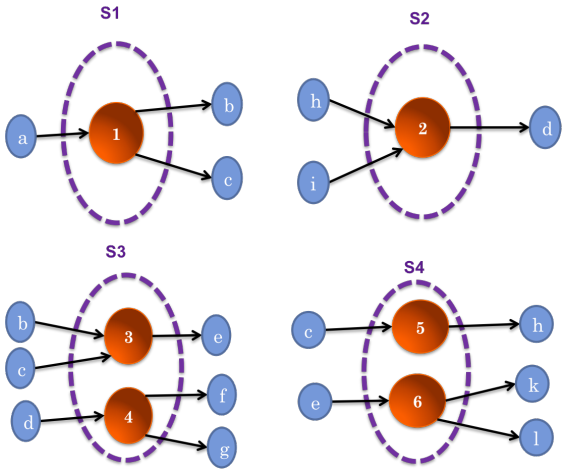


Figure 4.2 – Un exemple de quatres services web ainsi que leur opérations et paramètres d'entrée/sortie.

Opération	paramètres d'entrée	paramètres de sortie	liens de dépendance
1	{a}	{b, c}	{(a, b), (a, c)}
2	{h, i}	{d}	{(h, d), (i, d)}
3	{b, c}	{e}	{(b, e), (c, e)}
4	{d}	{f, g}	{(d, f), (d, g)}
5	{c}	{h}	{(c, h)}
6	{e}	{k, l}	{(e, k), (e, l)}

Table 4.1 – Paramètres des opérations et relations de dépendance associées.

4.4.2 Réseau d'interaction d'opérations

Un ***réseau d'interaction d'opérations*** est un graphe orienté dont les nœuds représentent l'ensemble des opérations et les liens traduisent les relations d'invocations et d'interactions entre les opérations. Une opération f_i peut être définie comme un couple (IN_{f_i}, OUT_{f_i}) où IN_{f_i} et OUT_{f_i} désignent respectivement l'ensemble de paramètres d'entrée et de sortie de f_i. Il existe deux modes d'invocation, à savoir l'invocation *partielle* et l'invocation *totale*, pour exprimer une relation d'interaction entre deux opérations (voir définition 4.6) :

Définition 4.6 (Invocation d'opérations) *Pour deux opérations $f_i = (IN_i, OUT_i)$ et $f_j = (IN_j, OUT_j)$:*

- *l'opération f_i est en invocation totale avec l'opération f_j, dénoté par $f_i \rightarrow f_j$, si et seulement si pour chaque paramètre d'entrée $p \in IN_j$ de l'opération f_j il existe un paramètre de sortie similaire $q \in OUT_i$ de l'opération f_i.*
- *l'opération f_i est en invocation partielle avec l'opération f_j, dénoté par $f_i \rightharpoonup f_j$,*

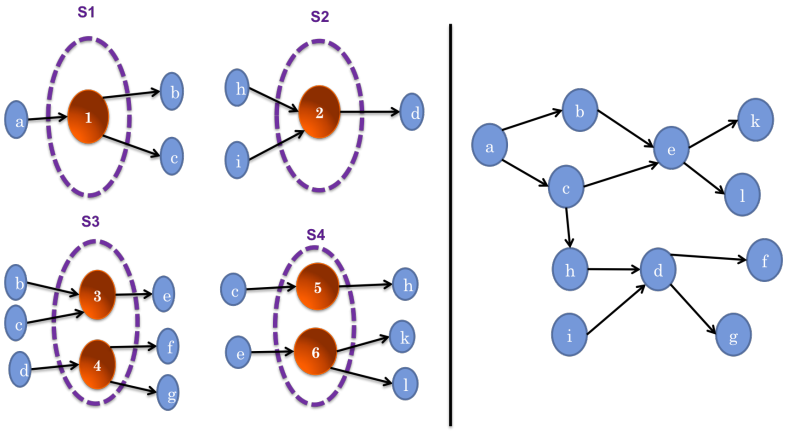


Figure 4.3 – Graphe de dépendance de paramètres (partie droite) à nœuds étiquetés de a à l obtenu à partir de 6 opérations numérotées de 1 à 6 (partie gauche).

s'il existe au moins un paramètre en sortie $q \in OUT_i$ de f_i similaire à un paramètre en entrée $p \in IN_j$ de f_j.

A titre d'exemple, considérons l'ensemble des services web de l'exemple précédent ainsi que l'ensemble de leur opérations (voir figure 4.4 (partie gauche)). La partie droite de la figure 4.4 correspond aux deux types de réseaux d'interaction d'opérations associés. Le réseau d'interaction d'opérations construit selon le mode d'invocation totale est représenté dans la partie supérieure. Le réseau de dépendance d'opérations construit selon le mode d'invocation partielle est représenté dans la partie inférieure.

Considérons le réseau en mode d'invocation totale, tous les paramètres d'entrée de l'opérations 3, c'est à dire b et c, sont inclus dans les sorties de l'opération 1 (c'est-à-dire b et c). Cela est traduit par la présence d'un lien de l'opération 1 vers l'opération 3. Notons dans ce cas, le nombre des entrées de l'opération 3 est le même que le nombre de sorties de l'opération1. Dans d'autres cas, il se peut qu'une opération cible ne puisse pas recevoir tous ses paramètres. Dans le graphe en mode d'invocation partielle, prenons l'opération 2 dont les paramètres d'entrée sont : $\{h, i\}$. Les paramètres de sortie de l'opération 5 sont $\{h\}$. L'opération 2 reçoit une partie de ses paramètres(c'est à dire le paramètre h). Un lien est créé de l'opération 5 vers l'opération 2.

4.4.3 Réseau d'interaction de services

Un ***réseau d'interaction de services web*** est défini comme un graphe orienté dans lequel les nœuds représentent l'ensemble des services et les liens représentent l'existence des opérations invocables entre les services. Autrement dit, dans un réseau d'interaction de services, les liens matérialisent un flux d'information entre les opérations des différents services considérés. Les réseaux de services web sont définis à partir des opérations et se

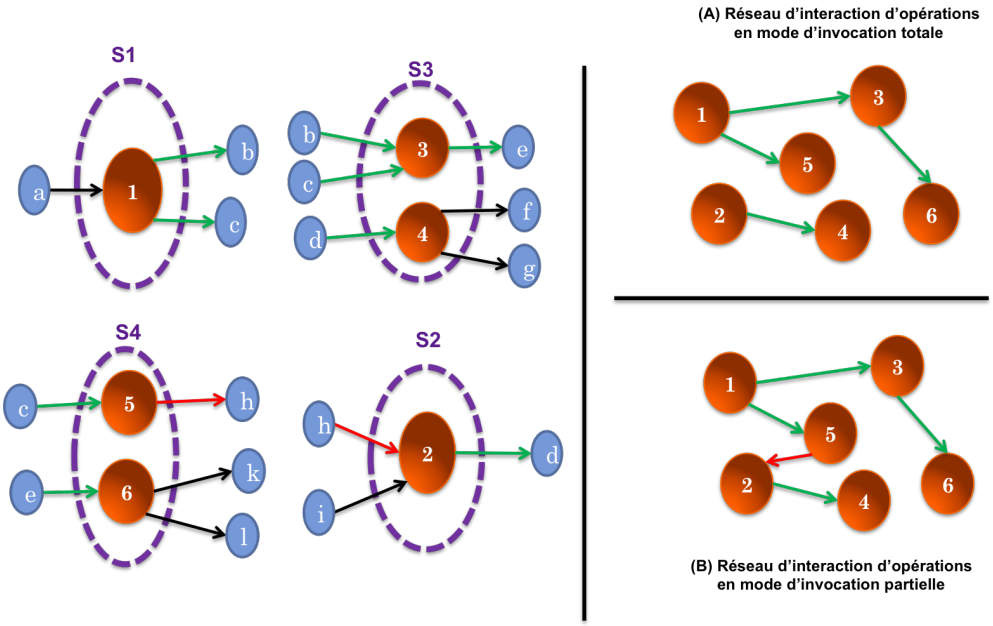


Figure 4.4 – Graphes d'interaction des opérations à invocation totale (A) et à invocation partielle (B) issus de 6 opérations numérotées de 1 à 6 de l'exemple 4.2

distinguent également en fonction du mode d'invocation (*totale* et *partielle*) :

- Dans le cas d'*invocation totale*, un lien est créé entre deux services s_i et s_j, s'il existe au moins une opération de s_i en invocation totale avec une opération de s_j.
- Dans le cas d'une *invocation partielle*, un lien est créé entre deux services s_i et s_j, s'il existe au moins une opération de s_i en invocation partielle avec une opération de s_j.

Formellement, un réseau d'interaction de services web est défini comme un graphe orienté $G = (S, V)$ où $S = \{s_1, s_2, \ldots, s_n\}$ est un ensemble de services web et V est un ensemble d'arêtes tel que :

- $V = \{(s_i, s_j) \in S \times S), \exists f_i \in s_i, \exists f_j \in s_j : f_i \rightarrow f_j\}$ dans le cas de l'invocation totale. $f_i \rightarrow f_j$ signifie que l'opération f_i est en invocation totale avec l'opération f_j.
- $V = \{(s_i, s_j) \in S \times S), \exists f_i \in s_i, \exists f_j \in s_j : f_i \rightharpoonup f_j\}$ dans le cas de l'invocation partielle. $f_i \rightharpoonup f_j$ signifie que l'opération f_i est en invocation partielle avec l'opération f_j.

  où $f_i \in s_i$ signifie que le service s_i offre l'opération f_i.

La figure 4.5 représente respectivement les réseaux d'interaction de services en mode d'invocation totale (A) et en mode d'invocation partielle (B) issus des 4 services libellés $S1$, $S2$, $S3$ et $S4$ de l'exemple précédent de la figure 4.2. Dans le réseau d'interaction en mode d'invocation totale, le service $S1$ invoque le service $S4$ au travers de l'opération 5

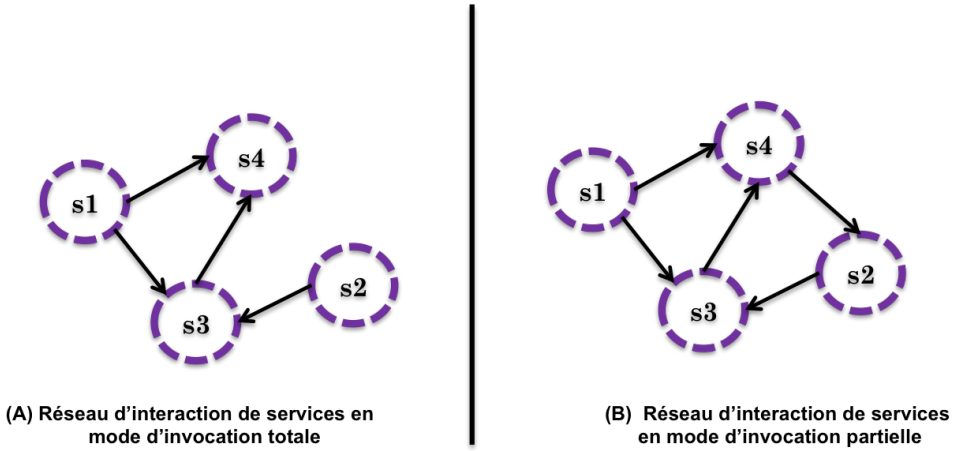


Figure 4.5 – Réseau d'interaction de services en mode d'invocation totale (A) et en mode d'invocation partielle (B) issus des 4 services libellés S1, S2, S3 et S4 de l'exemple 4.2

(voir le réseau d'interaction d'opérations en mode d'invocation totale représenté dans la figure 4.4 (A)). Le service $S1$ invoque également le service $S3$ au travers de l'opération 3. Dans le réseau d'interaction en mode d'invocation partielle, le service $S1$ invoque le service $S4$ qui à son tour invoque le service $S2$ au travers de l'opération 2 (voir le réseau d'interaction d'opérations en mode d'invocation partielle représenté dans la figure 4.4 (B)). Le réseau d'interaction en mode d'invocation partielle inclut le réseau d'interaction en mode d'invocation totale.

4.4.4 Méthode de construction de réseaux d'interaction de services web

Nous décrivons dans cette section, la méthode proposée pour construire le réseau d'interaction de services web. Afin de construire le réseau d'interaction de services, nous contruisons d'abord le réseau d'interaction d'opérations en se basant sur le formalisme de l'analyse de concepts formels (ACF) (voir chapitre 2, section 2.4). Dans notre approche, nous organisons dans un premier temps un ensemble d'opérations de services web et leurs paramètres d'entrée et de sorties sous forme de contexte formel. Un contexte formel est noté $\mathbf{K} = (O, P, I)$ où O est un ensemble d'objets (i.e. opérations), P est un ensemble d'attributs (i.e. paramètres d'entrées et de sorties), et I une relation binaire entre O et P ($I \subseteq O \times P$). Chaque couple $(o, p) \in I$ exprime que l'objet $o \in O$ contient l'attribut $p \in P$. Ensuite, nous calculons un ensemble de concepts formels organisé en treillis de concepts formels (voir section 2.4.3, page 36).

Nous décrivons ci-dessous les principales étapes permettant de construire le réseau d'interaction d'opérations :

– Construire les contextes binaires BC_{IN} et BC_{OUT} en se basant respectivement sur

les dépendances entre les opérations et leurs paramètres d'entrée et de sortie.
- Construire les treillis de concepts CL_{IN} et CL_{OUT} à partir de BC_{IN} et BC_{OUT} respectivement.
- Générer le réseau d'interaction d'opérations en exécutant une recherche dans les treillis de concepts CL_{IN} and CL_{OUT} selon le mode d'invocation totale ou partielle.

L'algorithme 1 présente la méthode de construction du réseau d'interaction d'opérations.

Réseau d'interaction de services web : En se basant sur le réseau d'interaction d'opérations, nous pouvons construire facilement le réseau d'interaction de services web comme suit :
- Dans le cas d'une invocation partielle, un lien est créé entre deux services s_1 et s_2, s'il existe au moins une opération de s_1 en invocation partielle avec une opération de s_2.
- Dans le cas d'une invocation totale, un lien est créé entre deux services s_1 et s_2, s'il existe au moins une opération de s_1 en invocation totale avec une opération de s_2.

Mise en correspondance entre les paramètres de services web

Comme nous l'avons déja décrit, un réseau d'interaction de services web est un ensemble de nœuds et de liens. Dans un premier temps, en tant que nœud, nous considérons trois niveaux de granularités (i.e., paramètres, opérations, services). Ensuite, pour les liens, nous utilisons la similitude entre les paramètres ou les invocations entre les opérations. La notion de similitude entre les paramètres est utilisée pour déterminer les relations de dépendance entre deux ensembles de paramètres d'entrée/sortie et pour déterminer les relations d'interaction et d'invocation entre les opérations. Dans les services web réels, il n'y a pas de consensus sur la manière de nommer les paramètres. Ainsi, les paramètres n'ayant pas les mêmes noms peuvent transmettre la même information. De la même manière, les paramètres dont les noms sont identiques peuvent véhiculer une information différente. Ainsi, pour évaluer la similitude entre les paramètres d'entrée/sortie, nous proposons une fonction permettant de comparer individuellement les paramètres et qui consiste à calculer la similitude entre deux chaînes de caractères. Notre fonction $paramMatch(p_1, p_2)$ determine si deux paramètres p_1 et p_2 sont similaires ou non. La façon la plus simple est de vérifier si ces deux paramètres ont le même nom et le même type : $(p_1.name = p_2.name)$ et $(p_1.type = p_2.type)$. Formellement, la fonction $paramMatch(p_1, p_2, f, \lambda)$ retourne vrai si : (1) $typeMatch(p_1.type, p_2.type) = vrai$, et (2) $nameMatch(p_1.name, p_2.name, f, \theta) = vrai$:
- **typeMatch** : une fonction $typeMatch(p_1.type, p_2.type)$ retourne vrai si : (1) $p_1.type = p_2.type$ ou (2) $p_1.type$ hérite de $p_2.type$ dans la hiérarchie des types.

Algorithme 1 Construction de réseau d'interaction d'opérations

Entrées :

- $O = \{o_1, \ldots, o_M\}$: un ensemble d'opérations et leur paramètre d'entrée/sortie.
- CL_{IN} : treillis de concepts représentant les opérations et leur paramètres d'entrée.
- CL_{OUT} : treillis de concepts représentant les opérations et leur paramètres de sortie.
- SearchType : type de recherche : Exact (cas d'invocation totale) ou Approximate (cas d'invocation partielle).

Sorties :

- Un ensemble d'arcs orientés *EdgeSet* construits.

1: $EdgeSet = \emptyset$
2: **Pour** chaque op $o_m \in O = \{o_1, \ldots, o_M\}$ **Faire**
3: $paramSet = GetInputParameters(op)$
4: **Si** $paramSet \neq \emptyset$ **Alors**
5: Recherche dans le treillis de concepts CL_{OUT}
6: $opSet = LatticeSearch(CL_{OUT}, SearchType, paramSet)$
7: Créer un arc orienté à partir de chaque opération de *opSet* vers l'opération sélectionnée *op*
8: Ajouter l'arc créé dans la liste *EdgeSet*.
9: **Sinon**
10: $paramSet = GetOutputParameters(op)$
11: Recherche dans le treillis de concepts CL_{IN}
12: $opSet = LatticeSearch(CL_{IN}, SearchType, paramSet)$
13: Créer un arc orienté à partir de l'opération sélectionnée *op* vers chaque opération trouvée dans *opSet*.
14: Ajouter l'arc créé dans la liste *EdgeSet*.
15: **Fin si**
16: **Fin pour**
17: **Retourner** Un ensemble d'arcs orientés *EdgeSet* construits.

Fonctions utilisées :

- $GetInputParameters(op)$: récupérer la liste des paramètres d'entrée de l'opération *op*.
- $GetOutputParameters(op)$: récupérer la liste des paramètres de sortie de l'opération *op*.
- $LatticeSearch(CL_{IN}, SearchType, paramSet)$: rechercher dans le treillis CL_{IN} et récupérer la liste des opérations qui offrent des paramètres d'entrée similaires à l'ensemble des paramètres *paramSet* tenant compte du type de recherche *SearchType* (Exact en cas d'invocation totale et Approximate en cas d'invocation partielle).
- $LatticeSearch(CL_{OUT}, SearchType, paramSet)$: rechercher dans le treillis CL_{OUT} et récupérer la liste des opérations qui offrent des paramètres de sorties similaires à l'ensemble des paramètres *paramSet* tenant compte du type de recherche *SearchType* (Exact en cas d'invocation totale et Approximate en cas d'invocation partielle).

– **nameMatch** : une fonction $nameMatch(p_1.name, p_2.name, f, \theta)$ retourne vrai si : la similitude entre $p_1.name$ et $p_2.name$, retournée par une mesure de similarité f, est supérieure à une certaine valeur de seuil λ : i.e, $f(p_1.name, p_2.name) \geq lambda$.

Afin de calculer la similitude entre les noms des paramètres (chaînes de caractères), nous proposons d'utiliser les fonctions suivantes :

- **Correspondance exacte** : $p_1.name$ et $p_2.name$ sont dits similaires si ($p_1.name = p_2.name$).
- **Fonction de similitude approximative** : utilise des fonctions de distances pour évaluer la similitude entre deux chaînes de caractères. $p_1.name$ et $p_2.name$ sont considérés comme similaires si la valeur d'une fonction de similitude entre ces deux noms de paramètres (chaînes de caractères) est au dessus d'une certaine valeur de seuil. Dans ce contexte, nous proposons d'utiliser par exemple la similarité **cosinus** ou la similarité basée sur **WordNet**. Les valeurs de la similarité de cosinus et WordNet sont dans l'intervalle [0,1] où 0 indique l'absence de similarité entre les vecteurs et 1 indique que les vecteurs sont identiques.

  1. **Similarité cosinus** : la similarité entre $p_1.name$ et $p_2.name$ est mesurée par la mesure de proximité *Cosinus* en calculant la valeur du cosinus de l'angle entre deux vecteurs TF-IDF v_1 et v_2 associés à ces paramètres, en utilisant l'équation 4.7. Les deux vecteurs v_1 et v_2 sont construits en calculant les poids TF-IDF des tokens caractérisant les chaînes de caractères associés aux noms des paramètres.
  2. **Similarité basée sur WordNet** : WordNet [1] mesure la similarité approximative entre deux mots. WordNet est une ressource linguistique de la langue anglaise largement utilisée pour découvrir des synonymes. Considérons par exemple la fonction $nameMatch("flight","plane", WordNet, 0.9)$ qui retourne vrai car les deux noms des paramètres ont une similarité égale à 1 (> 0.9).

4.5 Expérimentation et évaluation

4.5.1 Collection de services web

Toutes les expérimentations réalisées dans cette thèse sont réalisées sur des services web réels obtenus à partir de la collection de test publique appelée *SAWSDL-TC3* [2]. Cette collection de test est bien connue et est largement utilisée par la communauté des services Web. Elle est utilisée initialement dans la plate-forme d'évaluation de matchmakers sémantiques pour les services web SME2 [3] (Semantic Web Service Matchmaker Evaluation Environment). SME2 est aussi le corpus du concours annuel reconnu *Semantic Service Selection (S3)*, organisé depuis 2007 afin d'évaluer des approches sémantiques de découverte de services web. Cette collection se compose de 1088 documents WSDL

1. https://wordnet.princeton.edu/
2. http://www.semwebcentral.org/projects/sawsdl-tc
3. SME2:http://semwebcentral.org/projects/sme2/

annotés sémantiquement qui couvrent 9 différents domaines d'application. Chaque service web appartient à l'un des neuf domaines, à savoir : Communication, Education, Economy, Food, Geography, Medical, Military, Travel, Simulation. Le tableau 4.2 indique le nombre de services dans chaque domaine.

#	**Domaine**	**Services**
1	Communication	58
2	Economy	358
3	Education	285
4	Food	34
5	Geography	60
6	Medical	73
7	Military	40
8	Simulation	16
9	Travel	164
	Total	**1088**

Table 4.2 – Nombre de services utilisés pour chaque domaine

4.5.2 Préparation des données et extraction de thèmes

Avant d'appliquer les algorithmes et les systèmes proposés dans cette thèse, nous traitons le corpus de services web (i.e. documents WSDL) en appliquant quelques techniques d'extraction d'information présentées dans le chapitre 2 (voir section 2.3.1, page 23). L'objectif de ce traitement est d'identifier les descriptions textuelles des services (i.e. des mots potentiels) qui décrivent la sémantique de leurs fonctionnalités. Le processus du traitement de documents WSDL se compose de plusieurs étapes :

- Analyse et extraction d'information ;
- Tokénisation, suppression des mots vides ;
- Lemmatisation ;
- Pondération des termes et calcul des poids TF-IDF ;
- Construction des représenations vectorielles et la matrice transactionnelle des services STM (Service Transaction Matrix).

Aprés avoir identifié tous les termes, la fréquence de ces termes est calculée pour tous les services. La technique VSM (Vector Space Model) est utilisée pour représenter chaque service comme un vecteur de ces termes. En effet, cette technique présente la description d'un service sous forme d'une représentation vectorielle afin de faciliter l'analyse des données. VSM est identifiée comme la représentation la plus utilisée pour les documents et est une méthode trés utile pour analyser les descriptions des services dans la recherche d'information. L'algorithme TF-IDF est utilisé pour représenter l'ensemble des descriptions des services et les convertir sous forme de VSM. L'ensemble de services

sont donc représentés par une *matrice transactionnelle* où chaque ligne représente la description d'un service, chaque colonne représente un mot du corpus (vocabulaire) et chaque entrée représente le poids TF-IDF d'un mot apparaissant dans la description d'un service (voir chapitre 2, section 2.3.1).

La seconde étape importante du processus de préparation de données et l'apprentissage du modèle probabiliste à thèmes corrélés CTM et l'extraction d'un ensemble de thèmes à partir des descriptions de services. Les données textuelles observées produites dans l'étape précédente et représentées dans la matrice *STM* sont utilisées comme donnée d'entraînement pour construire le modèle thématique probabiliste CTM. Pour extraire les thèmes à partir de descriptions des services web, nous avons utilisé l'algorithme *ctm-c* [4] implémenté et distribués par Blei [5]. Cette implémentation utilise l'algorithme *Variationnel EM* pour l'estimation des paramètres et l'inférence (voir chapitre 2, section 2.3.3). Après avoir formé le modèle CTM, les distributions de probabilités de services sur les thèmes sont connues. Nous rappelons que pour construire notre modèle probabiliste, le nombre de thèmes doit être choisi avant la phase d'entraînement. Le choix du nombre de thèmes correspondant à l'ensemble de données d'origine a un impact sur l'interprétation des résultats [Steyvers & Griffiths, 2007]. Pour déterminer le nombre optimal de thèmes correspondant au corpus de données utilisé, nous calculons la mesure de *Perplexité* qui est largement utilisée pour évaluer la performance des modèles probabilistes [Blei et al., 2003]. Supposons que nous avons M services web dans le corpus de test D_{test} et chaque service s contient M_s mots. La perplexité d'un ensemble de données D_{test} est alors définie par l'équation 4.8. Des valeurs basses de perplexité indiquent de meilleures performances et le nombre optimale de thèmes.

$$Perplexity = \exp\left(-\sum_{i=1}^{M}\sum_{m=1}^{M_s}\frac{\log P(w_m|s_i)}{\sum_{i=1}^{M} M_i}\right) \tag{4.8}$$

Où $P(w_m|s_i)$ est la probabilité que le terme w_m appartienne au i-ième service.

La figure 4.6 montre les valeurs de la perplexité obtenues pour le modèle probabiliste CTM construit, à partir de la collection SAWSDL-TC3, en faisant varier le nombre de thèmes de 10 à 110 par un pas de 10. Nous remarquons que l'on obtient une meilleure performance du modèle avec le nombre de thèmes optimal $k = 90$. Nous utilisons donc, pour toutes les expérimentations réalisées dans cette thèse, le modèle généré avec 90 thèmes.

4. http://www.cs.columbia.edu/~blei/ctm-c/index.html
5. http://www.cs.columbia.edu/~blei

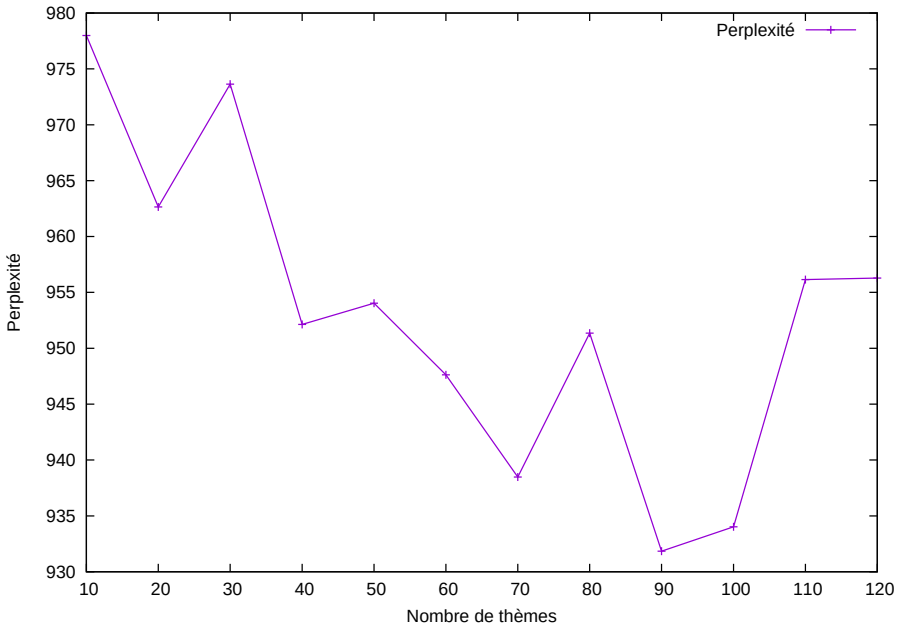


Figure 4.6 – Les valeurs de perplexité obtenues pour le modèle probabiliste CTM.

4.5.3 Protocole d'expérimentation

Les différentes étapes du protocole expérimental réalisé dans ce chapitre sont décrites ci-dessous :

1. **Préparation des données et extraction de thèmes :** voir la section précédente.
2. **Construction des réseaux d'intéraction** : comme nous l'avons décrit dans ce chapitre, le réseaux d'interaction de services est calculé à partir du réseaux d'interaction d'opérations. Pour construire les treillis de concepts utilisés pour produire les réseaux d'interaction de services, nous utilisons l'outil *Lattice Miner*[6] [Lahcen & Kwuida, 2010]. C'est un logiciel open source d'analyse de concepts formels pour la construction, la visualisation et la manipulation de treillis de concepts. Il permet l'extraction de concepts formels et la génération de règles d'association ainsi que la transformation de contextes formels, la réduction et la généralisation d'objets/attributs. Dans cette expérimentation, nous utilisons la correspondance exacte (i.e, l'égalité littérale =) pour déterminer la similarité entre les noms des paramètres d'entrée/sortie.

6. http://sourceforge.net/projects/lattice-miner/

	Réseaux d'interaction de services web	
Propriétés	**invocation totale**	**invocation partielle**
Taille (nombre de nœuds) Nombres d'arcs	1088 5776	1088 6729
nœuds isolés : - Nombre - Proportion	 507 46.60%	 435 39.78%
Densité	0.0052	0.0060
Degré moyen	10.9914	12.8049
Transitivité	0.0123	0.0162
Diamètre	8	7
- Nombre de petites composantes - Taille de la composante géante	5 539 nœuds (51.284%)	5 634 nœuds (60.324%)

Table 4.3 – Propriétés des réseaux d'interaction de services web extraits de la collection de test SAWSDL-TC3

3. **Construction des réseaux de similitude** : pour construire le réseaux de similitude de services web, nous nous basons sur la similarité Cosinus. La similarité entre les services est calculée en se basant sur les vecteurs de compte TF-IDF dans le cas de réseaux de similitude syntaxiques. Dans le cas de réseaux de similitude sémantiques, la distribution de probabilités sur les thèmes est utilisée comme critère de base pour calculer la similarité sémantique entre les services. Comme décrit auparavant, un lien est créé entre deux services si et seulement si le degré de similitude entre les deux vecteurs décrivant ces services est supérieur à un seuil donné. Dans nos expérimentations, nous allons construire et analyser les deux modèles de réseaux de similitude en évaluant plusieurs valeurs de seuil dans l'intervalle [0,1].

4. **Analyse des réseaux calculés** : afin d'évaluer la structure des réseaux d'interaction/similitude de services web extraits de la collection de test SAWSDL-TC3, nous avons calculé les propriétées suivantes : la densité, le diamètre, la transitivité le degré moyen (voir section 4.2.2). Pour calculer les différentes propriétés, nous avons utilisé l'outil Pajek [7]. Pajek est un logiciel largement utilisé pour analyser et visualiser des réseaux de grandes tailles.

4.5.4 Analyse et évaluation de réseaux d'interaction

Dans cette section, nous analysons les deux modèles de réseaux d'interaction de services web basés respectivement sur le mode d'invocation d'opérations (invocation totale et invocation partielle).

7. http://mrvar.fdv.uni-lj.si/pajek/

Les figures 4.7 et 4.8 représentent respectivement les réseaux d'interaction de services web en mode d'invocation totale et partielle, extraits à partir de la collection de test SAWSDL-TC3. Les différentes valeurs obtenues pour les principales propriétées des réseaux sont représentées dans le tableau 4.3. Les réseaux d'interaction de services, en mode d'invocation totale et partielle, possèdent tous les deux presque les mêmes caractéristiques globales. Le nombre de nœuds de services web est identique que ce soit dans le cas d'invocation partielle ou totale. Ce nombre est 1088 ; cette valeur correspond au nombre de documents WSDL dans la collection de test SAWSDL-TC3. Les nœuds du réseau sont répartis entre composantes géantes, petites composantes et un ensemble de nœuds isolés. Le réseau d'interaction en mode d'invocation totale contient moins d'arêtes (ou d'arcs) entre les nœuds que le réseau d'interaction en mode d'invocation partielle. En effet, les arêtes reflètent la présence d'opérations totalement ou partiellement invocables entre les nœuds de service web. Les opérations partiellement invocables comprennent un ensemble d'opérations qui sont totalement invocables. Nous pouvons également observer que le coefficient de transitivité est élevé dans le réseau d'interaction en mode d'invocation partielle. Cela s'explique par le fait que ce réseau possède le plus grand nombre d'arêtes. Le réseau d'interaction en mode d'invocation totale se caractérise par sa plus faible valeur de densité. La densité est généralement un indicateur de la proportion des arêtes. Ainsi, la recherche de possibles compositions est plus rapide dans le réseau d'interaction en mode d'invocation totale. Notons également que le diamètre indique le nombre d'opérations requises dans les grandes compositions. Il est défini comme la plus grande distance entre n'importe quelle paire de nœuds dans un réseau.

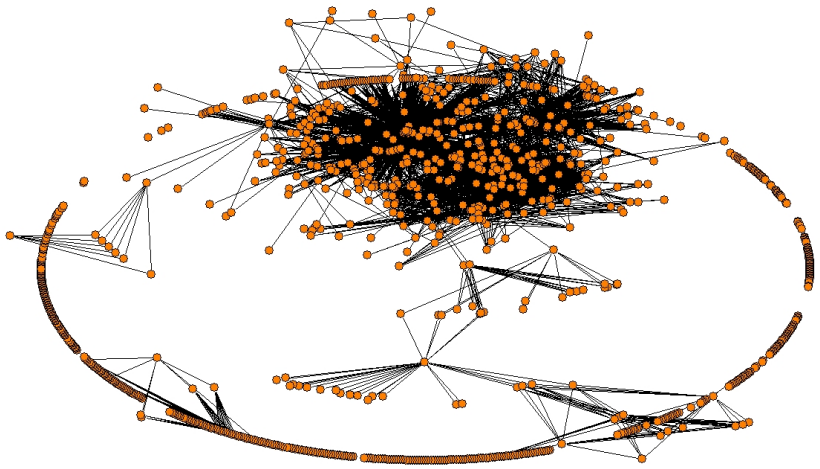


Figure 4.7 – Réseau d'interaction de services en mode d'invocation totale calculé avec la collection SAWSDL-TC3.

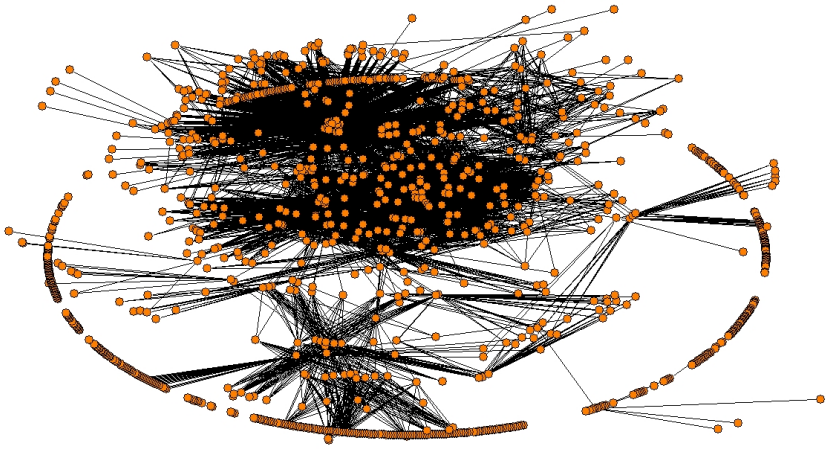


Figure 4.8 – Réseau d'interaction de services en mode d'invocation partielle calculé avec la collection SAWSDL-TC3.

4.5.5 Analyse et évaluation de réseaux de similitude

Dans cette section nous analysons les deux modèles de réseaux de similitude basés respectivement sur les descriptions syntaxiques et sémantiques des services web. Comme nous l'avons déjà évoqué, le degré de similitude entre deux services dans un réseau de similitude doit être supérieur à la valeur d'un seuil donné afin de pouvoir créer un lien entre ces deux services. Nous avons utilisé la mesure cosinus (voir équation 4.7) pour identifier la similarité entre les services web. Nous rappelons que les valeurs de la similarité de cosinus sont dans l'intervalle $[0, 1]$. Concentrons-nous tout d'abord sur les réseaux syntaxiques. Le tableau 4.4 résume les principales propriétés de ces réseaux en variant les valeurs du seuil. Les réseaux contiennent chacun 1088 nœuds. Le nombre de nœuds isolés augmente en augmentant la valeur du seuil. Nous observons que ces réseaux contiennent des faibles proportions de nœuds isolés pour un seuil inférieur ou égal à 0.5 (qui sont respectivement 0.46%, 1.29% et 4.41%). Cela s'explique par l'augmentation du nombre de liens présents dans ces réseaux. Autrement dit, cela offre à ces réseaux plus de possibilités de créer des liens. Lorsque le seuil augmente (i.e. $>= 0.75$), les proportions de nœuds isolés augmentent également et le nombre de liens entre les services diminue. La plus grande proportion des neouds isolés pour les réseaux syntaxique est 76.93% , quand le seuil est égal à 1. Cela signifie que ce réseau contient dans ce cas plus de services similaires.

Dans le cas du réseau sémantique dont les résultats sont reportés dans le tableau 4.5, nous remarquons que ce dernier a presque le même comportement que le réseau syntaxique concernant les proportions de nœuds isolés pour un seuil supérieur ou égal à 0.75. Lorsque le seuil varie entre 0.1 et 0.5, le réseau sémantique contient la plus faible

	Seuil					
Propriétés	**0.1**	**0.25**	**0.5**	**0.75**	**0.9**	**1.0**
Taille (nombre de nœuds)	1088	1088	1088	1088	1088	1088
Nombres de liens	52285	22976	10291	4384	2662	204
nœuds isolés :						
- Nombre	5	14	48	167	430	837
- Proportion	0.46%	1.29%	4.41%	15.35%	39.52%	76.93%
Densité	0.088	0.0388	0.0173	0.0074	0.0044	0.00034
Degré moyen	96.1121	42.2352	18.9172	8.0588	4.8933	0.3750
Transitivité	0.5384	0.6152	0.8462	0.96192	0.98297	0.9723
Diamètre	6	10	16	10	4	2
Petites composantes :						
- Nombre	2	3	20	77	85	25

Table 4.4 – Propriétés du réseau de similitude syntaxique de services web extraits de la collection de test SAWSDL-TC3

	Seuil					
Propriétés	**0.1**	**0.25**	**0.5**	**0.75**	**0.9**	**1.0**
Taille (nombre de nœuds)	1088	1088	1088	1088	1088	1088
Nombres de liens	31372	23151	14492	8672	5719	2238
nœuds isolés :						
- Nombre	1	1	1	10	80	354
- Proportion	0.09%	0.09%	0.09%	0.92%	7.35%	32.54%
Densité	0.0530	0.03911	0.02448	0.01465	0.0096	0.0037
Degré moyen	57.6691	42.5569	26.6397	15.9411	10.5128	4.1139
Transitivité	0.5344	0.6519	0.7943	0.9270	0.9513	0.9235
Diamètre	6	8	14	19	7	7
Petites composantes :						
- Nombre	4	7	15	40	89	86

Table 4.5 – Propriétés du réseau de similitude sémantique de services web extraits de la collection de test SAWSDL-TC3

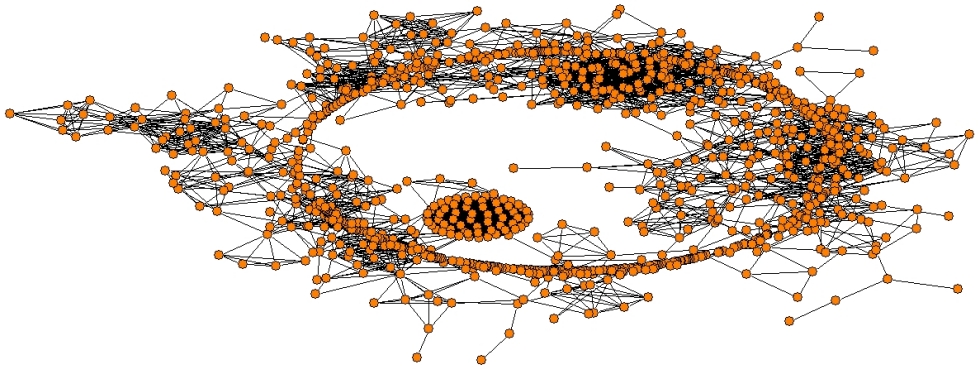


Figure 4.9 – Réseau de similitude syntaxique de services web extraits de la collection de test SAWSDL-TC3 - Seuil = 0.75

proportion de nœuds isolés (0.09%). Les différences observées peuvent s'expliquer par le fait que la construction du réseau sémantique se base sur les sémantiques capturées à travers les thèmes alors que le réseau syntaxique est construit en se basant sur les représentations vectorielles TF/IDF des services. Ainsi, le nombre de liens est plus grand dans le réseau sémantique. Si l'on se réfère à la densité, là encore les valeurs de cette dernière sont plus élevées dans le réseau sémantique que dans le réseau syntaxique. Cela se traduit naturellement par la présence de la plus grande proportion des liens dans le réseau sémantique. Les figures 4.9 et 4.10 représentent respectivement les deux réseaux de similitude syntaxique et sémantique construit avec le seuil 0.75.

Nous pouvons également observer que les valeurs de la transitivité relevées du réseau syntaxique sont proches de 1 et légèrement supérieures à celles du réseau sémantiques lorsque le seuil augmente (i.e., >= 0.5). Cela peut s'expliquer par la présence de triangles dans ces réseaux. En comparant le nombre de composantes selon les différentes valeurs du seuil, les différences observées entre ces deux réseaux ne sont pas très intéressantes. Lorsque le seuil est entre 0.5 et 0.75, le réseau syntaxique contient le plus grand nombre de composantes (dont la taille minimale d'une composante est 3 nœuds). Dans les autres cas, c'est le réseau sémantique qui possède le plus grand nombre de composantes.

4.6 Conclusion

Dans ce chapitre, nous avons présenté un ensemble de définitions et propriétés relatives à la topologie des réseaux ainsi que leurs domaines d'application. Ensuite, nous avons présenté deux modèles de réseaux de services web (i.e. réseaux d'interaction et réseaux de similitude) ainsi que leurs méthodes de constructions. Dans le cas de réseaux d'interaction,

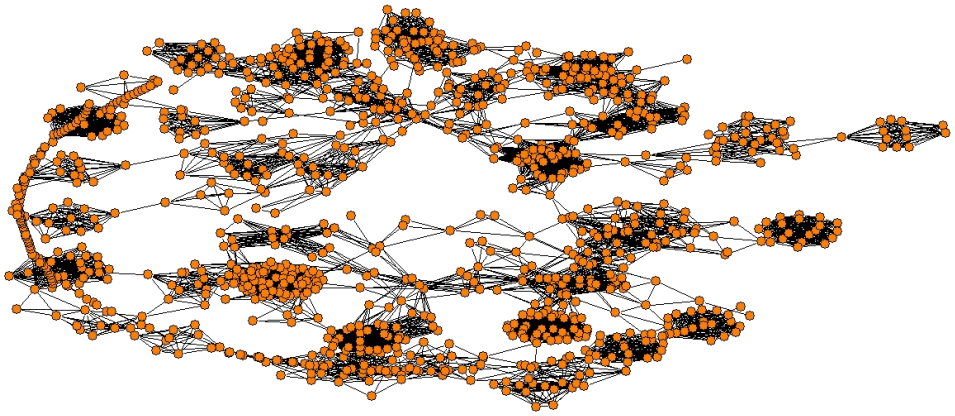


Figure 4.10 – Réseau de similitude sémantique de services web extraits de la collection de test SAWSDL-TC3 - Seuil = 0.75

nous avons considéré trois niveaux de granularité (i.e, paramètres, opérations et services). Nous avons dans un premier temps construit le réseau d'interaction d'opérations en calculant des treillis de concepts formels pour déduire ensuite le réseau d'interaction de services. Les réseaux d'interaction d'opérations se distinguent en fonction du mode d'invocation (totale ou partielle). Les réseaux d'interaction sont généralement utilisés pour identifier les liens de compositions entre les services web. Nous avons également proposé deux modèles de réseaux de similitude de services web qui se basent respectivement sur les descriptions syntaxiques et sémantiques des services web. Les réseaux de similitude sont conçus sur la base de la similitude entre les services et sont généralement liés à la classification des services web et peuvent également être utilisés pour identifier et construire des communautés de services web. Afin d'identifier la similarité entre deux services, nous avons utilisé une mesure de proximité qui utilise le cosinus de l'angle entre deux vecteurs (basés sur les distributions de probabilités sur les thèmes ou les vecteurs de comptes TF-IDF). Dans cette thèse, les réseaux d'interaction sont utilisés pour identifier les liens de compositions entre les services web (voir chapitre 5, section 5.5). Les réseaux de similitude sont utilisés pour étudier la structure communautaire de ces réseaux (voir chapitre 7).

Dans le chapitre suivant, nous proposons une nouvelle méthode pour diversifier les résultats de la découverte de services web. Nous proposons également une méthode permettant l'identification et la découverte de possibles compositions de services web.

5

Diversification des résultats de la découverte de services web et découverte de leurs possibles compositions

Sommaire

5.1 Introduction

La découverte de services web et la composition sont en général des tâches complexes qui nécessitent un effort considérable, surtout avec l'évolution rapide de la technologie des services web et la grande quantité de services web disponibles sur Internet. Plusieurs travaux ont été proposés dans la littérature pour améliorer le processus global de ces deux principales tâches. La majorité des systèmes existants découvrent des services web qui sont très similaires et/ou redondants. D'une part, ces systèmes ignorent complétement la notion de la diversité dans la liste des résultats retournée. D'autre part, certains systèmes n'arrivent pas parfois à découvrir des services pertinents qui peuvent satisfaire la demande de l'utilisateur. Il est donc nécessaire de renvoyer à l'utilisateur une séquence de services qui peuvent être composables afin de répondre à certaines requêtes complexes. Pour remédier à ces problèmes, nous proposons dans ce chapitre une nouvelle méthode qui permet de découvrir simultanément :

1. Un ensemble de services web divers et pertinents qui peuvent répondre à la requête de l'utilisateur ;
2. Un ensemble de compositions optimales possibles qui peuvent exister entre les services web découverts et/ou avec d'autres services.

Dans notre travail, nous proposons une méthode de diversification des résultats de la découverte en se basant à la fois sur la notion de pertinence (ou similarité), la diversité et la densité des services pour découvrir des services divers correspondant à une requête donnée. Nous exploitons également les interactions qui peuvent exister entre l'ensemble de services web disponibles exprimées sous forme d'un réseau d'interaction (ou graphe de dépendance) de services web. Nous pouvons résumer les principales contributions de ce chapitre comme suit :

1. Une méthode de découverte et de classement de services web en se basant sur les mesures de la pertinence, diversité et densité afin de minimiser la redondance dans la liste renvoyée à l'utilisateur (voir section 5.4).
2. Nous exploitons le réseau d'interaction de services web construit dans le chapitre 4 (voir section 4.4.4, page 82) pour identifier les compositions possibles des services web qui sont susceptibles d'être composables (voir section 5.5).
3. Une méthode permettant l'optimisation et la diversification des liens de compositions identifiés (voir section 5.5.3).

Dans nos expérimentations, nous avons comparé la précision de la méthode proposée avec trois méthodes proposées dans la littérature. Les résultats obtenus montrent que

l'on obtient des valeurs de précision élevées pour notre méthode par rapport à plusieurs systèmes de la littérature.

Le reste de ce chapitre est structuré comme suit. La section 5.2 introduit la motivation de ce travail en l'illustrant par un exemple motivant. La section 5.3 décrit et présente un aperçu global du système proposé dans ce chapitre. Dans la section 5.4, nous décrivons en détail notre méthode de diversification des résultats de la découverte de services web. La section 5.5 décrit la méthode d'identification et de construction de compositions de services web. Enfin, les expérimentations et évaluations réalisées sont présentées dans la section 5.6 avant de conclure dans la section 5.7.

5.2 Motivation et formulation du problème

Les architectures orientées services (Service Oriented Architecture ou SOA) permettent de concevoir des environnements distribués et mettent en oeuvre des services avec une forte cohérence interne. L'objectif principal des recherches focalisées sur les services web est la découverte des services pertinents permettant de satisfaire les besoins des utilisateurs. Comme le nombre de services web disponibles sur Internet augmente d'une manière significative, la tâche de découverte de services constitue toujours un défi majeur de ce domaine de recherche. Cependant, la majorité des services web recommandés à l'utilisateur sont en général très similaires et/ou redondants. En outre, la plupart des systèmes échouent parfois à apporter une réponse satisfaisante pour répondre aux requêtes utilisateurs complexes et/ou ambiguës. Il est donc nécessaire de combiner ou composer plusieurs services pour satisfaire ce genre de requêtes. Par conséquent, la composition flexible de services web pour répondre aux exigences complexes est considérée également parmi les objectifs les plus importants dans notre domaine de recherche.

Récemment, certains portails de services web et moteurs de recherche comme Biocatalogue [1], Seekda ! [2] et WS-Portal [3] [Aznag et al., 2015] offrent différentes caractéristiques permettant de faciliter la découverte de service web. Comme décrit précédemment, les moteurs de recherche découvrent en général des services très similaires et échouent parfois à satisfaire certaines requêtes complexes. Pour faire face à ce problème, deux questions importantes doivent être envisagées :

1. Comment peut-on choisir les services qui correspondent à une requête donnée q lorsque les services retournés sont très similaires ;
2. Comment composer plusieurs services pour satisfaire q lorsque aucun des services disponibles ne peut répondre individuellement à la requête.

1. https://www.biocatalogue.org/
2. http://webservices.seekda.com/
3. http://wsportal.aznag.net/

Dans ce qui suit, nous illustrons le problème de la découverte de services web et leurs compositions par un exemple motivant. Nous considérons huit services web décrits ci-dessous :

- Étant donné le titre d'un livre, le service *BookFinder* retourne des informations sur le livre (i.e. Numéro ISBN, auteur du livre, titre du livre, date de publication).
- Étant donné le numéro *ISBN*, le service *BookPublisher* retourne l'éditeur du livre.
- Étant donné le numéro *ISBN*, le service *BookTaxedpriceprice* retourne le prix et le prix imposé du livre concerné.
- Étant donné le numéro *ISBN*, le service *BookRecommendedprice* retourne le prix recommandé du livre en question.
- Étant donné l'auteur, le titre et l'éditeur du livre, le service *BookPricesizebook-type* retourne le prix, la taille et le type du livre.
- Étant donné l'auteur, le titre et l'éditeur du livre, le service *BookReaderreview* renvoie les lecteurs et les relecteurs du livre concerné.
- Étant donné l'auteur, le titre, l'éditeur et la date de publication du livre, le service *BookAuthortext* retourne le résumé du livre.
- Étant donné l'auteur du livre, le service *BookReaderreview* renvoie le prix maximum et le titre du livre.

Les services web décrits ci-dessus ont été obtenus à partir de la collection standard du test *SAWSDL-TC3*[4] (voir section 4.5.1, page 85). Cette collection a été largement utilisée pour tester de nombreuses approches dans la littérature et contient 1088 documents WSDL. Ces documents WSDL sont sémantiquement annotés et appartiennent aux neuf différents domaines d'applications. Les noms des documents WSDL et l'ensemble des paramètres associés aux services web de notre exemple sont respectivement présentés dans les tableaux 5.1 et 5.2. La figure 5.1 montre le graphe de dépendance des paramètres des services utilisés.

#	Service	Document WSDL
WS1	BookFinder	BookFinder.wsdl
WS2	BookPublisher	book_publisher.wsdl
WS3	BookTaxedprice	book_taxedpriceprice.wsdl
WS4	BookRecommendedprice	book_recommendedprice.wsdl
WS5	BookPricesizebook-type	book_pricesizebook-type.wsdl
WS6	BookReaderreview	book_readerreview.wsdl
WS7	BookAuthortext	book_authortext_service.wsdl
WS8	AuthorBookmaxprice	author_bookmaxprice.wsdl

Table 5.1 – Ensemble de services web de l'exemple de motivation.

Considérons maintenant les requêtes suivantes :

4. http://www.semwebcentral.org/projects/sawsdl-tc

#	Paramètre	#	Paramètre
A	Title	I	TaxedPrice
B	ISBN	J	RecommendedPrice
C	Author	K	Size
D	BookTitle	L	BookType
E	Date	M	Reader
F	Publisher	N	Reviewer
G	Price	O	AbstractText
H	MaxPrice		

Table 5.2 – Ensemble de paramètres de l'exemple de motivation.

- q_1 : Étant donné le titre d'un livre existant (i.e. *"Service-Oriented Architecture (SOA) : Concepts, Technology, and Design"*), nous cherchons à avoir le maximum d'informations sur ce livre, en particulier, l'auteur du livre (i.e. *"Thomas Erl"*), l'éditeur, le numéro ISBN et la date de publication, ainsi que ses prix respectifs (prix, prix recommandé et prix imposé).
- q_2 : Ensuite, nous aimerions trouver quelques informations supplémentaires concernant le livre tel que le résumé, les lecteurs et les relecteurs, ainsi que le type (par exemple livre relié ou broché), et la taille (petite, moyenne ou grande).
- q_3 : Enfin, nous voudrions trouver un autre livre publié (i.e *"SOA Design Patterns"*) du même auteur avec un prix maximum.

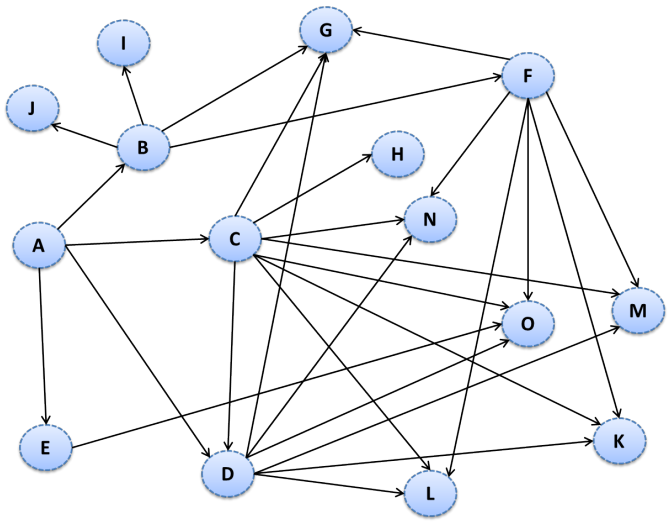


Figure 5.1 – Graphe de dépendance des paramètres I/0 des services utilisés dans l'exemple motivant.

Dans notre registre de services (i.e. collection de test), il n'y a aucun service Web qui peut satisfaire seul chacune de ces requêtes. Par conséquent, pour répondre à la requête q_1,

nous devons combiner une séquence de services Web (i.e. BookFinder, BookTaxedPrice et BookRecommendedPrice). Dans un premier temps, étant donné le titre du livre fourni par l'utilisateur, le service *BookFinder* peut être invoqué pour retourner l'auteur, le numéro ISBN, et la date de publication du livre. Ensuite, le numéro ISBN du livre renvoyé par le premier service est utilisé par le service *BookPublisher* pour avoir l'éditeur, et par le service web *BookTaxedPrice* qui à son tour retourne le prix et le prix imposé du livre, mais il ne parvient pas à retourner le prix recommandé. Par contre, le service web *BookRecommendedPrice* peut renvoyer le prix recommandé en utilisant le numéro ISBN du livre.

Afin de satisfaire la requête q_2, plusieurs services web peuvent être composés comme suit :

1. Invoquer le service web *BookFinder* pour avoir l'auteur, le numéro ISBN et la date de publication. Ensuite, pour trouver l'éditeur du livre, le service *BookPublisher* peut être appelé en utilisant le numéro ISBN renvoyé par le service *BookFinder*.
2. En utilisant, l'éditeur, l'auteur et le numéro ISBN, nous pouvons retourner le résumé du livre en appelant *BookAuthortext*, les lecteurs et relecteurs en invoquant le service *BookReaderReview*. Le type et la taille peuvent être fournis par le service *BookPricesizebook-type*.

De même pour la requête q_3, aucun des huit services web ne peut satisfaire cette requête. Étant donné le titre du livre, le service *BookFinder* peut être appelé pour retourner l'auteur, le numéro ISBN et la date de publication. Ensuite, nous pouvons invoquer le service *BookReaderReview* pour retourner le prix maximal en utilisant l'auteur du livre. En plus, pour retourner l'éditeur du livre en question, nous pouvons appeler le service *BookPublisher* en utilisant le numéro ISBN renvoyé par le service web *BookFinder*.

L'exemple que nous avons décrit ci-dessus n'illustre qu'un simple et petit problème de découverte et de composition de services web dans lequel plusieurs services doivent être combinés pour répondre à une demande de l'utilisateur. Cependant, les problèmes de la composition de services web sont en réalité plus complexes que notre exemple. Ceci est dû au nombre de services web qui augmente de façon significative et lorsqu'une demande de l'utilisateur ne peut pas être satisfaite par des services individuels et même par une composition de services web.

Comme nous l'avons mentionné précédemment, les services web découverts peuvent contenir des services très similaires redondants et inutiles. Par conséquent, le niveau de satisfaction de l'utilisateur se réduit. Afin d'éviter la redondance tout en maintenant la qualité des services web découverts, la diversité doit être prise en considération dans les systèmes de découverte et/ou de recommandation. Dans ce chapitre, nous présentons une nouvelle méthode de classement pour la diversification des résultats de la découverte de services web (voir section 5.4). La diversification des résultats de recherche a été déjà étudiée dans le contexte de la recherche de documents sur le web [Bache et al., 2013].

Notre système permet de sélectionner les services divers à partir des premiers services web classés en tenant compte des mesures de pertinence, de diversité et de densité.

Dans le cas de requêtes complexes, les services web disponibles ne répondent pas parfois à ce genre de requêtes. Par conséquent, la combinaison de plusieurs services web s'avère nécessaire pour satisfaire les demandes des utilisateurs. Nous traitons le problème de la composition de services web comme un problème de recherche dans un réseau de services web. Nous modélisons les dépendances entre un ensemble de services web sous forme de réseau où les noeuds représentent les services et les liens symbolisent les itérations entres ces services. Nous exploitons également la notion de diversité dans les graphes pour optimiser la liste des différentes compositions possibles (voir section 5.5).

5.3 Description du système proposé

Nous présentons dans cette section un aperçu global du système proposé ainsi que ces différentes composantes. Dans notre approche, nous supposons que tous les services sont décrits par le langage de description syntaxique WSDL et/ou le langage sémantique SAWSDL. Comme nous l'avons mentionné précédemment, la majorité des services web disponibles sur Internet sont décrits par un document WSDL. Le document WSDL contient la description ainsi que toutes les spécifications nécessaires pour utiliser le service web en décrivant le protocole de communication, le format de message requis pour communiquer avec le service, les opérations que le client peut invoquer et la localisation du service. Afin de traiter efficacement les services, nous avons extrait toutes les descriptions textuelles ainsi que les caractéristiques qui décrivent un service web. Nous avons également extrait à partir du document WSDL, toutes les opérations et leurs paramètres d'entrée/sortie (voir chapitre 4, section 2.3.1, page 23). Ces paramètres sont utilisés pour construire le réseau d'interaction décrivant les relations d'interaction entre les différents services web.

La figure 5.2 montre un aperçu global du système proposé dans ce chapitre avec ses différents modules ou services. Nous distinguons deux services principaux :

- Service de découverte et de classement de services web correspondant à la demande de l'utilisateur.
- Service d'identification et de sélection de compositions des services web renvoyés par le service de découverte.

La méthode de découverte et de classement de services web proposée dans ce chapitre étend la méthode de découverte proposée récemment par [Aznag et al., 2013a, Aznag et al., 2014]. Dans notre approche, nous utilisons le modèle thématique probabiliste à thèmes corrélés (i.e. CTM - Correlated Topic Model) pour extraire un ensemble de thèmes à partir des descriptions de services web. Chaque thème extrait est associé à un groupe de concepts textuels (i.e. mots) et/ou des concepts sémantiques qui apparaissent dans les descriptions de services web. Les thèmes sont exprimés sous forme de distribu-

tions de probabilités sur les mots. L'utilisation des modèles thématiques probabilistes vise à réduire la dimensionnalité de l'espace des services web en capturant des relations sémantiques entre thèmes-mots et services-thèmes, exprimées sous forme de distributions de probabilités (voir chapitre 2, section 2.3.2, page 27). La contribution principale de notre méthode est la diversification des résultats de la découverte des services web et minimisation de la redondance tout en découvrant des services pertinents.

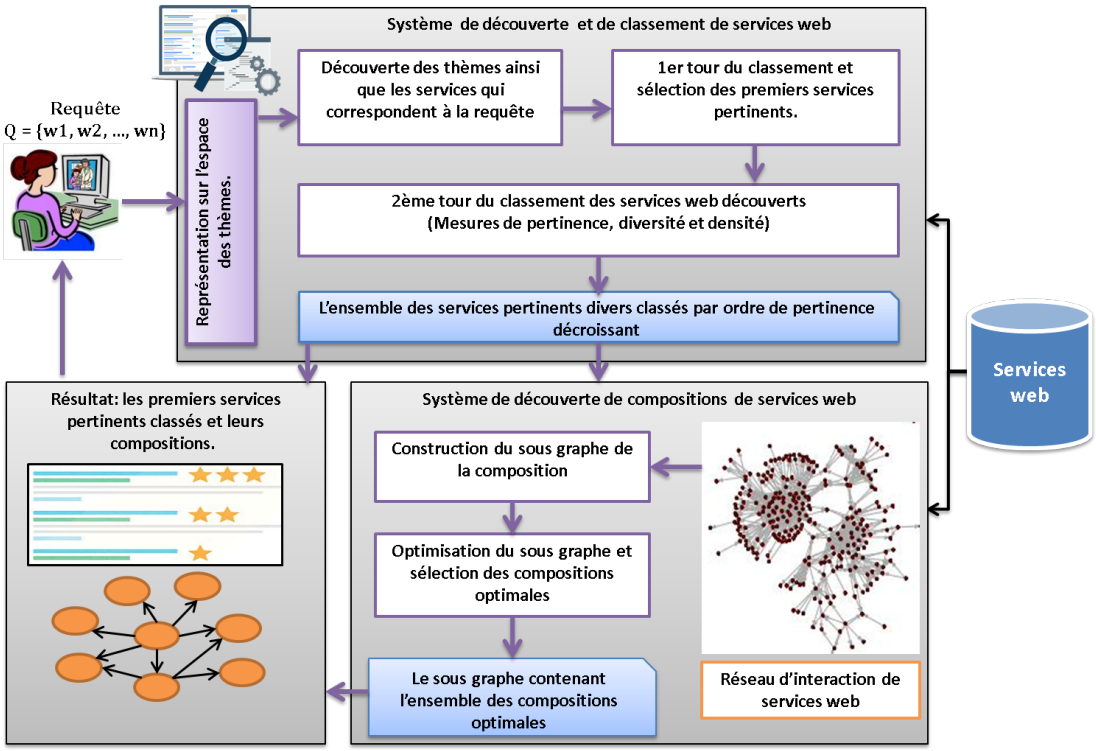


Figure 5.2 – L'aperçu général du sytème proposé dans ce chapitre.

Comme le montre la figure 5.2, notre système de découverte retourne un ensemble de services pertinents divers classés par ordre de pertinence décroissant. Pour cela, nous représentons, dans un premiers temps, la requête de l'utilisateur dans l'espace des thèmes afin de découvrir l'ensemble des premiers thèmes implicites qui sont sémantiquement associés à la requête. En se basant sur les thèmes sélectionnés, nous calculons la similarité entre la requête et les services web liés à ces thèmes. Ensuite, nous utilisons les scores de la similarité pour classer et sélectionner les services web. Afin de minimiser la redondance dans les résultats de découverte nous classons les k premiers services web classés dans l'étape précédente en tenant compte de la pertinence, la diversité et la densité des services (voir section 5.4). Ensuite, le système de découverte de compositions possibles prend comme entrée (1) la liste des services pertinents divers classés par le système

de découverte et (2) le réseau d'interaction de services web construit et retourne un sous graphe contenant l'ensemble des compositions optimales (voir la section 5.5). Pour cela, notre système effectue une recherche dans le réseau d'interaction de services afin d'extraire un sous-graphe contenant l'ensemble des compositions possibles. La méthode de construction de réseau d'interaction est décrite précédemment dans le chapitre 4 (voir section 4.4.4, page 82). Puis, le système exploite la notion de pertinence, de densité des services et la notion de diversité dans les graphes pour optimiser le sous-graphe généré et sélectionner un ensemble de compositions optimales (voir la section 5.5.3). Enfin, notre système renvoie à l'utilisateur un ensemble de services web pertinents classés et leurs compositions.

5.4 Méthode de découverte et de classement de services web

Les systèmes de découverte de services visent à trouver les services similaires répondant aux besoins des utilisateurs en comparant les descriptions de services avec la requête de l'utilisateur. Notre mécanisme de découverte de services fonctionne en calculant la similarité entre la requête et les descriptions des services dans l'espace des thèmes. L'objectif principal de notre approche est d'accroître la diversité de tous les services découverts et minimiser la redondance dans les résultats de la recherche. En effet, de nombreuses approches existantes ne prennent pas en compte la redondance qui résulte des services web très similaires et dupliqués. La diversification des résultats de la recherche a été déja étudiée dans le contexte de la recherche des documents sur le web [Bache et al., 2013].

Dans cette section, nous présentons notre méthode de sélection et de classement de services web en se basant à la fois sur la pertinence (i.e. la similarité), la diversité et la densité des services. Les principales étapes de notre méthode sont décrites comme suit :

1. Découvrir la liste des thèmes implicites $\mathbf{T}$ liés à la requête utilisateur ;
2. Découvrir et sélectionner les services web liés aux thèmes sélectionnés ;
3. Classer la liste $\mathbf{S_q}$ de services web trouvés ;
4. Re-classer la liste $\mathbf{S_q}$ des services web classés précédemment et retourner une nouvelle liste classée $\mathbf{S_q^*}$.

Nous expliquons plus en détail, dans ce qui suit, la méthode proposée de découverte et de classement des services.

5.4.1 Méthode de découverte de services et mesure de pertinence

Après avoir formé le modèle probabiliste, la distribution des mots pour chaque thème extrait est connue et les services peuvent être décrits comme une distribution de thèmes.

Supposons que :

- $\theta^{(s)}$ est la distribution multinomiale sur les thèmes extraits pour un service s ;
- $\phi^{(t)}$ la distribution multinomiale sur les mots pour un thème t.

Soit $q = \{w_1, w_2, \ldots, w_n\}$ la requête d'un utilisateur qui contient un ensemble de mots w_i. Le processus global de notre système de découverte de services est divisé en deux grandes étapes : (1) la découverte des thèmes implicites liés à la requête et (2) la sélection et le classement des services correspondants. Dans ce qui suit, nous expliquons plus en détail chaque étape de notre méthode.

Découverte des thèmes : Dans un premier temps, nous calculons la vraisemblance de la requête q ; dénotée $\theta^{(q)}$; en la représentant sous forme de distribution multinomiale sur l'espace des thèmes. Ainsi, pour chaque thème t, nous calculons la similarité sémantique entre la requête q et le thème t en se basant sur la distribution ϕ_t comme définie dans l'équation 5.1.

$$\theta_t^{(q)} = P(q|t) = \prod_{w_i \in q} P(w_i|t) \tag{5.1}$$

Où $P(w_i|t)$ est la probabilité d'avoir un mot w_i étant donné un thème t.

Cette étape vise à sélectionner l'ensemble des thèmes potentiels implicites qui sont sémantiquement associés à la requête q. Cette ensemble est ensuite classée par ordre décroissant du score de similarité (i.e. $P(q|t)$). Les thèmes maximisant la probabilité $\theta_q^t = P(q|t)$ seront sélectionnés.

La découverte des thèmes liés à la requête q permet de découvrir facilement l'ensemble de services web associés à ces thèmes. Dans notre approche, les thèmes sont utilisés pour réduire la dimensionnalité de l'espace de recherche de services tout en capturant le maximum de relations sémantiques qui peuvent exister entre les requêtes et les services.

Sélection et classement des services : L'étape suivante est de calculer la similarité entre la requête q et les services associés à l'ensemble des thèmes $\mathbf{T}$ trouvés dans l'étape précédente. Pour cela, nous utilisons les probabilités générées θ et ϕ comme des critères de base pour calculer cette similarité. Nous utilisons la probabilité conditionnelle de la requête $P(q|s_i)$ où q est un ensemble de mots contenus dans la requête et s_i un service donné. Ainsi, cette probabilité $P(q|s_i)$ peut être calculée par l'équation 5.2.

$$P(q|s_i) = \prod_{w_k \in q} P(w_k|s_i) = \prod_{w_k \in q} \sum_{j=1}^{|T|} P(w_k|t_j) P(t_j|s_i) \tag{5.2}$$

Où $P(w_k|t_j)$ est la probabilité d'avoir le mot w_k étant donné le thème t_j et $P(t_j|s_i)$ est la probabilité que le thème t_j décrit le service s_i.

Dans le reste de ce chapitre, nous définissons la mesure de pertinence comme la similarité sémantique entre la requête de l'utilisateur et les services web découverts. Nous notons cette mesure par M_{rel} comme définie dans l'équation 5.3.

$$M_{rel}(q|s_i) = P(q|s_i) \tag{5.3}$$

Afin de sélectionner les services qui correspondent à la requête de l'utilisateur, nous classons ces services par ordre décroissant de leur score de pertinence M_{rel}. Par conséquent, le système de découverte renvoie les services qui maximisent la fonction M_{rel}.

5.4.2 Distance sémantique, diversité et densité

Dans cette section, nous allons décrire comment calculer la densité et la diversité dans notre contexte. La densité et la diversité sont considérées parmi les facteurs les plus importants dans le processus de sélection. Il existe de nombreuses mesures pour calculer la distance entre deux distributions θ_x et θ_y (i.e. dans notre cas ; θ_x et θ_y sont des distributions de probabilité sur les thèmes). Dans notre travail, nous considérons la divergence de *Kullback Leibler (KLD)* [Kullback & Leibler, 1951]. Nous définissons dans ce qui suit les trois mesures que nous utilisons dans ce chapitre.

Mesure de la distance sémantique : Nous utilisons la divergence de Kullback-Leibler pour mesurer le degré de la dissimilarité entre un service donné et tous les autres services sélectionnés. La divergence de *Kullback Leibler (KLD)* entre deux distributions de probabilité θ_x et θ_y, est définie dans l'équation 5.4 :

$$KLD(\theta_x, \theta_y) = \sum_{i=1}^{C_{xy}} \theta_x^i \log_2 \frac{\theta_x^i}{\theta_y^i} \tag{5.4}$$

Où $C_{xy} = |\theta_x^i| = |\theta_y^i|$ et $\theta_x^i = P(i \mid x)$ représente la probabilité du thème i étant donné le service x. La divergence KLD n'est pas symétrique et $KLD = 0$ si $\theta_x^i = \theta_y^i$ pour tout i.

Comme la divergence KLD n'est pas symétrique, nous utilisons la divergence KLD symétrisée, dénotée $SymmetricKLD$, pour mesurer la distance sémantique entre les services web. La mesure de distance sémantique $SymmetricKLD$ définie dans l'équation 5.5 obtient la symétrie en additionnant les deux divergences KLD entre deux distributions de probabilités.

$$SymmetricKLD(x, y) = \frac{1}{2}[KLD(\theta_x, \theta_y) + KLD(\theta_y, \theta_x)] \tag{5.5}$$

Mesure de diversité : Comme nous l'avons évoqué précédemment, la diversification des résultats de recherche a gagné en popularité dans la communauté de Recherche

d'Information. Elle peut être utilisée comme un critère secondaire pour compléter et classer les résultats des moteurs de recherche qui ne considèrent que l'information pertinente. Par conséquent, les utilisateurs peuvent recevoir trop de documents redondants. Par analogie, dans le cadre de la découverte des services web, si nous utilisons les services les mieux classés qui contiennent trop d'informations redondantes, le système risque de renvoyer à l'utilisateur trop de services redondants. Afin de diversifier les services web sélectionnés dans l'étape de découverte, nous calculons la mesure de diversité de chaque service candidat en respectant l'ensemble S_q^* contenant les services déjà sélectionnés. La mesure de diversité, dénotée M_{div}, est définie par l'équation 5.6 en se basant sur la distance sémantique introduite précédemment.

$$M_{div}(s_i, S_q^*) = \arg\max_{s_j \in S_q^*} SymmetricKLD(s_i, s_j) \tag{5.6}$$

Mesure de densité : Dans notre approche, nous mesurons la densité des premiers services initialement classés lors de l'étape de la découverte. Pour cela, nous approchons la densité d'un service web donné, en mesurant la distance moyenne entre ce service et les autres services candidats. En général, une moyenne élevée de la distance $SymmetricKLD$ indique que le service web se trouve dans une région de faible densité. Par conséquent, nous utilisons la moyenne négative de la distance $SymmetricK$L comme décrit dans l'équation 5.7, pour mesurer la performance approximative de la densité. Cela reflète la similarité sémantique de ce service par rapport aux autres services.

$$M_{dens}(s_i) = \frac{-1}{|S_q|} \sum_{s_j \in S_q} SymmetricKLD(s_i, s_j) \tag{5.7}$$

Où $|S_q|$ est le nombre de services web sélectionnés dans l'ensemble de résultats initialement trouvés.

Un service web dont la valeur de $M_{dens}(s)$ est plus élevée, peut être considéré comme similaire aux autres services dans S_q et ainsi plus représentatif et moins susceptible d'être extrême.

5.4.3 Méthode de diversification et de re-classement de services web

L'ensemble de services pertinents, retournés dans l'étape de découverte (voir section 5.4.1), contient essentiellement des services très similaires satisfaisant la requête de l'utilisateur. Cependant, le critère de classement appliqué dans ce processus de découverte des services web, considère uniquement l'information pertinente et ignore les services redondants qui peuvent être retournés. Comme nous l'avons déjà mentionné, nous avons

besoin de capturer la diversité des services web sélectionnés afin de minimiser la redondance dans l'ensemble de services découverts. Par conséquent, nous cherchons à accroître la diversité des services web retournés qui répondent à la requête de l'utilisateur en maximisant explicitement la distance entre les nouveaux services et les services déjà sélectionnés. La sélection des services web pertinents dont la densité est élevée permet

Algorithme 2 Algorithme de diversification des résultats de découverte de services web

Entrées :

- $S_q = \{s_1, \ldots, s_K\}$: l'ensemble de services classés découverts lors de l'étape de découverte (voir section 5.4), $K = |S_q|$ nombre de services web candidats.
- α : paramètre d'ajustement $\alpha \in [0, 1]$.
- K^* : nombre maximum de services web à retourner.

Sorties :

- S_q^* : ensemble de K^* services à renvoyer à l'utilisateur.

$S_q^* = \emptyset$, $Rel = \emptyset$, $Dens = \emptyset$
//Calculer la pertinence $Rel(s)$ et la densité $Dens(s)$ pour chaque service candidat s
Pour chaque service $s \in S_q = \{s_1, \ldots, s_K\}$ **Faire**
  $Rel(s) = GetRelevance(\theta_s, \theta_q)$ (voir équation 5.3)
  $Dens(s) = GetDensity(s)$ (voir équation 5.7)
Fin pour
Pour k de 0 à $K^* - 1$ **Faire**
  $maxValue = -1$;
  $s_{next} = null$;
  Pour chaque service $s_i \in S_q$ **Faire**
    $Div(s_i) = GetDiversity(s_i, S_q^*)$ //Calculer la diversité du service s_i (voir équation 5.6)
    $score = \alpha.Rel(s_i) + (1 - \alpha)(Dens(s_i) + Div(s_i))$ //Calculer le score du classement pour le service s_i (voir équation 5.8)
    Si $score > maxValue$ **Alors**
      $maxValue = score$;
      $selectedService = s_i$;
    Fin si
  Fin pour
  Si $s_{next} != null$ **Alors**
    $S_q^* = S_q^* \cup s_{next}$ //Ajouter le service s_{next} à la nouvelle liste S_q^*.
    $S_q.remove(s_{next})$ //Supprimer le service s_{next} de la liste S_q.
  Fin si
Fin pour
Retourner S_q^*.

de retrouver les services les plus pertinents lors du second tour du classement. Ainsi, nous

pouvons avoir de meilleures performances pour le processus de découverte et de sélection. Dans notre approche, l'algorithme de re-classement de services (voir algorithme 2) prend tout d'abord un ensemble de services mieux classés $S_q = \{s_1, \ldots, s_K\}$, découverts lors de l'étape de découverte (voir section 5.4.1). Ensuite, le système sélectionne itérativement les services web à partir de S_q pour former un ensemble représentatif de services sélectionnés S_q^* en considérant à la fois les mesures de pertinence, de diversité et de densité (lignes 4, 5 et 11). Plus précisément, chaque service candidat s_i est associé à un score RS du classement qui représente une combinaison linéaire de ces mesures, comme défini par l'équation 5.8 (ligne 12). Ainsi, nous obtenons automatiquement un classement efficace des services retrouvés.

$$RS = \arg\max_{s_i \in S_q} [\alpha . M_{rel}(q|s_i) + (1-\alpha)(M_{den}(s_i) + M_{div}(s_i, S_q^*))] \tag{5.8}$$

Où

- M_{rel} est la mesure de pertinence (i.e. la similarité sémantique entre la requête de l'utilisateur q et un service web sélectionné s_i, voir section 5.4.1, équation 5.3)
- $M_{dens}(s_i)$, représente la densité de chaque service web $s_i \in S_q$ (i.e. S_q représente les k premiers services web initialement classés, voir section 5.4.2, équation 5.7).
- $M_{div}(s_i, S_q^*)$ représente la diversité de chaque service web $s_i \in S_q$ (S_q^* représente la nouvelle liste produite, voir section 5.4.2, équation 5.6)
- $\alpha \in [0, 1]$ est un paramètre d'ajustement.

Pour réduire le nombre de calcul à faire lors du deuxième tour de sélection, nous sélectionnons K^* services web à partir de L premiers services classés dans liste S_q. Par exemple, les tailles raisonnables de L et K^* pourraient être 50 et 10 respectivement. Plus spécifiquement, le processus de sélection décrit dans l'équation 5.8 va être exécuté de manière itérative jusqu'à ce que la liste S_q^* contienne K^* services web à retourner après le second tour de sélection. Notre méthode de re-classement et de sélection des services peut donc être résumée comme suit :

- 1 : $S_q^* = \emptyset$
- 2 : **Répéter**
- 3 : Trouver un service s_{next} qui maximise le score RS (voir équation 5.8)
- 4 : $S_q^* = S_q^* \cup s_{next}$
- 5 : **Jusqu'à** $|S_q^*| = K^*$

5.5 Méthode de découverte de compositions possibles de services

5.5.1 Composition des services web

Comme nous l'avons évoqué précédemment, la composition de service web traite la situation où une requête de l'utilisateur ne peut pas être satisfaite par un service web atomique disponible. En effet, une séquence de services web peut répondre à une telle requête en combinant ou en composant plusieurs services web disponibles. Ces services doivent alors être intégrés pour créer des services composites à valeur ajoutée. Formellement, un service web s peut être défini comme $s = (s_{in}, s_{out})$ où $s_{in} = \{in_1, in_2, \ldots\}$ est l'ensemble de ses paramètres d'entrée et $s_{out} = \{out_1, out_2, \ldots\}$ représente ses paramètres de sortie. Nous supposons que l'ensemble des paramètres d'entrée s_{in} doit être fourni pour invoquer le service s qui à son tour doit retourner des paramètres de sortie s_{out}. Le problème de la composition de services web peut donc être défini comme une recherche d'un ensemble de services web $\{s_1, s_2, \ldots, s_n\}$ pouvant être composés d'une manière séquentielle ou parallèle. Par conséquent, la recherche de compositions de services peut être considérée comme une extension du problème de la découverte à un ensemble de services liés par des relations d'interaction.

D'une part, les problèmes liés à la composition sont en réalité bien plus complexes et difficiles à résoudre étant donné le nombre croissant de services web. D'autre part, la composition de service web est conçue pour permettre une collaboration et une communication entre les services dans le but de satisfaire des requêtes complexes qu'un seul service ne pourrait y répondre. Cette coordination entre les services exige également la prise en considération de leurs dépendances. Ainsi, le véritable défi est de faciliter le processus de la composition de services web dans le monde réel. Comme nous allons voir dans ce qui suit, la problématique de la composition de services web peut être résolue à travers des réseaux d'interaction (voir section 5.5.2). Une composition peut être donc formée d'un enchaînement de services reliés par leurs paramètres d'entrée/sortie.

Nous notons que nous ne nous produisons pas un service composite offrant une nouvelle fonctionnalité, créé à partir d'un ensemble de services web existants. Nous proposons dans cette thèse, une méthode de découverte de services susceptibles d'être composables avec les services pertinents retournés par notre système de découverte, décrit précédemment dans la section 5.4. Dans notre approche, notre système de découverte et de classement utilise les mesures de pertinence, de diversité et de densité des services afin de minimiser la redondance dans l'ensemble de services découverts. Nous expliquons plus en détail, dans la section 5.5.3, notre méthode de découverte de compositions de services.

5.5.2 Réseaux d'interaction et composition de services web

Les réseaux constituent un moyen adéquat pour représenter une collection de services web, et permettent une visualisation et une analyse en tirant parti des relations de similarité ou de dépendances entre eux. Ils peuvent être considérés comme des réseaux complexes et certains travaux de la littérature ont déjà utilisé cette approche pour étudier les réseaux du monde réel [Albert & Barabási, 2002]. Dans le contexte des services web, plusieurs travaux ont montré que les réseaux complexes ou les graphes orientés sont un moyen efficace pour traiter certains problèmes liés à la composition de services web [Oh et al., 2008, Mier et al., 2015].

Comme mentionné précédemment, le problème de composition peut être considéré comme un problème de recherche dans le réseau d'interaction de services web. Afin de modéliser les relations de dépendances et d'interaction entre un ensemble de services web sous forme de réseaux, les noeuds peuvent être définis à partir des trois niveaux de granularité (paramètres, opérations et services). Ces trois modèles sont décrits dans le chapitre 4. Nous rappelons que dans un réseau de paramètres, les noeuds représentent les paramètres de services web et les liens symbolisent les relations de dépendance entre les paramètres des opérations. Dans le réseau d'interaction d'opérations, les noeuds représentent l'ensemble des opérations et les liens traduisent les relations d'invocations et d'interactions entre les opérations. Les noeuds peuvent également représenter les services web dans un réseau d'interaction de services web, tandis que les liens matérialisent les dépendances et les interactions entre ces services (voir chapitre 4, section 4.4, page 78). Dans notre approche, nous construisons tout d'abord le réseau d'interaction d'opérations pour déduire ensuite le réseau d'interaction de services. Nous avons utilisé l'analyse de concepts formels pour organiser les opérations des services web en treillis de concepts formels en fonction de leur paramètres d'entrées et de sorties. Nous rappelons que dans notre approche, l'ACF prend comme entrée un ensemble d'opérations de services web et leurs paramètres d'entrée et de sorties représentés sous forme de contexte formel. Le résultat produit est un treillis de concepts formels (voir chapitre 4, section 4.4.4, page 82).

Nous avons définit précédemment deux modes d'invocation entre les services web, à savoir : l'invocation totale et l'invocation partielle. Dans le cas d'invocation totale, une dépendance est créée entre deux services s_1 et s_2, s'il existe au moins une opération de s_1 en invocation totale avec une opération de s_2. Dans le cas d'une invocation partielle, une dépendance est créée entre deux services s_1 et s_2, s'il existe au moins une opération de s_1 en invocation partielle avec une opération de s_2. Nous considérons dans notre approche deux mises en correspondance (i.e. totale et partielle) basées sur ces deux modes d'invocations. Dans l'exemple motivant décrit dans la section 5.2, l'opération $BookPublisher$ du service $WS1$ a comme entrée le paramètre B (i.e. $ISBN$) et comme sortie le paramètre F (i.e. $Publisher$). Cela est représenté par une arête entre ces deux paramètres dans le réseaux de

paramètres (voir figure 5.1). Le service $WS1$ produit l'ensemble des paramètres nécessaires pour invoquer les services $WS2$ (i.e. BookPublisher), $WS3$ (i.e. BookTaxedprice), $WS4$ (i.e. BookRecommendedprice) et $WS6$ (i.e. BookReaderreview). On parle dans ce cas de l'invocation totale entre les services. La figure 5.3 (Gauche) représente le réseau de services en mode d'invocation totale. Comme montré dans le réseau de services en mode d'invocation partielle (voir figure 5.3 (Droite)), les services $WS1$ et $WS2$ peuvent être en mode d'invocation partielle avec les services $WS5$ (i.e. BookPricesizebook-type), $WS6$ (i.e. BookReaderreview) et $WS7$ (i.e. BookAuthortext).

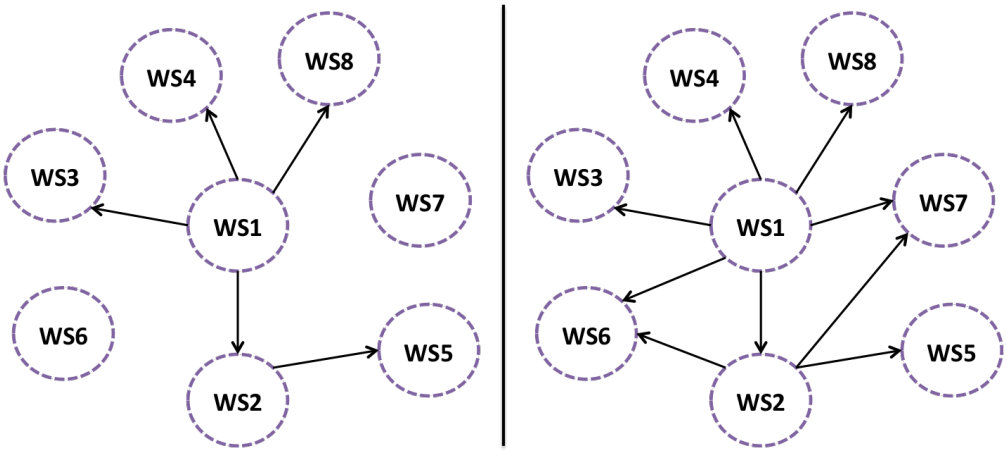


Figure 5.3 – Graphe de dépendance de services décrits dans l'exemple motivant. (Gauche) Graphe de dépendance en mode d'invocation totale. (Droite) Graphe de dépendance en mode d'invocation partielle.

5.5.3 Algorithme d'identification des compositions optimales des services

Nous décrivons dans cette section la méthode proposée pour la découverte des compositions optimales de services web. Notre objectif est de trouver des liens de compositions potentielles qui peuvent exister entre les différents services pertinents découverts dans l'étape de découverte (voir section 5.4) et/ou avec d'autres services web. Nous recherchons donc pour chaque service web candidat, l'ensemble de services web qui peut être composé avec lui en se basant sur les relations d'interaction existantes entre ces services. Il est important de noter que les services web composés ne sont pas nécessairement corrélés sémantiquement avec la requête de l'utilisateur. Par conséquent, le résultat de la composition permet d'enrichir les services pertinents découverts lors de l'étape de la découverte. Ces séquences de services peuvent répondent à certaines requêtes complexes qui ne peuvent pas être satisfaites par des services atomiques.

Avant de procéder à la sélection de ces compositions, nous extrayons tout d'abord le

sous-graphe de dépendances regroupant les services concernés et permettant de représenter la composition entre ces services. Le processus global de notre méthode d'identification des compositions de services est composé de trois étapes principales, comme décrit ci-dessous :

1. Découverte et sélection des services pertinents correspondant à la requête utilisateur (voir section 5.4) ;
2. Construction du sous-graphe de la composition.
3. Optimisation du sous-graphe et sélection des compositions optimales.

La première étape de la construction du graphe de composition est ainsi la découverte et la sélection des services pertinents qui répondent aux demandes des utilisateurs. Basé sur les premiers services divers classés selon l'ordre d'importance retournés par le système de découverte et de classement (voir section 5.4), un ensemble de services web découverts et leurs compositions peuvent être retournés en extrayant le sous-graphe de dépendance de service contenant toutes les compositions possibles. Le sous-graphe est construit selon le mode d'invocation partielle ou bien totale comme décrit précédemment. Une fois le sous-graphe construit, l'étape suivante consiste à réduire la taille du graphe afin d'améliorer la composition optimale.

Dans [Li & Yu, 2013], les auteurs ont proposé une mesure qui permet de classer et de diversifier les noeuds dans des graphes très larges. Cette mesure permet de capturer à la fois la pertinence et la diversité en se basant sur l'algorithme de base MMR proposé par [Carbonell & Goldstein, 1998]. De même, nous proposons dans notre contexte d'étendre la mesure de classement proposée dans [Li & Yu, 2013] en rajoutant un terme supplémentaire qui reflète la diversité et la densité des services web comme décrit dans la section 5.4. Dans ce qui suit, nous considérons les définitions suivantes :

Définition 5.1 ***Ensemble expanseur :*** *Soit S un sous-ensemble du graphe de dépendance de services web $G = (V, E)$, l'ensemble expanseur de l'ensemble S est définie par $N(S)$ telle que $N(S) = S \bigcup \{v \in (V \setminus S) | \exists u \in S, (u, v) \in E\}$.*

Définition 5.2 ***Taux d'expansion :*** *L'expansion d'un ensemble S est définie par $|N(S)|$ où $N(S)$ est l'ensemble expanseur introduit dans la définition 5.1 et $|N(S)|$ désigne son cardinal. Le taux d'expansion de l'ensemble S est le quotient $\sigma = \frac{|N(S)|}{|V|}$.*

Dans notre approche, nous proposons d'utiliser le taux d'expansion d'un graphe (voir définition 5.2) pour évaluer la diversité des services web dans le sous-graphe de dépendance généré pour les premiers services initialement classés par le système. Il est à noter que la définition du taux d'expansion est basée sur la structure topologique du graphe de services web. Le taux d'expansion est défini comme une mesure de connectivité d'un graphe. Intuitivement, plus le taux d'expansion d'un ensemble de noeuds est élevé, plus les noeuds sont dispersés dans le réseau, et entraînant ainsi une meilleure diversité

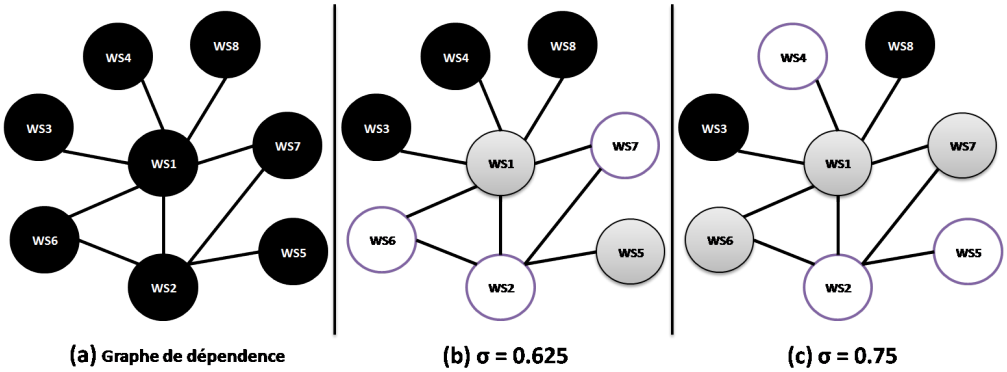


Figure 5.4 – Illustration du taux d'expansion et la diversité dans le graphe de dépendance des services web.

[Li & Yu, 2013]. Autrement dit, deux noeuds sont dissimilaires s'ils ne partagent pas le même voisinage dans le graphe de service web. Par conséquent, la définition du taux d'expansion peut être considérée comme une mesure de diversité.

Avec la définition 5.2, nous pouvons déduire qu'un ensemble de noeuds avec un grand taux d'expansion sont dissemblables les uns aux autres. Prenons par exemple le graphe de dépendance en mode d'invocation partielle de l'exemple motivant décrit précédemment (voir figure 5.4 (a)). Nous allons sélectionner trois noeuds de ce graphe. La figure 5.4 (b) et la figure 5.4 (c) sont deux cas différents, où les noeuds blancs représentent les noeuds sélectionnés. Il y a cinq noeuds (noeuds blancs et noeuds gris) dans la frontière de l'ensemble des noeuds sélectionnés pour le cas de la figure 5.4 (b), et six noeuds pour le cas de la figure 5.4 (c). On peut en déduire que le taux d'expansion des noeuds sélectionnés dans la figure 5.4 (b) et figure 5.4 (c) sont respectivement $\sigma = 0,626$ et $\sigma = 0,75$. Les noeuds sélectionnés dans la figure 5.4 (b) sont bien connectés, donc ils sont probablement semblables les uns aux autres. Au contraire, les noeuds sélectionnés sur la figure 5.4 (c) ne sont pas tous connectés. Par conséquent, les noeuds sélectionnés dans la figure 5.4 (c) sont plus divers que ceux de la figure 5.4 (b). Cet exemple montre que les noeuds avec un taux d'expansion plus grand ont une meilleure diversité.

Notre objectif consiste à trouver les k noeuds (dénotés par S) dans le sous-graphe de services web généré ayant les plus grands scores de classement RS (voir section 5.4.3, équation 5.8) et le plus grand taux d'expansion $\sigma = \frac{|N(S)|}{|V|}$. Formellement, nous cherchons à trouver les services web qui maximisent la mesure de sélection définie dans l'équation 5.9 :

$$H(S) = (1 - \lambda) \sum_{s \in S} RS(s) + \lambda \frac{|N(S)|}{|V|} \tag{5.9}$$

Où $RS(s)$ dénote le score du noeud s (i.e. équation 5.8), et $\lambda \in [0, 1]$ est un paramètre

d'ajustement. Le premier terme dans l'équation 5.9 est la somme des scores des noeuds sélectionnés, qui reflète la pertinence, la diversité et la densité des services sélectionnés (voir section 5.4.3, équation 5.8). Le deuxième terme représente le taux d'expansion des noeuds sélectionnés. En général, un plus grand taux d'expansion indique une meilleure diversité. Par conséquent, les Top-k services divers et optimaux dans le réseaux de services web sont ceux qui maximisent l'équation 5.9.

La mesure de sélection $H(S)$ (voir équation 5.9) que nous avons proposé pour la sélection des compositions de services ne tient compte que du voisinage immédiat d'un ensemble S. Naturellement, nous pouvons généraliser cette mesure de sélection en considérant les plus proches voisins. Nous introduisons donc une nouvelle mesure de sélection que nous appelons la mesure de sélection généralisée, que nous notons par $H_l(S)$. Cette mesure se base sur le voisinage d'ordre l d'un ensemble de services web. Dans ce qui suit, nous définissons le taux d'expansion et l'ensemble expanseur d'ordre l.

Définition 5.3 ***L'ensemble expanseur d'ordre l :*** *Soit S un sous-ensemble du graphe de dépendance de services web $G = (V, E)$, l'ensemble expanseur d'ordre l de l'ensemble S est définie par $N_l(S)$ telle que $N_l(S) = S \bigcup \{v \in (V \setminus S) | \exists u \in S, d(u, v) \leq l\}$. Où $d(u, v)$ désigne la longueur du chemin le plus court de u à v.*

Définition 5.4 ***Taux d'expansion d'ordre l :*** *L'expansion d'ordre l d'un ensemble S est la carinalité de l'ensemble expanseur $N_l(S)$ d'ordre l, notée par $|N_l(S)|$. Le taux d'expansion d'ordre l de l'ensemble S est défini comme le quotient $\sigma = \frac{|N_l(S)|}{|V|}$.*

En se basant sur les deux définitions 5.3 et 5.4, nous définissons une nouvelle mesure de sélection $H_l(S)$ comme suit :

$$H_l(S) = (1 - \lambda) \sum_{s \in S} RS(s) + \lambda \frac{|N_l(S)|}{|V|} \tag{5.10}$$

Nous remarquons donc que $H(S)$ est un cas particulier de $H_l(S)$ lorsque $l = 1$.

5.6 Expérimentation

Nous décrivons dans cette section les expérimentations réalisées et les résultats obtenus pour la méthode de découverte et de classement de services web proposée dans ce chapitre.

5.6.1 Collection de services web

Nos expérimentations sont réalisées sur des services web réels obtenus de la collection de test *SAWSDL-TC3*[5] décrite précédemment (voir chapitre 4, section 4.5.1, page 85). Nous

5. http://www.semwebcentral.org/projects/sawsdl-tc

rappelons que cette collection contient 1088 documents WSDL annotés sémantiquement appartenant aux neuf différents domaines d'applications (voir tableau 4.2, page 86). La collection de test *SAWSDL-TC3* contient aussi 42 requêtes avec une réponse pertinente contenant un ensemble de services pour chaque requête. Cela permet d'évaluer la précision des systèmes de découverte de services web. Le tableau 5.3 indique le nombre de requêtes utilisées pour chaque domaine. Nous utilisons l'ensemble de ces requêtes pour évaluer notre méthode de découverte proposée dans ce chapitre. Les 42 requêtes sont fournies aussi avec des réponses pertinentes contenant un ensemble de services pour chaque requête. La réponse pour chaque requête se compose d'un ensemble de services et à chaque service est associé une valeur de pertinence appartenant à l'ensemble des échelles de pertinence {0, 1, 2, 3} où 3 indique *Très Pertinent*, 2 *Pertinent*, 1 *Potentiellement Pertinent* et 0 *Non Pertinent*. Nous utilisons l'ensemble des 42 requêtes pour évaluer notre méthode de découverte et de classement de services.

#	**Domaine**	**Requêtes**
1	Communication	2
2	Economy	12
3	Education	6
4	Food	1
5	Geography	10
6	Medical	1
7	Military	1
8	Simulation	3
9	Travel	6
	Total	**42**

Table 5.3 – Nombre de requêtes utilisées pour chaque domaine

Avant d'appliquer le système de découverte de services proposé, nous traitons le corpus de services web (i.e. documents WSDL) en appliquant quelques techniques d'extraction d'information présentées dans le chapitre 2 (voir section 2.3.1, page 23). L'objectif de ce traitement est d'identifier les descriptions textuelles des services (i.e. des mots potentiels) qui décrivent la sémantique de leurs fonctionnalités. Le processus du traitement de documents WSDL se compose de plusieurs étapes : *analyse et extraction d'information, tokénisation, suppression des mots vides, lemmatisation, pondération des termes et construction de la matrice transactionnelle des services (i.e. matrice transactionnelle STM)*. Les données textuelles observées sont représentées dans la matrice *STM* que nous utilisons comme donnée d'entraînement pour construire le modèle thématique probabiliste CTM. Nous rappelons que pour construire notre modèle probabiliste, le nombre de thèmes doit être choisi avant la phase d'entraînement. Nous avons montré précédemment dans le chapitre 4 que l'on obtient une meilleure performance du modèle avec le nombre de

thèmes optimal $K = 90$ (voir section 4.5.2, page 86). Nous utilisons donc, pour ces expérimentations, le modèle généré avec 90 thèmes.

5.6.2 Mesures d'évaluation

Nous évaluons la précision de notre méthode de découverte en calculant deux mesures standards très utilisées dans le domaine de *la recherche d'information* : *Précision à n* (*Précision@n*) et le *Gain cumulatif réduit normalisé* (*NDCG*)[Rijsbergen, 1979, Manning et al., 2008, Järvelin, 2002]. Ces mesures permettent de mesurer la précision et la performance des systèmes de découverte.

Précision à n (précision@n) : La mesure de *précision@n* est utilisée dans notre contexte pour mesurer la précision de notre système de découverte et classement de services web en prenant en considération les n premiers services découverts [Rijsbergen, 1979, Manning et al., 2008]. La *précision@n* reflète ainsi le nombre de services qui sont pertinents à la requête de l'utilisateur (voir équation 5.11)

$$Precision@n = \frac{|RelevantServices \cap RetrievedServices|}{|RetrievedServices|} \tag{5.11}$$

Où $RelevantServices$ est la liste des services pertinents à une requête donnée (ces services sont définis dans la collection de test) alors que $RetrievedServices$ est la liste des n premiers services découverts par notre système.

Normalized Discounted Cumulative Gain : La mesure de *NDCG* (*Normalized Discounted Cumulative Gain*) est utilisée pour l'évaluation de la pertinence et le classement des services web renvoyés à l'utilisateur par notre système de découverte et classement. Cette mesure utilise des jugements de pertinence gradués. Les mesures graduées visent à pénaliser les systèmes classant un service non pertinent après un service pertinent, tout en favorisant les systèmes qui renvoient les services pertinents dans les tous premiers rangs. La mesure $NDCG_n$ pour n premiers services retournés est donnée par l'équation 5.12.

$$NDCG_n = \frac{DCG_n}{IDCG_n} \tag{5.12}$$

Où DCG_n est le Gain cumulatif réduit normalisé et $IDCG_n$ est le gain cumulatif réduit normalisé idéal. Le $IDCG_n$ se retrouve par le calcul du DCG_n des n premiers idéaux services pertinents à une requête donnée (les services qui sont définis dans la collection de test). Le DCG_n est calculé par l'équation 5.13.

$$DCG_n = \sum_{i=1}^{n} \frac{2^{relevance(i)} - 1}{log_2(1+i)} \tag{5.13}$$

Où n est le nombre de services récupérés et *relevance(i)* est la pertinence graduelle du service dans la *i-ème* position dans la liste classée.

Le calcul de la moyenne des valeurs $NDCG_n$ obtenues pour toutes les requêtes permet de mesurer la performance moyenne d'un algorithme de classement. $NDCG_n$ donne des scores plus élevés aux systèmes qui classent une liste de résultats de la recherche présentant la plus grande pertinence et pénalisent les systèmes qui renvoient des services peu adaptés. Les valeurs de $NDCG_n$ varient de 0 à 1.

5.6.3 Résultats et discussion

Nous avons évalué l'efficacité de notre méthode de découverte et de classement de services basée sur les mesures de pertinence, de diversité et de densité. Nous calculons ainsi la *Precision@n* et le $NDCG_n$ pour évaluer cette méthode que nous appelons dans le reste de ce chapitre *Topic-RDD*. Les exemples de requêtes (i.e. 42 requêtes) fournis sont tous sous la forme de documents SAWSDL contenant des exigences sémantiques ainsi qu'une description textuelle de la fonctionnalité demandée.

Notre méthode Topic-RDD a été comparée à trois méthodes de la littérature citées ci-dessous :

1. La méthode *MMR* (i.e. Maximal Marginal Relevance) proposée dans [Carbonell & Goldstein, 1998]. Cette méthode prend en compte la diversité dans les résultats de la recherche dans le contexte de la découverte de documents.
2. La méthode basée sur la syntaxe et implémentée par le matchmaker *Apache Lucene*[6].
3. La méthode hybride implémentée par le matchmaker *SAWSDL-MX2 Matchmaker*[7] pour les services SAWSDL [Klusch et al., 2009].

Dans nos expérimentations, la découverte est effectuée pour SAWSDL-MX2 en utilisant l'interface utilisateur de l'environnement d'évaluation SAWSDL-MX. Pour la méthode implémentée par Apache Lucene, les descriptions textuelles extraites des documents SAWSDL représentant les requêtes, sont utilisées en tant que chaîne d'interrogation du système. Pour notre méthode Topic-RDD et l'algorithme MMR, les 42 requêtes sont représentées dans un vecteur de thèmes en utilisant l'équation 5.1 et correspondent donc aux services web de ces thèmes selon le mécanisme décrit dans la section 5.4.1.

Avant de commencer à analyser les résultats d'évaluation obtenus pour chaque méthode, nous définissons tout d'abord la mesure MMR dans notre contexte comme décrit par l'équation 5.14.

$$MMR \stackrel{def}{=} \arg\max_{s_i \in R \setminus S} [\alpha . Sim(q|s_i) - (1-\alpha) \max_{s_i \in R} Sim(s_i, s_j)] \tag{5.14}$$

6. http://lucene.apache.org/
7. http://projects.semwebcentral.org/projects/sawsdl-mx

Où :

1. R représente l'ensemble de services web initialement classés.
2. S représente le sous-ensemble de services web de R ;
3. α est un paramètre d'ajustement.
4. $Sim(q, s) = P(q|s)$ désigne la mesure de similarité utilisée pour classer les services retournés par ordre de pertinence par rapport à la requête q (i.e. l'équation 5.2).

Alpha	Precision@5	Precision@10	Precision@15	Precision@20
0.00	0.748	0.750	0.726	0.710
0.10	0.748	0.750	0.726	0.709
0.25	0.748	0.750	0.726	0.710
0.50	0.748	0.750	0.729	0.710
0.75	0.748	0.752	0.731	0.709
0.90	0.748	0.750	0.726	0.708
1.00	0.724	0.714	0.699	0.690

Table 5.4 – Comparaison des valeurs de la précision moyenne obtenues, sur les 42 requêtes avec différentes valeurs du paramètre α, par notre méthode Topic-RDD.

Alpha	Precision@5	Precision@10	Precision@15	Precision@20
0.00	0.686	0.695	0.710	0.721
0.10	0.686	0.695	0.710	0.721
0.25	0.686	0.698	0.712	0.721
0.50	0.705	0.717	0.715	0.716
0.75	0.705	0.719	0.712	0.714
0.90	0.700	0.714	0.712	0.714
1.00	0.724	0.714	0.699	0.690

Table 5.5 – Comparaison des valeurs de la précision moyenne obtenues, sur les 42 requêtes avec différentes valeurs du paramètre α, par la méthode MMR.

Comme pour notre méthode, nous remarquons la présence d'un paramètre d'ajustement dans la mesure MMR. La première expérimentation consiste donc à analyser l'impact du paramètre $\alpha \in [0, 1]$ (i.e. équation 5.8) qui permet d'ajuster les valeurs de pertinence, de densité et de diversité. Les valeurs moyennes de $Precision@n$ et $NDCG@n$ sont obtenues pour les 42 requêtes pour notre méthode Topic-RDD et l'algorithme MMR tenant compte des différentes valeurs de α. Si $\alpha = 1$, le système utilise seulement la mesure de pertinence pour sélectionner les services web satisfaisant la requête de l'utilisateur ; si $\alpha = 0$, les services sélectionnés sont seulement basés sur les mesures de densité et de diversité. Les résultats sont présentés dans les tables suivantes :

– Comparaison des valeurs de la précision moyenne obtenues, sur les 42 requêtes avec différentes valeurs du paramètre α, par notre méthode Topic-RDD (voir table 5.4).

– Comparaison des valeurs de la précision moyenne obtenues, sur les 42 requêtes avec différentes valeurs du paramètre α, par la méthode MMR (voir table 5.5).
– Comparaison des valeurs de $NDCG_n$ moyen obtenues, sur les 42 requêtes avec différentes valeurs du paramètre α, par notre méthode Topic-RDD (voir table 5.6).
– Comparaison des valeurs de $NDCG_n$ moyen obtenues, sur les 42 requêtes avec différentes valeurs du paramètre α, par la méthode MMR (voir table 5.7).

Alpha	NDCG@5	NDCG@10	NDCG@15	NDCG@20
0.00	0.537	0.578	0.599	0.607
0.10	0.540	0.579	0.599	0.607
0.25	0.540	0.579	0.598	0.607
0.50	0.540	0.580	0.599	0.607
0.75	0.544	0.582	0.598	0.608
0.90	0.525	0.565	0.598	0.602
1.00	0.542	0.597	0.642	0.640

Table 5.6 – Comparaison des valeurs de $NDCG_n$ moyen obtenues, sur les 42 requêtes avec différentes valeurs du paramètre α, par notre méthode Topic-RDD.

Alpha	NDCG@5	NDCG@10	NDCG@15	NDCG@20
0.00	0.435	0.478	0.580	0.612
0.10	0.435	0.478	0.580	0.612
0.25	0.435	0.479	0.581	0.611
0.50	0.440	0.487	0.586	0.611
0.75	0.437	0.484	0.581	0.607
0.90	0.443	0.484	0.584	0.611
1.00	0.542	0.597	0.642	0.640

Table 5.7 – Comparaison des valeurs de $NDCG_n$ moyen obtenues, sur les 42 requêtes avec différentes valeurs du paramètre α, par la méthode MMR.

Les résultats présentés dans la table 5.4 montrent que notre méthode obtient une précision élevée quand les trois mesures (pertinence, diversité et densité) sont combinées pour sélectionner les services les plus pertinents correspondant à la requête de l'utilisateur. Nous observons également de ces résultats que notre méthode donne une plus grande précision si $\alpha = 0.75$ (i.e. precision@10 = 0.752). Les résultats de cette expérimentation montrent également que notre méthode Topic-RDD est plus performante en terme de précision par rapport à la méthode MMR pour toutes les valeurs de α et pour les 15 premiers services renvoyés. Comme décrit dans nos expériences, le paramètre α dans l'équation 5.8 nous permet d'ajuster la pertinence, la diversité et la densité. Le concept de la diversité et de la densité permet de diversifier correctement les résultats de la recherche tout en conservant les services pertinents pour une requête. En général, les utilisateurs

sont intéressés par les dix premiers services retournés par un moteur de recherche. Voilà pourquoi il est important d'évaluer seulement les n premiers services renvoyés par un système de découverte. Nous rappelons ici que les "services jugés concernés" fournis pour chaque requête de la collection de test SAWSDL-TC3 contiennent essentiellement des services très similaires. Cette collection ne prend pas en compte le concept de diversité. C'est pour cela que les valeurs de la moyenne de la précision commencent à baisser à partir de la $Precision@20$. Les résultats obtenus pour $NDCG$ montre aussi que notre méthode classe bien les services retournés par rapport à la méthode MMR (voir table 5.7). En effet, dans la recherche d'information, le $NDCG$ donne des scores plus élevés aux systèmes qui classent une liste de résultats de recherche ayant une valeur de pertinence élevée et pénalise les systèmes dont la pertinence est faible.

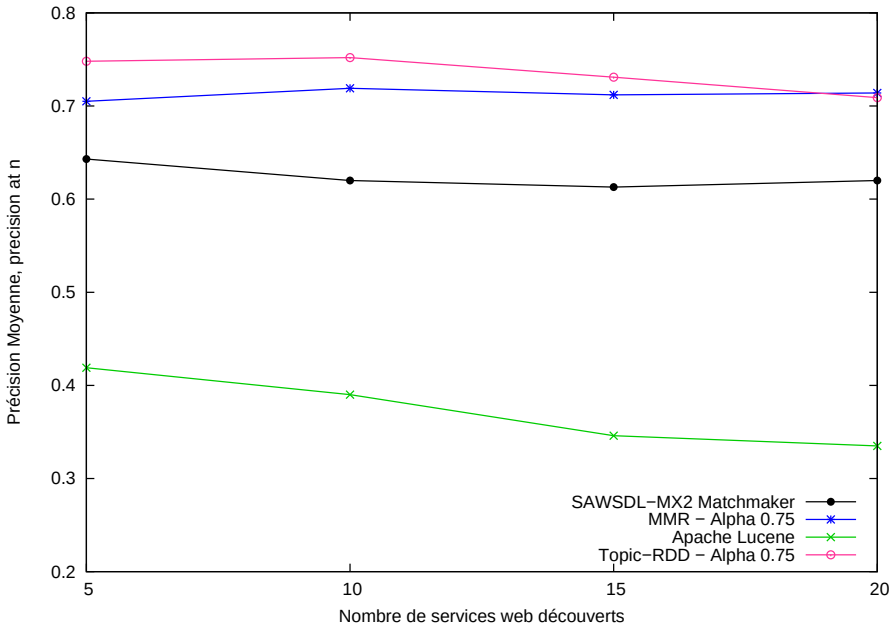


Figure 5.5 – Comparaison des valeurs moyennes de $Precision@n$ obtenues, sur les 42 requêtes, pour notre méthode Topic-RDD avec $\alpha = 0.75$ et les autres méthodes de la littérature.

Les valeurs moyennes de $Precision@n$ et $NDCG_n$ sont également obtenues sur les 42 requêtes pour les deux autres systèmes (i.e. *ApacheLucene*, *SAWSDL-MX2 Matchmaker*). Les résultats obtenus sont respectivement représentés dans les figures 5.5 et 5.6. Dans les deux cas, les résultats montrent que notre méthode Topic-RDD donne les plus grandes moyennes de $Precision@n$ et $NDCG_n$ pour les 42 requêtes. En effet, notre méthode est plus performante par rapport aux autres méthodes. Les résultats montrent que les deux systèmes ApacheLucene et SAWSDL-MX2 sont incapables de découvrir certains services

web pertinents qui ne sont pas directement liés à certaines requêtes via des mots-clés ou des descriptions logique. Cela reflète le fait que les services obtenus par notre méthode sont spécifiques à la requête de l'utilisateur. ApacheLucene et SAWSDL-MX2 obtient des valeurs faibles de $NDCG_n$ parce que, comme le montrent les résultats de $Precision@n$, les deux approches sont incapables de trouver des services hautement pertinents. Nous rappelons que notre méthode basée sur le modèle probabiliste à thèmes corrélés, exploite les informations pertinentes capturées par les thèmes pour découvrir les services. La méthode Topic-RDD combine les trois facteurs de sélection basés sur la pertinence ou la similarité, la densité et la diversité des services pour classer la liste finale à renvoyer à l'utilisateur.

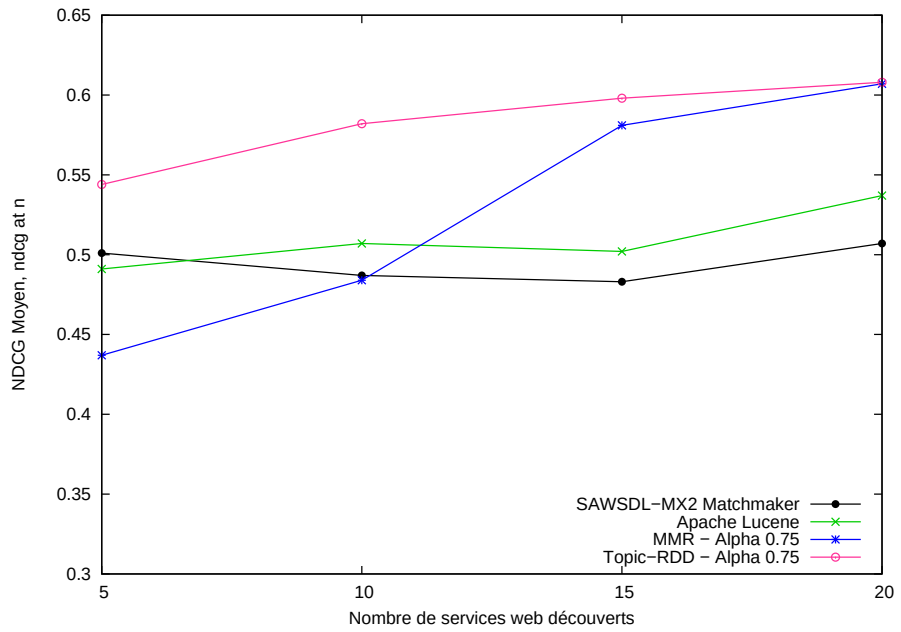


Figure 5.6 – Comparaison des valeurs moyennes de $NDCG_n$ obtenues, sur les 42 requêtes, pour notre méthode Topic-RDD avec $\alpha = 0.75$ et les autres méthodes de la littérature.

5.7 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle méthode de découverte et de composition des services web qui sont suceptibles d'être composable afin de minimiser la redondance et augmenter le niveau de satisfaction des utilisateurs. L'algorithme de diversification et de classement est proposé pour découvrir les k premiers services web en se basant sur les mesures de pertinence, de diversité et de densité des services. Dans

notre approche, nous avons utilisé le modèle probabiliste à thèmes corrélés pour extraire les thèmes à partir des descriptions sémantiques des services web. Les thèmes générés sont utilisés comme critère de base pour calculer un score global pour chaque service candidat en combinant la pertinence, la diversité et la densité de services. Notre approche permet également d'effectuer une recherche sur le réseau d'interaction de services web pour trouver un ensemble de compositions en générant un sous-graphe contenant toutes les compositions possibles de services web. Ensuite, un processus de sélection est effectué sur le sous-graphe pour identifier et sélectionner des compositions optimales avant de renvoyer le résultat final à l'utilisateur.

Des expérimentations ont été réalisées sur des services web réels pour valider notre méthode. Les résultats obtenus montrent que notre méthode donne une précision élevée si les trois mesures (i.e. pertinence, diversité et densité) sont combinées pour sélectionner les services les plus pertinents qui répondent à la requête de l'utilisateur. Nous avons comparé également la précision de la méthode proposée avec trois méthodes proposées dans la littérature. Les résultats obtenus montrent que l'on obtient des précisions plus élevées avec notre méthode par rapport aux systèmes de la littérature.

Dans le chapitre suivant, nous proposons un système de recommandation hybride basé sur le contenu et le filtrage collaboratif. Nous combinons les modèles thématiques probabilistes et les motifs fréquents pour capturer la sémantique commune maximale d'un ensemble de services. L'approche proposée combine également la similarité sémantique et un ensemble d'attributs de la qualité de services pour classer et sélectionner les différents services web recommandés.

6

Recommandation de services web basée sur l'extraction de motifs sémantiques

Sommaire

6.1 Introduction

L'explosion des services web ayant des fonctionnalités identiques ou similaires sur Internet, est devenue un problème gênant pour les utilisateurs finaux. Comment peuvent-ils sélectionner les meilleurs services à partir d'un ensemble de services ayant les mêmes fonctionnalités demandées ? Les systèmes de recommandation de services web permettent

de pallier ce problème en assistant la recherche de l'utilisateur et en l'orientant vers l'information pertinente qui répond à ses besoins.

Dans ce chapitre, nous proposons un système de recommandation hybride basé sur le contenu et le filtrage collaboratif basé sur la qualité de service (i.e. la réputation, temps de réponse, disponibilité, ...). L'originalité du système proposé vient de la combinaison des modèles thématiques probabilistes et des motifs fréquents pour capturer la sémantique commune maximale d'un ensemble de services. Notons qu'à notre connaissance, l'approche proposée constitue la première contribution qui combine ces deux domaines. Dans notre approche, nous combinons également la similarité sémantique et un ensemble d'attributs de la qualité de services pour classer et sélectionner les différents services web recommandés. Pour évaluer la méthode proposée, des expérimentations ont été réalisées sur un ensemble de services web réels. Nous avons comparé les performances de notre système avec deux systèmes de la littérature. Les résultats obtenus montrent que notre système obtient des valeurs de précision élevées et un temps de réponse moyen très faible.

Le reste de ce chapitre est organisé comme suit. La section 6.2 présente un aperçu général de l'approche proposée dans ce chapitre et décrit les différents modules de notre système. Dans la section 6.3, nous présentons en détail la méthode d'extraction des motifs sémantiques. Dans la section 6.4, nous décrivons la méthode de recommandation de services web proposée. Les différentes expérimentations et évaluations réalisées sont présentées dans la section 6.5. Enfin, nous concluons dans la section 6.6.

6.2 Description du système de recommandation proposé

Dans cette section, nous décrivons le système de recommandation de services web proposé dans ce chapitre. Nous présentons dans un premier temps un aperçu général du système. Ensuite, nous allons détailler les différentes étapes de notre approche qui repose sur les notions de thèmes et de motifs sémantiques.

6.2.1 Description de l'approche proposée

Un système de recommandation de services web a pour objectif de fournir à un utilisateur des services pertinents en fonction de ses préférences. Il est défini comme un processus d'identification automatique des services utiles à recommander aux utilisateurs finaux. Comme nous l'avons décrit dans le chapitre 3 (voir section 3.3, page 54), de nombreux travaux de recherche sur la recommandation ont été récemment proposés et se concentrent sur deux approches : le filtrage collaboratif et la recommandation basée sur le contenu. Les approches de filtrage collaboratif sont presque utilisées dans tous les systèmes de recommandation. Ces approches permettent de sélectionner les services pertinents pour l'utilisateur courant en collectant des informations auprès d'autres utilisateurs similaires. Les approches basées sur le contenu recommandent les services web en se

basant sur la similarité entre les requêtes de l'utilisateur et la description des services web (i.e. fonctionnalités du service). Dans ce chapitre, nous proposons une nouvelle approche hybride qui combine ces deux approches (le filtrage collaboratif et l'approche basée sur le contenu). Notre approche exploite les avantages de ces deux techniques qui permettent des recommandations plus précises avec une variété riche.

L'objectif principal de notre système est d'identifier un ensemble de services web qui sont très sémantiquement liés par rapport à un service donné et ayant de meilleure qualité de services. Pour cela, nous avons introduit la notion de motifs sémantiques (voir définition 6.1). Ces motifs sémantiques correspondent à des motifs fréquents maximaux de thèmes qui sont utilisés ensuite pour extraire un ensemble de motifs de services (voir sections 6.3.2 et 6.3.3). Les thèmes sont des concepts introduits par les modèles thématiques probabilistes. Les modèles thématiques probabilistes représentent une famille de modèles graphiques probabilistes génératifs basés sur l'hypothèse que les documents sont générés par un mélange de thèmes qui sont des distributions de probabilité sur des termes (voir chapitre 2, section 2.3.2, page 27). Dans notre contexte, les modèles thématiques probabilistes sont utilisés comme techniques efficaces de réduction des dimensions en capturant des relations sémantiques entre mots-thèmes et thèmes-services, interprétés sous forme de distributions de probabilités. Dans notre approche, l'utilisation des thèmes produits par le modèle thématique probabiliste permet essentiellement de :

1. Réduire la dimensionnalité de l'espace du contexte d'extraction de motifs à partir d'une grande collection de services web.
2. Capturer des relations sémantiques entre thèmes-mots et services-thèmes. Par conséquent, capturer la sémantique commune maximale d'un ensemble de services.

La découverte de motifs fréquents maximaux calcule les ensembles maximaux d'items (à savoir, les thèmes), selon l'inclusion, qui apparaissent ensemble dans au moins un certain nombre de transactions (à savoir, les services) présentes dans un contexte d'extraction. Les motifs sémantiques permettent de regrouper les services qui sont similaires. Ces motifs sémantiques sont utilisés pour capturer la sémantique commune maximale entre un ensemble de services. Les services d'un motif sémantique sont très intéressants : ils sont sémantiquement liés et maximaux. Les services correspondant aux motifs sémantiques sont utilisés par le système pour les recommander à l'utilisateur après les avoir classés par ordre de pertinence. Afin de calculer les motifs sémantiques et les ensembles correspondants de services, nous avons construit le treillis de concepts fréquents [Zaki & Hsiao, 2005]. Ces ensembles de services sont ensuite stockés dans une structure spéciale, appelée MFI-tree [Grahne & Zhu, 2005], afin d'effectuer des recherches rapides par le système de recommandation. A partir d'un service spécifié, le système de recommandation utilise cet arbre pour trouver les services qui lui sont sémantiquement similaires. Les services obtenus sont ensuite classés et recommandés à l'utilisateur. Afin de classer les services web concernés, nous avons introduit une mesure de pertinence M_{rank} permettant de

mesurer le degré de pertinence de ces services par rapport au service demandé. Dans notre approche, nous combinons la similarité sémantique (basée sur la distribution de probabilité sur les thèmes) et un ensemble de propriétés non-fonctionnelles (la qualité de services - QoS) pour calculer la mesure M_{rank}. La QoS joue un rôle important dans les systèmes de recommandation, dans lesquels un ensemble de services similaires peuvent être classés et sélectionnés pour les utilisateurs. Dans notre contexte, nous considérons un ensemble d'attributs de la qualité de services, tels que la réputation (i.e. annotation numérique), la disponibilité et le temps de réponse (voir section 6.4).

6.2.2 Architecture générale du système proposé et ces différents modules

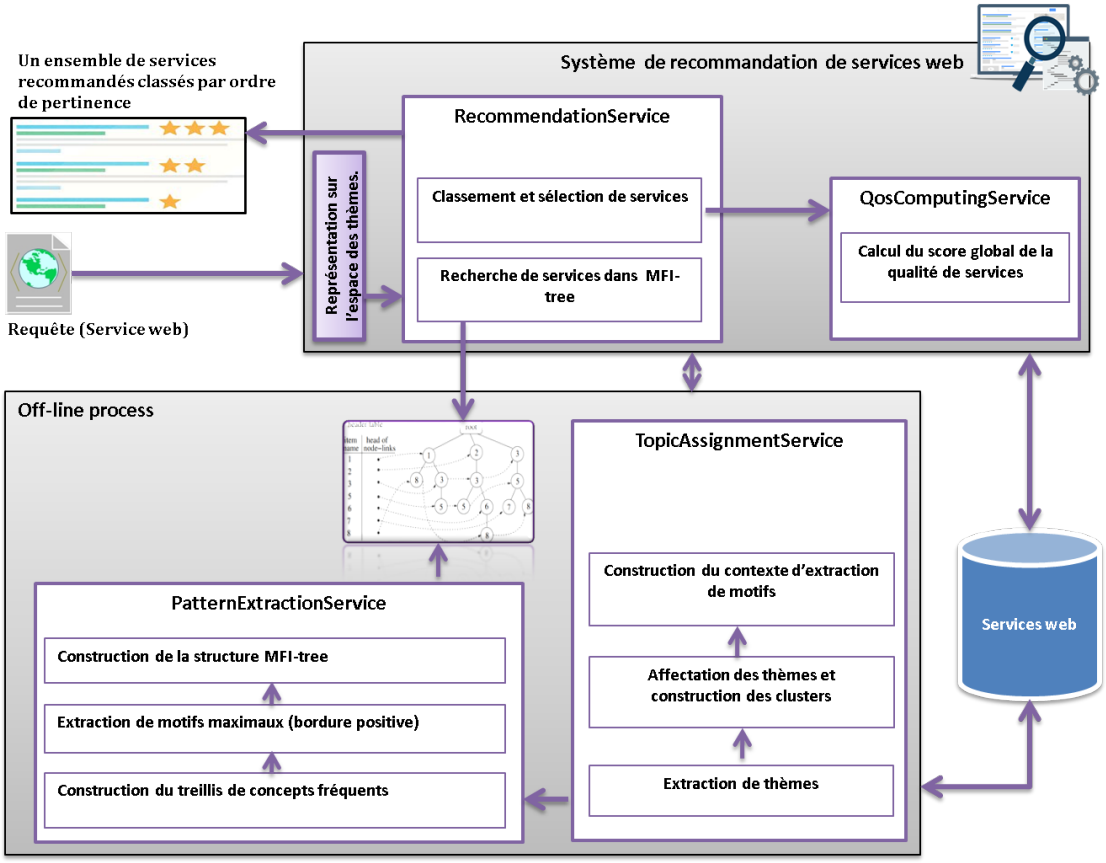


Figure 6.1 – Un aperçu du système de recommandation de services proposé.

La figure 6.1 montre un aperçu général du système de recommandation proposé. Comme le montre cette figure, notre système se compose de quatres modules ou services principaux ci-dessous :

1. **TopicAssignmentService :** ce service permet l'affectation des thèmes et la construction du contexte d'extraction. Il se base sur le principe du clustering pour affecter un ensemble de thèmes à chaque service web référencé dans notre référentiel de services. Ces services sont utilisés pour construire le contexte d'extraction nécessaire à l'extraction de motifs sémantiques (voir section 6.3.1).

2. **PatternExtractionService :** ce service s'occupe de l'extraction de motifs de services. Ces motifs de services sont utilisés pour construire l'espace de recherche utilisé lors de la tâche de recommandation de services. Pour construire cet espace de recherche un ensemble de tâches doivent être préalablement exécutées, à savoir : (1) construction de treillis de concepts fréquents, (2) extraction de la bordure positive (i.e. motifs sémantiques), (3) extraction de motifs de services et (4) construction de la structure MFI-tree (voir sections 6.3.2 et 6.3.3)

3. **QosComputingService :** ce service permet de calculer le score global de la qualité de services en se basant sur les différents attributs de la Qos considérés. Il permet de calculer le score global $OverallQosScore$ de la qualité de service pour chaque service résultat (voir équation 6.7).

4. **RecommendationService :** ce service s'occupe de la tâche de recommandation de services web. Une fois les différentes tâches décrites ci-dessus achevées, nous pouvons facilement recommander des services web à partir d'un service sélectionné par l'utilisateur dans la liste des services retournés par notre système de découverte décrit dans le chapitre 5. Plus précisément, notre système de recommandation de services utilise l'espace de recherche construit précédemment pour découvrir et classer un ensemble de services web pertinents correspondant au service demandé (voir section 6.4). Le service *RecommendationService* contacte le service *QosComputingService* pour récuperer le score global de QoS pour chaque service web recommandé. Notons que c'est le seul service qui s'exécute en ligne pour accomplir la tâche de recommandation de services web.

6.3 Extraction de motifs sémantiques de services web

Nous présentons dans cette section, la tâche d'extraction de motifs dans le contexte de services web. Nous commençons tout d'abord par la construction du contexte d'extraction en se basant sur l'ensemble des thèmes extraits à partir des descriptions de services web. Ensuite, nous décrivons la notion de motifs sémantiques que nous proposons avant de décrire comment extraire les motifs de services web. Nous rappelons que la section 2.5 (voir chapitre 2, page 38) introduit les définitions et les notions de base liées à l'extraction de motifs fréquents.

6.3.1 Affectation des thèmes et contexte d'extraction

La première étape du processus d'extraction de motifs consiste à construire le contexte d'extraction. Dans notre approche, nous utilisons la représentation thématique des services web pour construire le contexte d'extraction de motifs. Pour cela, nous nous basons sur les thèmes extraits à partir des descriptions de services web. Nous rappelons que nous utilisons le modèle thématique probabiliste à thèmes corrélés **CTM** [Blei & Lafferty, 2007] pour extraire les thèmes à partir des descriptions de services (voir chapitre 2, section 2.3.3, page 29). Après avoir entrainé le modèle probabiliste CTM, la distribution des mots pour chaque thème est connue et tous les services dans le jeu de données peuvent être décrits comme une distribution de thèmes (i.e. $\overline{s} = \{z_1, z_2, ..., z_K\}$) où chaque dimension z_k reflète la probabilité que le service s appartienne au thème k. Chaque thème extrait est associé à un groupe de concepts textuels (i.e. mots) et/ou des concepts sémantiques qui apparaissent dans les descriptions de services web. Les thèmes sont exprimés sous forme de distributions de probabilités sur les mots.

Algorithme 3 Algorithme d'affectation de services web à un ensemble de clusters

Entrées : – $S = \{s_1, \ldots, s_M\}$ représente l'ensemble de services web (M le nombre de services).
– T le nombre de thèmes.
– $nbAssign$: le nombre de thèmes à assigner à chaque service.

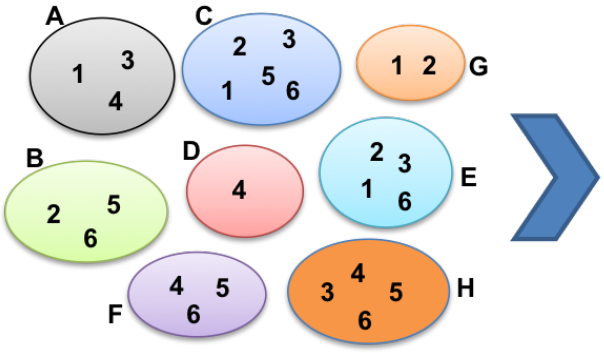
Sorties : K Clusters de services.

1: Appliquer le modéle thématique probabiliste sur l'ensemble de services web $S = \{s_1, \ldots, s_M\}$.
2: $Z = ExtractTopics(S, T)$
3: **Pour** chaque théme $t \in Z = \{t_1, \ldots, t_K\}$ **Faire**
4: $C_t = \emptyset$;
5: **Fin pour**
6: **Pour** chaque service $s_i \in S = \{s_1, \ldots, s_M\}$ **Faire**
7: $topicsMap = \emptyset$;
8: **Pour** Pour chaque théme $t \in Z = t_1, \ldots, t_K\}$ **Faire**
9: Calculer $p = P(s_i|t)$;
10: $topicsMap.put(t, p)$;
11: **Fin pour**
12: $SortMap(topicsMap)$;
13: $nbAssign(s_i, topicsMap, nbAssign)$;
14: $C_k = C_k \cup \{s_i\}$;
15: **Fin pour**
16: **Retourner** $\{C_j : j = 1, \ldots, T\}$;
17: **Retourner** l'ensemble de clusters T.

Supposons que $\theta^{(s)}$ est la distribution multinomiale sur les thèmes pour un service

s, et $\phi^{(j)}$ la distribution multinomiale sur les mots pour un thème j. Nous exploitons le mécanisme du regroupement de services (i.e. clustering) pour définir l'ensemble des relations binaires du contexte d'extraction qui peuvent exister entre les services (i.e. transactions) et les thèmes (i.e. attributs). Pour cela, nous construisons K clusters ; où K est le nombre de thèmes générés (i.e. un cluster pour chaque thème). La distribution multinomiale sur les thèmes $\theta^{(s)}$ pour le service s est utilisée pour déterminer quel thème k correspond le mieux au service s. Plus précisément, si la distribution de probabilité $\theta^{(s)}$ à travers z_j pour un service s est élevée, le service s est ensuite assigné au cluster C_j correspondant à ce thème. Si un service est relié à plusieurs thèmes, il sera donc assigné à chacun des clusters qui correspondent à ces thèmes [Aznag et al., 2014]. Pour simplifier et éviter la problématique du seuil à choisir, nous utilisons la stratégie de l'*affectation multiple* pour attribuer un ensemble de thèmes à chaque service web en sélectionnant les K premiers thèmes qui représentent le mieux un service donné. Ainsi, un service peut être affecté à plusieurs clusters (par exemple, les clusters associés aux trois meilleurs thèmes). Cela permet d'augmenter la portée de chaque cluster. L'algorithme 3 présente l'ensemble des étapes permettant l'affectation des services web à un ensemble de clusters.

Nous nous basons sur les clusters de services identifiés pour construire le contexte d'extraction dénoté $\mathcal{D} = (\mathcal{O}, \mathcal{A}, \mathcal{R})$ où $\mathcal{O}$ est un ensemble de transactions (i.e., services web), $\mathcal{A}$ représente l'ensemble d'attributs (i.e. thèmes), et $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{A}$ est une relation reliant les transactions et les attributs. Chaque couple $(s, t) \in \mathcal{R}$ exprime le fait que le service s est en relation avec le thème t. Prenons par exemple, 6 services web regroupés sur 8 clusters différents (i.e. $A = \{1, 3, 4\}$, $B = \{2, 5, 6\}$, $C = \{1, 2, 3, 5, 6\}$, $D = \{4\}$, $E = \{1, 2, 3, 6\}$, $F = \{4, 5, 6\}$, $G = \{1, 2\}$, $H = \{3, 4, 5, 6\}$). La table de la figure 6.2 (Droite) montre le contexte d'extraction représentant les relations entre les 6 services et les 8 clusters. Dans cet example, le service 1 est lié aux thèmes A, C, E et G.



	A	B	C	D	E	F	G	H
1	x		x		x		x	
2		x	x		x		x	
3	x		x		x			x
4	x			x		x		x
5		x	x			x		x
6		x	x		x	x		x

Figure 6.2 – (Gauche) Exemple de clusters de services. (Droite) Contexte d'extraction associé à l'ensemble de services.

6.3.2 Extraction de motifs sémantiques

Nous présontons dans cette section, la seconde étape du processus de l'extraction de motifs dans notre contexte. Nous introduisons dans un premier temps la notion de motifs sémantiques. Soit $\mathcal{D} = (\mathcal{O}, \mathcal{A}, R)$ un contexte formel. Un *motif d'attributs* est défini comme un sous ensemble d'attributs $\mathcal{A}$ (i.e. thèmes) et un *motif d'objets* est défini comme un sous ensemble de transactions $\mathcal{O}$ (i.e. services). Nous rappelons qu'une transaction t supporte un motif d'attributs X si $\forall i \in X, (t, i) \in \mathcal{R}$. Un motif d'attributs X est ***fréquent*** si le nombre de transactions qui le supportent, est supérieur (ou égal) à une valeur de seuil minimale ; dénoté *minsup* (voir définition 2.20, page 39). L'ensemble de tous les motifs d'attributs fréquents est $Th(\mathcal{L}_{\mathcal{A}}, \mathcal{D}, q) = \{X \subseteq \mathcal{A},\ |s \in \mathcal{O},\ \forall t \in X\ \ (s, t) \in \mathcal{R}\}| \ \geq minsup\}$.

Comme nous l'avons mentionné précédemment, nous utilisons dans notre approche la représentation condensée par motifs fréquents fermés et maximaux. L'ensemble de tous les ***motifs d'attributs fréquents maximaux*** (MFI pour Maximal Frequent Itemsets), en respectant l'ensemble d'inclusion, dans le contexte d'extraction $\mathcal{D}$ est la ***bordure positive*** de $Th(\mathcal{L}_{\mathcal{A}}, \mathcal{D}, q)$ (voir définition 2.24, page 43), dénotée $Bd^{+}(Th(\mathcal{L}_{\mathcal{A}}, \mathcal{D}, q))$, et égale à $\{X \in Th(\mathcal{L}_{\mathcal{A}}, \mathcal{D}, q) \mid \forall Y \supset X,\ Y \notin Th(\mathcal{L}_{\mathcal{A}}, \mathcal{D}, q)\}$ [Mannila & Toivonen, 1997]. Considérons l'exemple représenté dans la figure 6.2 (Droite) :

- Si $minsup = 2$ alors le motif d'attributs H est fréquent car il est supporté par 4 transactions (3, 4, 5 et 6).
- BG n'est pas fréquent parce qu'il est supporté seulement par 2.
- CE est fréquent mais non maximal car CEH est aussi fréquent.

L'ensemble des motifs d'attributs fréquents maximaux est la bordure positive $Bd^{+}(Th(\mathcal{L}_{\mathcal{A}}, \mathcal{D}, q))$ et est égale à $\{AH,\ ACE,\ BCE,\ CEG,\ CEH,\ BCFH\}$.

Nous introduisons dans la définition 6.1 la notion de motifs sémantique. Pour chaque motif sémantique, les transactions (i.e., services) contenant ce motif peuvent être associées. Remarquons que les services associés à un motif sémantique sont trés intéressants : ils sont sémantiquement liés et maximaux. Ainsi, ces services sont utilisés par le système de recommandation proposé dans ce chapitre. Le seuil minimum du support *minsup* permet de fixer le nombre minimum de services pour chaque motif sémantique.

Définition 6.1 (Motif sémantique) *Étant donné un seuil minimum de fréquence minsup, un* ***motif sémantique*** *est défini comme un motif d'attributs fréquent maximal de thèmes.*

Pour extraire les motifs sémantiques et les services associés, nous calculons les concepts formels fréquents. Comme le montre la définition 2.25 (voir page 44), nous nous basons sur la notion de *fermetures de Galois* h et h' (respectivement sur $2^{\mathcal{A}}$ et $2^{\mathcal{O}}$ (voir définition 2.10, page 35) pour définir la notion de motifs fermés. Soit X un motif d'attributs, si $h(X)$=X, alors X est un *motif d'attributs fermé*. En effet, un concept formel est composé d'un motif d'attributs fermé et d'un ensemble de transactions contenant ce motif

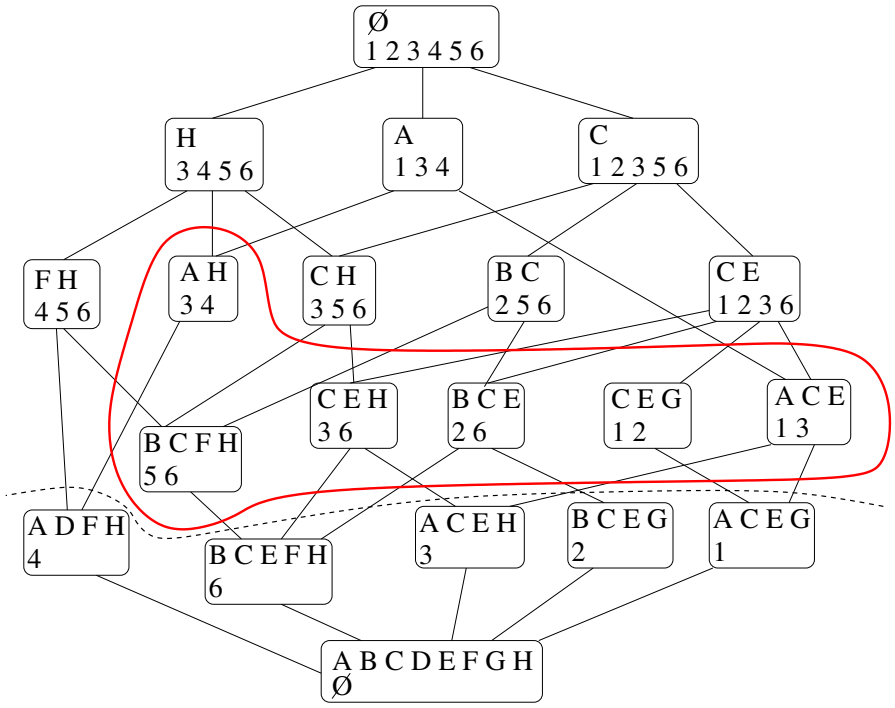


Figure 6.3 – Treillis de concept associé au contexte d'extraction présenté dans la figure 6.2 (la bordure positive Bd^+ est encerclée pour le support $minsup$=2)

d'attributs fermé. Un ***treillis de concept fréquent*** est formé en utilisant les concepts formels ayant au moins $minsup$ transactions dans leur extension. Le concept infimum (i.e., $(\mathcal{A}, \emptyset)$) est conservé. Vu que les intentions des concepts formels fréquents forment l'ensemble de tous les motifs fréquents fermés [Pasquier et al., 1999] et que l'ensemble de tous les motifs d'attributs fréquents maximaux est un sous-ensemble de motifs fréquents fermés, nous pouvons donc trouver facilement $Bd^+(Th(\mathcal{L}_\mathcal{A}, \mathcal{D}, q))$ (i.e. l'ensemble de motifs sémantiques) à partir du treillis de concepts fréquents. La bordure positive correspond aux concepts formels fréquents juste au-dessus de l'infimum. La figure 6.3 représente le treillis de concept obtenu en utilisant l'exemple représenté dans la figure 6.2 (Droite). L'infimum est $(A\ B\ C\ D\ E\ F\ G\ H, \emptyset)$. Avec $minsup$=2, les concepts formels fréquents sont au-dessus de la ligne en pointillé. Les concepts formels correspondant à la bordure $Bd^+(Th(\mathcal{L}_\mathcal{A}, \mathcal{D}, q))$ sont encerclés. Par conséquent, les motifs sémantiques sont $\{AH, ACE, BCE, CEG, CEH, BCFH\}$. Remarquons que les concepts de la bordure positive $Bd^+(Th(\mathcal{L}_\mathcal{A}, \mathcal{D}, q))$ peuvent avoir plus de $minsup$ transactions dans leur extension.

6.3.3 Extraction et stockage de motifs de services

Notre processus d'extraction de motifs produit les concepts formels correspondants à la bordure positive $Bd^+(Th(\mathcal{L}_\mathcal{A}, \mathcal{D}, q))$ (i.e. l'ensemble des motifs sémantiques). Chaque

motif sémantique représente un ensemble maximal de thèmes partagés par un ensemble de services. Cet ensemble de services correspond à l'extension du concept formel correspondant. Ces services sont donc implicitement associés à un motif sémantique. Nous sélectionnons dans cette étape, les extensions des concepts formels de la bordure positive $Bd^{+}(Th(\mathcal{L}_{\mathcal{A}}, \mathcal{D}, q))$ pour former les ensembles de services web qui vont être utilisés par notre système de recommandation de services. Dans l'exemple 6.3, les ensembles de services associés aux différents motifs sémantiques extraits sont : $\{\{3,4\}\{1,3\}\{1,2\}\{2,6\}\{3,6\}\{5,6\}\}$. Prenons par exemple le motif sémantique $BCFH$, les services qui lui sont associés sont 5 et 6 car ces deux services partagent les mêmes thèmes B, C, F et H (voir le contexte d'extraction de la figure 6.2). Remarquons qu'il n'y a pas d'autres thèmes en commun, et il n'y a pas d'autres services correspondants aux thèmes B, C, F et H : ceci est une relation maximale. Ainsi, les services 5 et 6 sont fortement et sémantiquement liés (en raison des thèmes B, C, F et H). Par conséquent, il est intéressant d'utiliser ce résultat pour la recommandation de services web. En effet, si l'utilisateur est intéressé par le service web 5 (resp. service 6), il y a une forte probabilité qu'il sera également intéressé par le service 6 (resp. service 5).

Dans notre approche, nous considérons les ensembles de services web identifiés comme des motifs de services (i.e. motif d'objets). Afin de faciliter la tâche de la recommandation, nous stockons ces motifs de services dans une variante de FP-tree (i.e. Frequent Pattern tree ; l'arbre de motifs fréquents) appelée arbre MFI-tree (Maximal Frequent Itemset tree ; l'arbre de motifs fréquents maximaux) [Grahne & Zhu, 2005]. Cela nous permet un gain d'espace et une recherche rapide des motifs contenant un service donné en utilisant des index. Chaque branche de l'arbre représente un motif. La compression est réalisée en construisant l'arbre de telle sorte que les motifs partagent les préfixes de la branche correspondante. Les noeuds fils de la racine constituent des sous-arbres d'objets. Chaque noeud de ces sous-arbres est composé de :

- Un objet
- Une liste des noeuds fils.
- Un lien vers le noeud parent.
- Un lien inter-noeud vers un autre noeud ayant le même objet.

L'avantage de cette structure de données est, qu'en suivant les liens inter-noeuds, on peut facilement connaître tous les motifs où figure l'objet. Tous les noeuds ayant le même objet, sont reliés entre eux. Le lien inter-noeud pointe sur le noeud suivant dont le nom d'élément est le même. Une table d'entête (i.e. header table) est construite pour les éléments dans l'arbre MFI-tree. Chaque entrée de la table d'entête se compose de deux champs ; le nom de l'objet (i.e. item name) et l'entête d'un lien inter-noeud (i.e. head of node-links). Le lien inter-noeud pointe sur le premier noeud ayant le même nom d'élément dans l'arbre.

Pour des raisons pratiques et par souci de clarté et de simplification, nous utilisons

un autre exemple (différent de l'exemple de la figure 6.3) pour illustrer la construction de la structure MFI-tree. La taille de l'exemple présenté dans la figure 6.2 est adaptée pour obtenir un treillis de concepts facile à lire et à comprendre. Mais, les ensembles obtenus de transactions sont trop petits pour illustrer la construction de l'arbre MFI-tree (tous ces ensembles contiennent deux transactions : $\{3,4\}$, $\{5,6\}$, $\{3,6\}$, $\{2,6\}$, $\{1,2\}$, $\{1,3\}$). Par conséquent, nous considérons un deuxième exemple, plus complet que le premier, dans lequel nous avons extrait directement les motifs sémantiques. La table 6.1 présente les ensembles de motifs de services web associés aux motifs sémantiques extraits. Notons que ce deuxième exemple est trop grand pour construire un treillis de concepts lisible.

Identifiant du motif	**Motifs de services**
I01	$\{1,8\}$
I02	$\{1,3,5\}$
I03	$\{2,3,5\}$
I04	$\{3,5,7\}$
I05	$\{3,5,8\}$
I06	$\{2,3,6,8\}$

Table 6.1 – Les ensembles de motifs de services utilisés pour construire l'arbre MFI-tree sont illustrés dans la figure 6.4

La figure 6.4 illustre la construction de l'arbre pour les deux premiers motifs. Le premier motif est $\{1,8\}$. Il est inséré directement dans l'arbre (Figure 6.4 (a)). Nous insérons ensuite $\{1,3,5\}$ dans l'arbre (Figure 6.4 (b)). La figure 6.4 (c) représente l'arbre complet pour cet exemple.

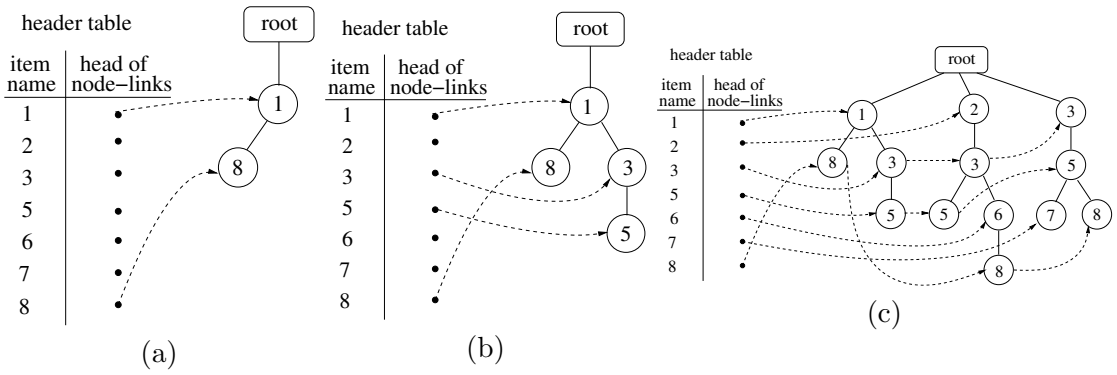


Figure 6.4 – Construction de la structure de MFI-tree

6.4 Recommandation de services web basée sur l'extraction de motifs

Nous présentons dans cette section l'algorithme de recommandation de services web proposé dans ce chapitre. Nous nous basons sur les motifs sémantiques préalablement calculés pour recommander aux utilisateurs du système un ensemble de services web. Plus précisément, nous cherchons à découvrir l'ensemble de services présents dans les motifs de services calculés dans l'étape précédente. Nous notons que toutes les tâches précédentes liées à l'extraction de motifs de services sont préalablement exécutées (voir section 6.2). La tâche de la recommandation est la seule tâche en ligne qui s'exécute pour trouver les services à recommander. Comme décrit dans la section précédente (section 6.3.3), les motifs de services sémantiquement liés sont stockés dans un MFI-tree. Nous utilisons cette structure pour découvrir tous les motifs contenant le service web considéré. Ensuite, nous récupérons tous les services présents dans ces motifs. Enfin, le système classe ces services et renvoie le résultat à l'utilisateur.

L'algorithme 4 présente la méthode de recherche de services à recommander à partir d'un service web s en utilisant l'arbre construit dans l'étape précédente. Il retourne les objets (i.e. services) présents dans les motifs contenant s. L'idée de cet algorithme est d'utiliser la table des entêtes et les index de l'arbre pour accéder directement aux différents motifs contenant le service s. Pour chaque noeud N correspondant au service s (i.e. l'étape 2), nous avons besoin de trouver les préfixes communs (PX) des motifs (i.e. lignes 3 à 8). Cela correspond à parcourir l'arbre jusqu'à la racine via les liens parents. Ensuite, nous trouvons toutes les extrémités possibles des motifs (i.e., les suffixes SX, ligne 10). Les items des préfixes et des suffixes sont fusionnés (i.e. lignes 11 et 12) et les motifs reconstruits sont retournés à la fin de l'algorithme.

Considérons l'exemple suivant : le service 5 et l'arbre de la figure 6.4(c). Pour le premier noeud correspondant au service 5, PX={1, 3} et SX={}, nous avons R={{1, 3}}. Pour le second noeud, PX={2, 3} et SX={}, nous avons donc R={{1, 3}, {2, 3}}. Dans le cas du dernier noeud, PX={3} et SX={{7}, {8}}. L'ensemble des motifs trouvés sont donc : R={{1, 3}, {2, 3}, {3, 7}, {3, 8}}. Nous fusionnons l'ensemble des services trouvés et nous renvoyons la liste des services recommandés $\{1, 2, 3, 7, 8\}$.

Notons qu'il est possible de recommander des services web à partir d'un ensemble de services S en fusionnant l'ensemble des services recommandés obtenus pour chaque service $s \in S$.

Une fois les services à recommander découverts en utilisant l'algorithme 4, ces services sont classés par ordre décroissant de leur pertinence par rapport au service demandé. Dans notre approche nous calculons un score de similarité entre les différents services web trouvés et le service demandé. Nous utilisons la distribution de probabilité sur les thèmes θ comme le critère de base pour calculer la similarité sémantique entre le service

Algorithme 4 Algorithme de recommandation de services web

Entrées :

- s : un service donné
- T : l'arbre MFI-tree contenant les motifs de services.

Sorties : R : ensemble de services à recommander.

```
1: N ← T.header-table[s] ; // noeud N : l'entête des liens inter-noeud pour le service s.
2: Tantque N != null Faire
3:    Parent ← N.parent − link ; //le noeud parent de N
4:    PX ← ∅ ; //préfix commun
5:    Tantque Parent := null Faire
6:       PX ← PX ∪ {Parent.item − name} ;
7:       Parent ← Parent.parent − link ;
8:    Fin tantque
9:    SX ← ∅ ; // l'ensemble de motifs trouvés à partir de N
10:   findSuffixes(N, ∅, SX) ; // trouver les motifs à partir de N
11:   Pour chaque motif m ∈ SX Faire
12:      R ← R ∪ {PX ∪ m} ; // ajout des motifs de services à recommander
13:   Fin pour
14:   N ← N.node − link ; // noeud suivant corespondant à s
15: Fin tantque
16: merge(R) ;// fusionner les motifs de R
17: Retourner R ;
```

demandé et les services candidats. Plus précisément, nous utilisons la mesure de proximité appelée *Multidimensional Angle* (ou *Similarité Cosinus*) ; définie dans l'équation 4.7 et qui utilise le cosinus de l'angle entre deux vecteurs (voir chapitre 4, section 4.3.3, page 77). Nous calculons la similarité entre le service demandé (i.e. requête) et chaque service à recommander en calculant la similarité Cosinus entre le vecteur p contenant la distribution de probabilité sur les thèmes du service demandé et le vecteur q contenant la distribution de probabilité sur les thèmes du service à recommander. La similarité Cosinus entre un vecteur p et un vecteur q est calculée en utilisant l'équation 6.1. Les valeurs de similarité sont dans l'intervalle [0,1] où 0 indique l'absence de similarité entre les vecteurs et 1 indique que les vecteurs sont identiques.

$$SimilarityScore = Cosinus(p, q) \tag{6.1}$$

Afin de recommander des services web pertinents, nous proposons d'utiliser un autre critère secondaire de sélection pour permettre aux services web ayant de meilleure qualité d'être classés en tête de la liste à retourner. Ainsi, nous considérons un ensemble d'attributs de la qualité de service (QoS) pour calculer un score global de qualité de services que nous combinons avec le score de pertinence *SimilarityScore* calculé précédemment. Dans

Algorithme 5 findSuffixes : trouver les motifs à partir d'un noeud.

Entrées :

- **N** : un noeud dans l'arbre
- **items** : les motifs trouvés dans une branche de l'arbre (à partir de N)
- **SX** : l'ensemble de motifs trouvés à partir de N

```
1: Si N.children-list=null Alors
2:   SX ← SX ∪ {items} ;
3: Sinon
4:   Pour chaque item dans N.children-list Faire
5:     items ← items ∪ item.item-name ;
6:     findSuffixes(item, items, SX) ;
7:   Fin pour
8: Fin si
```

notre approche, nous considérons un ensemble d'attributs tels que, la *disponibilité*, le *temps de réponse* et la *réputation numérique*, etc. Notre équipe de recherche a récemment développé un portail de services appelé WS-Portal [1] [Aznag et al., 2015] qui permet de calculer automatiquement les valeurs de ces différents attributs pour tous les services web référencés dans le portail. Nous présentons ci-dessous ces différents attributs QoS et la manière dont ils sont calculés et collectés [Aznag, 2015]. Notons que nous n'utilisons que les propriétés que nous pouvons calculer coté client du service web indépendemment des fournisseurs de services.

Réputation des services web : la réputation des services web est considérée en général comme une agrégation des évaluations fournies par les consommateurs de services. Elle représente un niveau de satisfaction des clients fourni au moment de l'interaction avec le service. Nous utilisons la formule 6.2 pour calculer le score de réputation numérique d'un service s, notée $\mathcal{R}(s)$. Ce score est défini comme la moyenne pondérée de tous les scores donnés par les consommateurs [Xu et al., 2007].

$$\mathcal{R}(s) = \frac{\sum_{i=1}^{N_s} s^i \times \lambda^{d_i}}{\sum_{i=1}^{N_s} \lambda^{d_i}} \tag{6.2}$$

où :

- N_s est le nombre de scores pour le service s,
- s^i est le i^{eme} score de service i,
- le seuil $\lambda \in [0, 1]$. Un petit λ signifie que les évaluations les plus récentes sont plus importantes par rapport aux plus anciennes et ont un impact plus important dans le calcul du score de réputation,

1. http://wsportal.aznag.net/

– d_i est la date du i^{eme} score de service.

Disponibilité et temps de réponse : après la publication d'un service dans notre registre de services, sa disponibilité est calculée automatiquement. Notre système mesure la disponibilité des services en appelant périodiquement les interfaces de chaque service. Le temps entre deux appels peut être configuré pour chaque service individuellement. Nous utilisons la méthode la plus populaire pour calculer la disponibilité des services en ligne ; notée $\mathcal{D}(s)$; qui consiste à calculer la fraction de la durée de vie de fonctionnement du service au cours de laquelle il a été accessible (voir l'équation 6.3).

$$\mathcal{D}(s) = \frac{MTBF}{(MTBF + MTTR)} \tag{6.3}$$

Où :

– **MTBF** (Mean Time Between Failure) est le temps moyen estimé entre deux défaillances d'un service donné. Ceci reflète la fréquence de défaillance du service considéré.
– **MTTR** (Mean Time To Repair) est le temps moyen estimé pour réparer le système suite à une défaillance.

Plus précisément, nous calculons le temps de réponse et la disponibilité de toutes les interfaces d'un service. A un instant donné, le temps de réponse et la disponibilité d'un service sont obtenus en calculant la moyenne des valeurs de ces mesures.

Débit ou fréquence d'utilisation : le débit d'un service présente le nombre d'utilisation d'un service web. Elle se réfère à la fréquence d'utilisation d'un service web au cours d'une période déterminée. Si un service web a été utilisé **nbUse** fois au cours d'une période allant d'une date d_1 à une date d_2 (avec $d_2 > d_1 - 1$), alors la fréquence d'utilisation est définie par la formule.

$$\mathcal{F}(s) = \frac{nbUse}{(d_2 - d_1)} \tag{6.4}$$

Taux d'utilisation : notre portail de services offre la possibilité d'invoquer automatiquement les services web référencés. Nous pouvons donc calculer facilement le taux d'utilisation (ou "l'utilisabilité") d'un service quelconque. Le taux d'utilisation est une mesure qui représente un certain nombre d'appels en utilisant l'URL d'un service web lors de l'invocation ou la découverte de ce service. La formule 6.5 présente le taux d'utilisation d'un service web en tenant compte du nombre d'invocations **nbrInvok** et le nombre de découverts **nbrDiscover** d'un service.

$$\mathcal{U}(s) = \frac{nbrInvok}{nbrDiscover} \tag{6.5}$$

Les scores de la qualité de services décrits ci-dessus ont des unités de mesure différentes. Par exemple, la disponibilité est mesurée en *pourcentage*, le temps de réponse

en *millisecondes*, le taux d'utilisabilité et le débit ont des valeurs de probabilité. Nous allons donc normaliser ces différents scores pour ne prendre en compte que des valeurs de probabilité appartenant à l'intervalle [0, 1]. Pour trouver la mesure de probabilité, nous divisons chaque score Qos individuel $q(s)$ par le score QoS maximum $MAX(q(s))$ de l'ensemble des services web candidats trouvés. Nous utilisons l'équation pour calculer la mesure de probabilité d'un score QoS pour chaque service, dénotée $P(q(s))$. On dit qu'un attribut QoS est *monotone croissant* lorsque l'augmentation de sa valeur reflète des améliorations de la qualité de service, alors que l'attribut est *monotone décroissant* lorsque que la diminution en valeur reflète des améliorations de la qualité de service. Par exemple, la disponibilité est un attribut monotone croissant et le temps de réponse est monotone décroissant.

$$P(q(s)) = \begin{cases} \frac{q(s)}{MAX(q(s))}, & \text{Si l'attribut QoS est monotone croissant} \\ 1 - \frac{q(s)}{MAX(q(s))}, & \text{Si l'attribut QoS est monotone décroissant} \end{cases} \tag{6.6}$$

Pour calculer le score global $OverallQosScore$ qui reflète la qualité de service pour chaque service web, nous combinons d'une manière logique l'ensemble des mesures de probabilités obtenues pour les différents scores QoS présentés ci-dessus (voir l'equation 6.7).

$$OverallQosScore(s) = \sum_{i=1}^{m} wp_i * P(q(s)) \tag{6.7}$$

Où m représente le nombre total des attributs QoS, $P(q_i(s)$ représente le score QoS du i-ème attribut pour le service s et wp_i représente le poids du i-ème attribut QoS ($wp_i \in [0, 1]$ et $\sum_{i=1}^{m} wp_i = 1$). Les poids wp_i peuvent être collectés à partir des préférences utilisateurs. Dans ce cas, un système de recommandation doit fournir aux utilisateurs des interfaces adéquates pour pouvoir renseigner ces différentes préférences. Dans notre approche, nous supposons que ces attributs de la qualité de services sont tous importants pour classer les services et ont le même poids wp. Il est vraiment compliqué dans la pratique de demander aux utilisateurs de renseigner l'ordre de préférence et le poids de chaque attribut QoS et surtout lorsque l'on considère plusieurs attributs.

Après avoir défini le score global de la qualité de services, nous définissons une nouvelle mesure, que nous appelons M_{rank}, permettant de classer les différents services web découverts dans l'étape précédente. L'équation 6.8 introduit cette mesure en combinant les deux scores décrits précédemment, à savoir le score de similarité sémantique $SimilarityScore$ et le score global de la qualité de service $OverallQosScore$.

$$M_{rank} = \alpha * SimilarityScore + (1 - \alpha) * OverallQosScore \tag{6.8}$$

Où $\alpha \in [0, 1]$ représente un paramètre d'ajustement.

Nous utilisons donc ce nouveau score de classement pour classer l'ensemble des services web concernés par ordre décroissant de M_{rank}. Ainsi, nous obtenons automatiquement un classement efficace des services à recommander.

6.5 Expérimentation et évaluation

Dans cette section, nous présentons les différentes expérimentations réalisées et les résultats obtenus pour la méthode de recommandation de services web proposée dans ce chapitre.

6.5.1 Corpus de services web et préparation des données

Nos expérimentations sont réalisées sur des services web réels obtenus de la collection de test *SAWSDL-TC3*[2] décrite précédemment (voir chapitre 4, section 4.5.1, page 85). Nous rappelons que cette collection contient 1088 documents WSDL annotés sémantiquement appartenant aux neuf différents domaines d'applications (voir tableau 4.2, 86). Le corpus de services web (i.e. documents WSDL) a été traité en appliquant un ensemble de techniques d'extraction d'information présentées dans le chapitre 2 (voir section 2.3.1, page 23). L'objectif de ce traitement consiste à identifier les descriptions textuelles des services (i.e. des mots potentiels) qui décrivent la sémantique de leurs fonctionnalités. Le processus du traitement de documents WSDL se compose de plusieurs étapes : *analyse et extraction d'information, tokénisation, suppression des mots vides, lemmatisation, pondération des termes et construction de la matrice transactionnelle des services (i.e. matrice STM).*

6.5.2 Protocole d'expérimentation et mesures d'évaluation

Comme nous l'avons décrit dans la section 6.2, un processus comportant un ensemble d'étapes doit être exécuté lors du traitement des descriptions de services :

1. Extraction de thèmes ;
2. Calcul des motifs sémantiques ;
3. Extraction des motifs de services ;
4. Construction de l'arbre MFI-tree.

L'objectif principal de ce processus consiste à extraire un ensemble de motifs de services qui seront utilisés par notre système de recommandation de services web. Les données textuelles observées sont représentées dans la matrice *STM* que nous utilisons comme donnée d'entraînement pour construire le modèle thématique probabiliste CTM,

2. http://www.semwebcentral.org/projects/sawsdl-tc

afin d'extraire un ensemble de thèmes. Nous rappelons que pour construire notre modèle probabiliste, le nombre de thèmes doit être choisi avant la phase d'entraînement. Nous avons précédemment montré dans le chapitre 4 que l'on obtient une meilleure performance du modèle avec le nombre de thèmes optimal $K = 90$ (voir section 4.5.2, page 86). Nous utilisons donc, pour ces expérimentations, le modèle généré avec 90 thèmes. Pour extraire les motifs sémantiques, décrit dans la section 6.3.2, nous utilisons l'algorithme CHARM-L [Zaki & Hsiao, 2005]. CHARM-L est un algorithme efficace qui permet de construire le treillis de concepts fréquents en spécifiant le support minimum *minsup* (seuil minimum). Pour construire le contexte d'extraction de motifs, nous construisons un ensemble de clusters de services en affectant un ou plusieurs thèmes à un service web donné. Pour simplifier nous utilisons la stratégie de l'affectation multiple des thèmes qui consiste à affecter un ensemble de thèmes à chaque service web en sélectionnant les $nbAssign$ premiers thèmes qui représentent mieux un service donné (voir section 6.3.1).

Pour simuler la partie en ligne (tâche de recommandation) nous procédons comme suit. Pour chaque requête présente dans la collection :

1. Rechercher les services recommandés en utilisant l'algorithme 4 ;
2. Classer les services trouvés par ordre décroissant de leur score de pertinence M_{rank} défini dans l'équation 6.8 ;
3. Évaluer la qualité des n premiers services trouvés en calculant la Précision@n et le NDCG.

Afin d'évaluer la performance de notre système de recommandation, nous calculons les mesures de *Précision@n* et le *NDCG* (*Normalized Discounted Cumulative Gain*) utilisées précédemment pour l'évaluation de la pertinence de notre système de découverte et de classement de services web (voir chapitre 5). Nous rappelons que la mesure de *Précision@n* reflète le nombre de services qui sont pertinents pour la requête de l'utilisateur. La mesure du gain cumulatif *NDCG* pénalise les systèmes classant un service non pertinent après un service pertinent, tout en favorisant les systèmes qui renvoient les services pertinents dans les premiers rangs (voir section 5.6.2, page 116). Comme nous l'avons décrit dans la section 5.6, la collection de test *SAWSDL-TC3* contient aussi 42 requêtes avec une réponse pertinente pour chaque requête. Le tableau 5.3 (voir page 115) indique le nombre de requêtes utilisées pour chaque domaine. Une réponse se compose d'un ensemble de services et à chaque service est associée une valeur de pertinence appartenant à l'ensemble des échelles de pertinence {0, 1, 2, 3} où 3 indique *Très Pertinent*, 2 *Pertinent*, 1 *Potentiellement Pertinent* et 0 *Non Pertinent*. Nous utilisons l'ensemble des 42 requêtes pour évaluer notre méthode de recommandation de services.

Notons que toutes nos expérimentations ont été réalisées sur un ordinateur personnel avec un processeur Intel Core2Duo, 2.4 GHz et 6 Go de RAM.

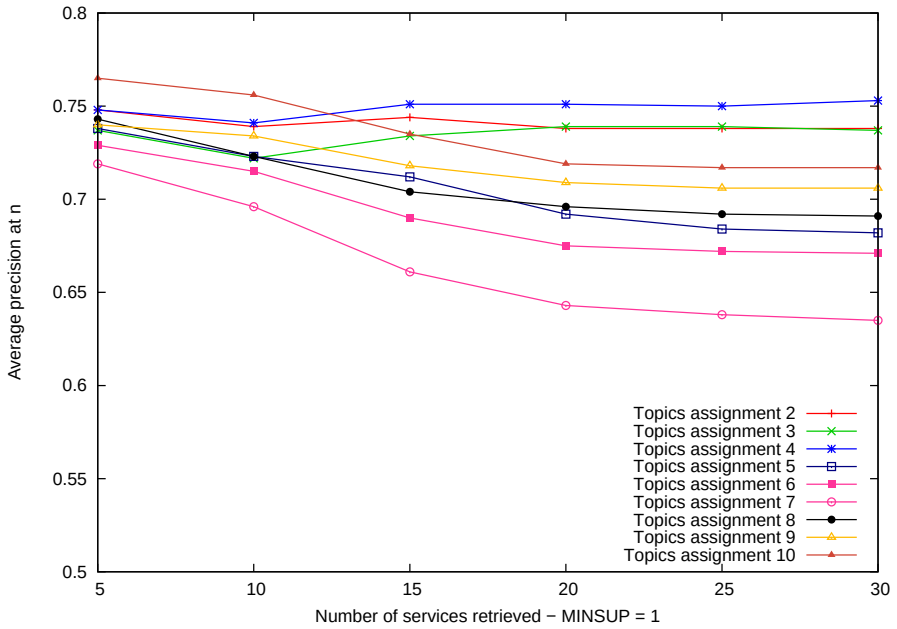


Figure 6.5 – Comparaison des valeurs de la précision moyenne obtenues pour notre méthode, sur les 42 requêtes, avec *minsup=1* et *nbAssign* variant de 2 à 10 (nombre de thèmes affectés à chaque service).

6.5.3 Résultats et discussion

Nous décrivons dans cette section, les résultats obtenus pour notre système de recommandation de services web, que nous appelons *Topic-MFI* dans le reste de ce chapitre.

Analyse de l'impact des paramètres *minsup* **et** *nbAssign* **:** la première expérimentation consiste donc à analyser l'impact de ces paramètres *minsup* et *nbAssign* en comparant les valeurs moyennes de la $Precision@n$ obtenues sur les 42 requêtes pour notre méthode Topic-MFI. Nous évaluons notre méthode en tenant compte des différentes valeurs de *minsup* (i.e. de 1 à 6) et *nbAssign* (i.e. de 2 à 10 thèmes assignés à chaque service). Notons que nous n'avons pas évalué l'impact de l'utilisation des différents attributs de la qualité de services, sur le résultat du classement de services (voir section 6.4). En effet, la collection *SAWSDL-TC3* utilisée ne fournit aucune information concernant la QoS des services web étudiés. Par conséquent, le paramètre α présent dans l'équation 6.8 est fixé à 1. Nous utilisons donc seulement le score de similarité sémantique pour classer les services web recommandés. Les résultats sont présentés dans les figures suivants :

– Comparaison des valeurs de la précision moyenne obtenues pour notre méthode,

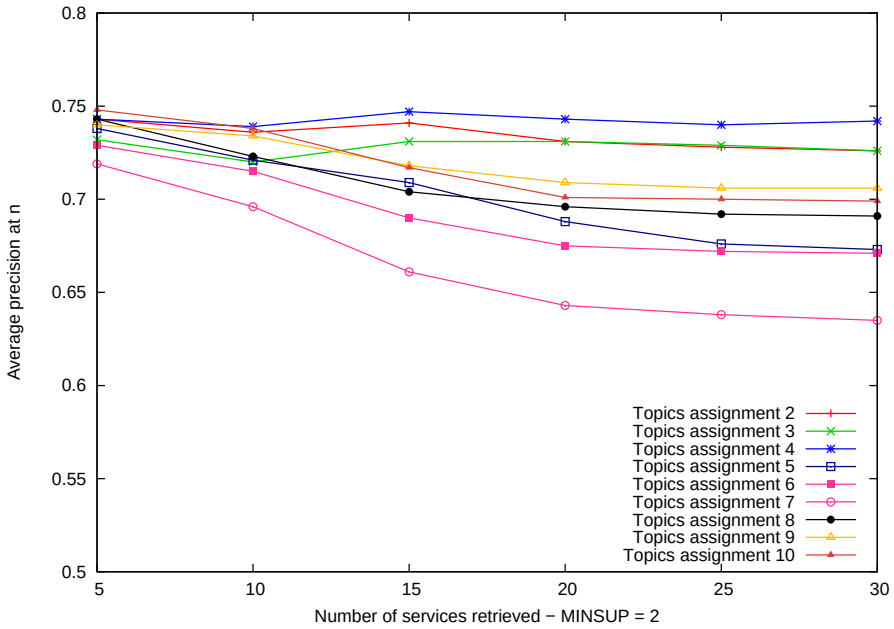


Figure 6.6 – Comparaison des valeurs de la précision moyenne obtenues pour notre méthode, sur les 42 requêtes, avec *minsup=2* et *nbAssign* variant de 2 à 10 (nombre de thèmes affectés à chaque service).

sur les 42 requêtes, avec *minsup=1* et *nbAssign* variant de 2 à 10 (voir figure 6.5) ;

- Comparaison des valeurs de la précision moyenne obtenues pour notre méthode, sur les 42 requêtes, avec *minsup=2* et *nbAssign* variant de 2 à 10 (voir figure 6.6) ;
- Comparaison des valeurs de la précision moyenne obtenues pour notre méthode, sur les 42 requêtes, avec *minsup=3* et *nbAssign* variant de 2 à 10 (voir figure 6.7) ;
- Comparaison des valeurs de la précision moyenne obtenues pour notre méthode, sur les 42 requêtes, avec *minsup=4* et *nbAssign* variant de 2 à 10 (voir figure 6.8) ;
- Comparaison des valeurs de la précision moyenne obtenues pour notre méthode, sur les 42 requêtes, avec *minsup=5* et *nbAssign* variant de 2 à 10 (voir figure 6.9) ;
- Comparaison des valeurs de la précision moyenne obtenues pour notre méthode, sur les 42 requêtes, avec *minsup=6* et *nbAssign* variant de 2 à 10 (voir figure 6.10).

Nous remarquons, à partir de ces résultats, que notre méthode donne une plus

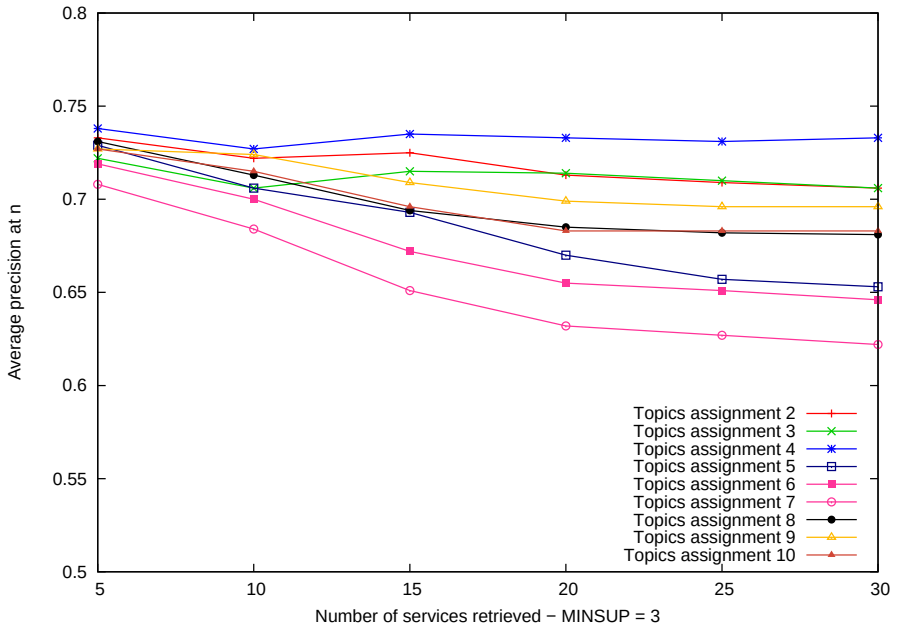


Figure 6.7 – Comparaison des valeurs de la précision moyenne obtenues pour notre méthode, sur les 42 requêtes, avec *minsup=3* et *nbAssign* variant de 2 à 10 (nombre de thèmes affectés à chaque service).

grande précision lorsque $nbAssign = 4$ (quelle que soit la valeur de *minsup*). Une petite ou grande valeur de *nbAssign* ne donne pas de bon résultat. La valeur la plus faible de précision est obtenue pour $nbAssign = 7$. Nous avons donc étudié plus précisément notre système pour $nbAssign = 4$ dont les valeurs de précision sont plus élevées.

Table 6.2 – Nombre et taille des motifs obtenus pour *assign=4* (en fonction de *minsup*).

minsup	# patterns of services	Avg. size of a pattern
1	307	3.54
2	205	5.01
3	159	6.37
4	137	7.21
5	111	8.71
6	101	9.71

Le tableau 6.2 indique le nombre de motifs de services obtenus et le nombre moyen de services dans un motif, pour $nbAssign = 4$ et *minsup* variant de 1 à 6. Plus la valeur de *minsup* est petite, plus le nombre de motifs est élevé. La taille moyenne

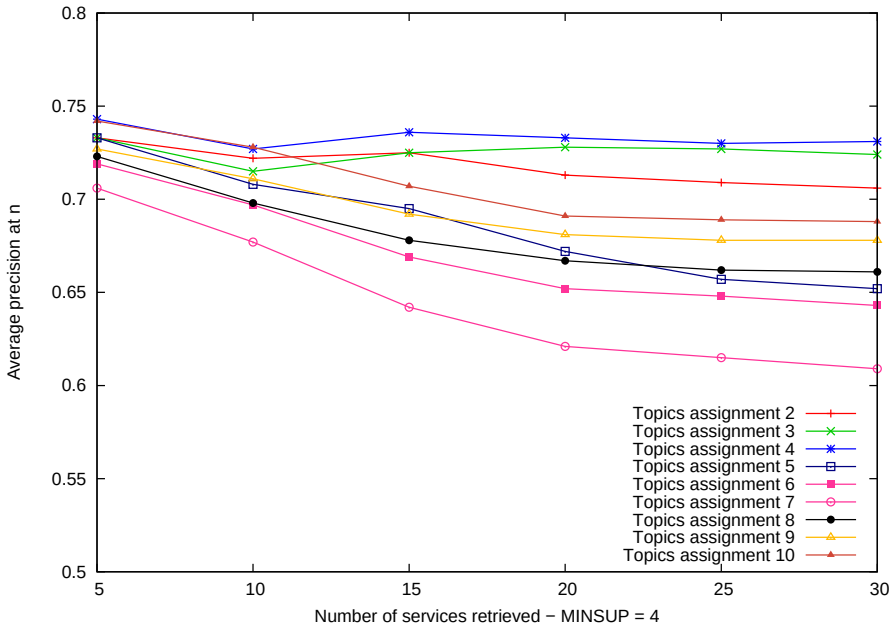


Figure 6.8 – Comparaison des valeurs de la précision moyenne obtenues pour notre méthode, sur les 42 requêtes, avec *minsup=4* et *nbAssign* variant de 2 à 10 (nombre de thèmes affectés à chaque service).

d'un motif est plus intéressante. Par exemple, si *minsup* est égal à 1, un motif peut ne contenir qu'un seul service. Néanmoins, nous pouvons observer que le nombre moyen de services dans un motif est plus grand que la valeur de *minsup*. Les services sont donc souvent corrélés. Notre système est capable de trouver ces corrélations et ne se limite pas à la valeur de *minsup*.

Évaluation de la performance du système en terme de précision et ndcg :
dans nos expérimentations, nous avons aussi comparé notre méthode de recommandation Topic-MFI avec une méthode basée sur la syntaxe et implémentée par le matchmaker *Apache Lucene*[3] et la méthode hybride sémantique implémentée par le matchmaker *SAWSDL-MX2 Matchmaker*[4]. Les figures 6.11 et 6.12 présentent respectivement les valeurs moyennes de la $Precision@n$ et $NDCG@n$ obtenues pour cette deuxième expérimentation. Ces mesures sont obtenues sur les 42 requêtes pour *ApacheLucene* et *SAWSDL-MX2 Matchmaker*. En général, les services retournés les plus pertinents sont sélectionnés et utilisés par les utilisateurs. Les valeurs moyennes de $Precision@n$ et $NDCG_n$ sont mesurées pour les 30 services premiers

3. http://lucene.apache.org/
4. http://projects.semwebcentral.org/projects/sawsdl-mx

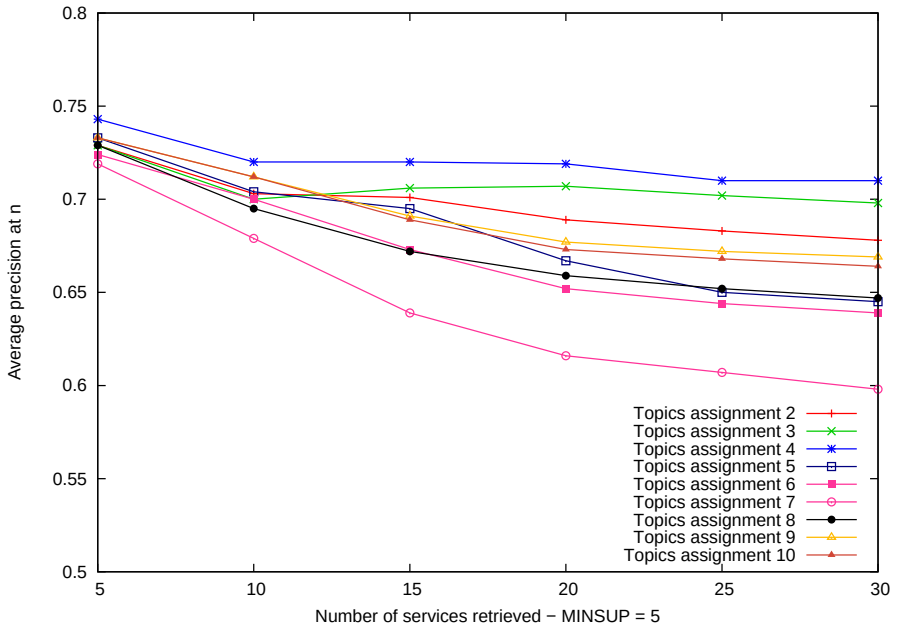


Figure 6.9 – Comparaison des valeurs de la précision moyenne obtenues pour notre méthode, sur les 42 requêtes, avec *minsup=5* et *nbAssign* variant de 2 à 10 (nombre de thèmes affectés à chaque service).

extraits de la liste complète des résultats. Dans la recherche d'information, le $NDCG$ donne des scores plus élevés aux systèmes qui classent une liste de résultats de recherche ayant une valeur de pertinence élevée et pénalise les systèmes dont la pertinence est faible. Les résultats montrent que notre méthode Topic-MFI obtient de meilleures valeurs en terme de $Precision@n$ et $NDCG_n$ quelque soit le nombre de services recommandés. En effet, notre méthode est plus performante par rapport aux autres méthodes. Les résultats montrent que ApacheLucene et SAWSDL-MX2 ont été incapables de trouver certains des services web pertinents qui ne sont pas directement liés aux requêtes via des mots-clés ou descriptions logiques. Cela montre que les services recommandés par notre système sont spécifiques à la requête de l'utilisateur. Comme nous l'avons déjà signalé dans le chapitre 5 (voir section 5.6.3) les deux systèmes ApacheLucene et SAWSDL-MX2 obtiennent des valeurs faibles de $NDCG_n$ parce que, comme le montrent les résultats de $Precision@n$, ces deux méthodes sont incapables de trouver quelques services parmi les plus pertinents. Les résultats obtenus pour notre méthode reflètent l'exactitude de notre système de recommandation.

Évaluation du temps de réponse moyen : la troisième et la dernière expérimenta-

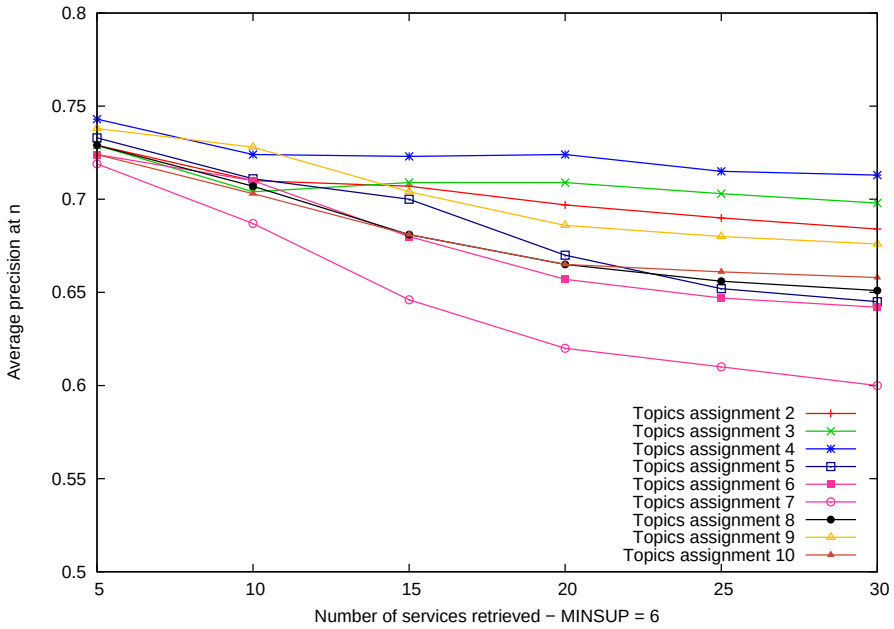


Figure 6.10 – Comparaison des valeurs de la précision moyenne obtenues pour notre méthode, sur les 42 requêtes, avec *minsup=6* et *nbAssign* variant de 2 à 10 (nombre de thèmes affectés à chaque service).

Table 6.3 – Temps de réponse moyen obtenu pour les 42 requêtes.

Système	**Temps de réponse moyen (ms)**
Topic-MFI	68
ApacheLucene	1163
SAWSDL-MX2	3045

tion réalisée dans ce chapitre consiste à évaluer le temps de réponse obtenu pour chaque système. La table 6.3 présente le temps de réponse moyen obtenu pour les systèmes *ApacheLucene*, *SAWSDL-MX2* et *Topic-MFI* en considérant l'ensemble des 42 requêtes fournies dans la collection de test. Nous remarquons, à partir de ces résultats, que notre système de recommandation *Topic-MFI* obtient une valeur très faible de temps de réponse moyen pour l'ensemble des requêtes par rapport aux autres systèmes. Ce résultat et les résultats d'évaluation de la $Precision@n$ et $NDCG@n$ obtenus montrent que notre système de recommandation est plus efficace par rapport aux systèmes de la littérature et qu'il est rapide.

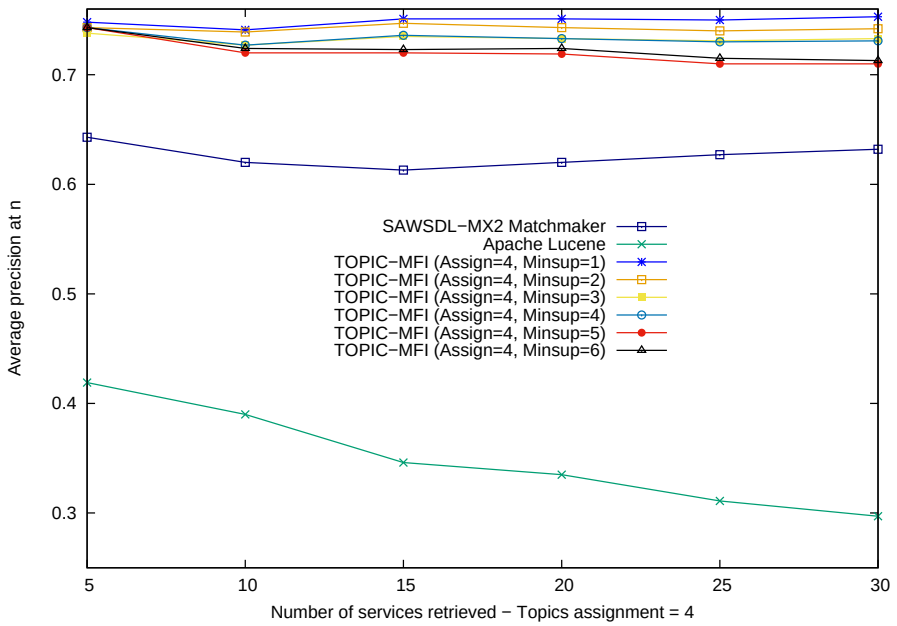


Figure 6.11 – Comparaison des valeurs de la précision moyenne obtenues pour notre méthode, sur les 42 requêtes, avec *minsup* variant de 1 à 6 et *Assign=4* (nombre de thèmes affectés à chaque service).

6.6 Conclusion

Dans ce chapitre, nous avons proposé un systéme de recommandation de services web qui combine les avantages de deux techniques, à savoir la recommandation basée sur le contenu et le filtrage collaboratif basé sur la qualité de service (i.e. la réputation, temps de réponse, disponibilité, ...). Dans notre approche, nous utilisons la notion de thème et l'extraction de motifs pour capturer la sémantique commune maximale d'un ensemble de services. Pour cela, nous avons défini la notion de motifs sémantiques qui correspondent aux motifs fréquents maximaux de thèmes. Pour calculer ces motifs et les ensembles correspondants de services web, nous avons utilisé le treillis de concepts fréquents. Afin d'effectuer des recherches rapides en se basant sur des index, notre systéme stocke l'ensemble des motifs de services extraits dans une structure particulière appelée MFI-tree. Le moteur de recommandation utilise cet arbre pour chercher les services à recommander à partir d'un service donné. Les services obtenus sont classés et recommandés à l'utilisateur. Dans notre approche, nous avons défini une nouvelle mesure de classement en combinant la similarité sémantique (basée sur la distribution de probabilité des services sur les thèmes) et un ensemble d'attributs de la qualité de service. Les résultats des expérimentaions réalisées sur des services web réels montrent que

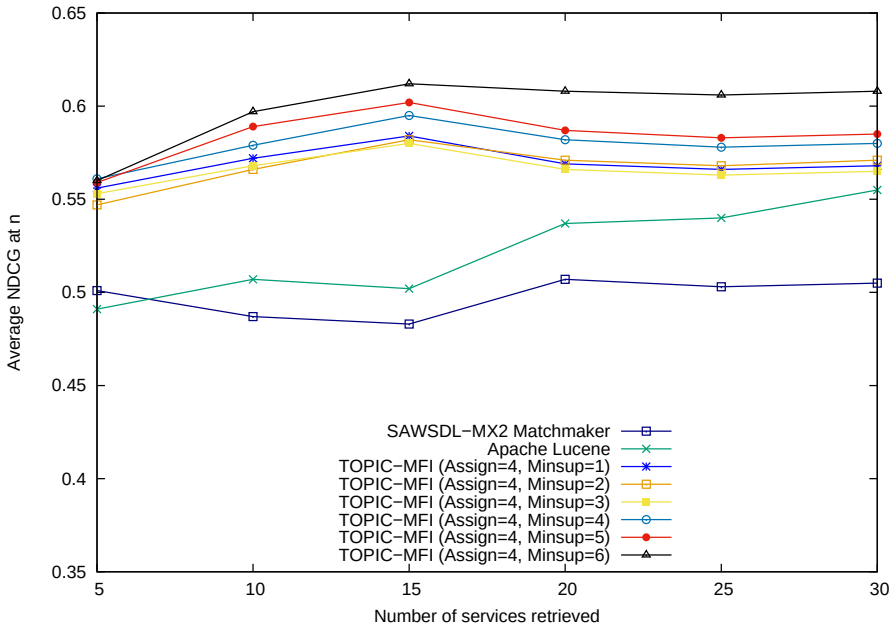


Figure 6.12 – Comparaison des valeurs moyennes de NDCGn obtenues, sur les 42 requêtes, pour notre méthode Topic-MFI, *ApacheLucene*, et *SAWSDL-MX2 Matchmaker*, pour *minsup* variant de 1 à 6 et *Assign=4* (nombre de thèmes affectés à chaque service).

notre système obtient des valeurs de précision élevées et un temps de réponse moyen très faible par rapport aux systèmes de la litérature étudiés (*ApacheLucene* et *SAWSDL-MX2 Matchmaker*).

Au lieu de ne traiter que des services individuels, nous considérons aussi un ensemble de services regroupés sous forme de communautés de services. Dans le chapitre suivant, nous proposons, une nouvelle méthode qui combine la sémantique et la topologie dans les réseaux afin d'évaluer la qualité et la cohérence sémantique des communautés détectées. La méthode proposée permet également de classer les algorithmes de détection de communautés.

7

Évaluation de la cohérence sémantique des communautés de services web et classement des algorithmes de détection de communautés

Sommaire

7.1 Introduction

Les services web sont de plus en plus adoptés pour accéder aux données et applications via le Web. Afin de faciliter le processus de découverte de services Web, il est suggéré de regrouper les services similaires sous forme de groupes, aussi appelés communautés ou sous-graphes optimaux. Dans cette optique, la détection de communautés dans un réseau ou graphe a fait l'objet de nombreuses recherches ayant abouti à plusieurs méthodes et algorithmes. L'identification des communautés implique la découverte des groupes ayant des propriétés communes telles que la similitude entre les éléments du groupe ou une structure densément connectée [Maamar et al., 2009]. La détection de communautés dans un réseau complexe est une tâche que l'on peut rapprocher de la classification non-supervisée réalisée en fouille de données classique. Le plus grand problème avec la majorité des algorithmes de détection des communautés est qu'ils ignorent complètement la *sémantique* des noeuds et considèrent uniquement la nature *topologique* des communautés [Reihanian et al., 2015]. Ainsi, les communautés produites ne sont pas homogènes, mais les noeuds sont regroupés dans plusieurs classes dont la probabilité de connexion entre les noeuds de la même classe est plus élevée qu'entre les noeuds des autres classes. Dans ce chapitre, nous visons à évaluer la qualité et la cohérence sémantique des communautés détectées. Pour cela, nous proposons une nouvelle méthode qui combine la sémantique et la topologie afin d'évaluer et classer les algorithmes de détection de communautés [Naim et al., 2016b]. Nous résumons ci-dessous les principales étapes de notre méthode (voir section 7.4 pour plus de détail) :

- Évaluation de la qualité des communautés produites.
- Évaluation de la cohérence sémantique des communautés produites.
- Classement des algorithmes de détection de communautés.

La plupart des mesures utilisées en classification non-supervisée ont été appliquées en détection de communautés. Afin d'évaluer la qualité des communautés détectées, nous utilisons des mesures classiques, à savoir la pureté et l'entropie [Zhao & Karypis, 2001]. Ces mesures sont largement utilisées pour évaluer les performances des techniques d'apprentissage supervisé et non supervisé, en particulier, le clustering. Pour évaluer la cohérence sémantique des communautés, nous introduisons une nouvelle mesure appelée la divergence sémantique des communautés basée sur la divergence de *Kullback Leibler (KL)* [Steyvers & Griffiths, 2007]. Dans notre approche, nous utilisons les thèmes et les distributions de probabilités produites par le modèle probabiliste à thèmes corrélés CTM pour calculer la divergence sémantique de chaque communauté. Dans nos expérimentations, nous avons utilisé un ensemble d'algorithmes classiques de détection de communautés, qui sont décrits dans la section 7.3, pour valider notre méthode.

La suite de ce chapitre est structurée comme suit. Tout d'abord, la section 7.2, introduit brièvement le contexte de notre travail tout en décrivant la problématique traitée dans ce chapitre. Dans la section 7.3, nous présentons la liste des algorithmes que nous avons sélectionné pour la détection des communautés. Ensuite, nous décrivons en détail dans la section 7.4, la méthode proposée pour l'évaluation et le classement des algorithmes de détection de communautés. Enfin nous présentons dans la section 7.5, les différentes expérimentations et évaluations réalisées pour valider notre méthode avant de conclure dans la section 7.6.

7.2 Contexte et formulation du problème

Comme nous l'avons précédemment mentionné dans le chapitre 3 (voir section 3.4.3, page 62), la plupart des algorithmes de détection de communautés adoptent de plus en en plus des approches topologiques. Mais malheureusement, il n'y a généralement pas de définition exacte acceptée pour la notion de communauté ; chaque algorithme fait des hypothèses différentes en se basant sur différents concepts intuitifs. La plupart des approches proposées dans la littérature supposent qu'un graphe ou un réseau d'éléments contient un ensemble plat de communautés disjointes. Ainsi, une communauté est définie comme une partition ou un sous-graphe composé de sommets densément reliés entre eux et faiblement liés aux autres sommets du graphe [Fortunato, 2010]. La détection de communautés dans un réseau a fait l'objet de nombreuses recherches ayant abouti à plusieurs méthodes et algorithmes. Néanmoins, la majorité de ces algorithmes ignorent complètement la sémantique des noeuds et considèrent uniquement la nature topologique des communautés [Reihanian et al., 2015]. Beaucoup de méthodes ont été proposées dans la littérature pour juger la qualité d'une partition [Mancoridis et al., 1998] et comparer toutes les partitions produites par les algorithmes de détection de communautés d'une manière supervisée [Leskovec et al., 2010]. Un réseau d'information est souvent représenté par un graphe où les noeuds correspondent à des entités de données et les arêtes matérialisent les relations entre ces entités [Liu et al., 2015b]. En plus de la structure topologique, les noeuds sont généralement représentés par divers types d'informations permettant de décrire leur sémantique. Dans un réseau social par exemple, une communauté peut être un groupe d'individus ayant des intérêts communs. Plusieurs réseaux sociaux vont au-delà de l'aspect topologique et considèrent la dimension sémantique. En effet, les noeuds ne sont pas seulement des boîtes noires mais il peut y avoir d'autres attributs spécifiques à ces noeuds.

Considérons, dans notre contexte, un réseau de services web qui peut être modélisé sous forme de graphe dont chacun des noeuds est considéré non seulement comme une boîte noire mais contenant une certaine information spécifique représentant une donnée textuelle.

Définition 7.1 *Un **réseau de services web** est un graphe $G = (V, E)$ où V représente un ensemble de services web et E est un ensemble d'arêtes. Le voisinage d'un noeud v, noté N_v, est l'ensemble des noeuds qui lui sont liés.*

Définition 7.2 *Une **communauté** C dans un réseau G est un sous-graphe $G_C = (V_C, E_C)$ tel que $V_C \subset V$ et $E_C \subset \{link(x, y) : x \in V_C \wedge y \in N_x\}$ telle qu'une propriété prédéfinie $P(C, G)$ est vraie.*

La détection des communautés consiste à formuler une ou plusieurs propriétés $P(C, G)$ et développer une méthode pour construire ce sous graphe avec une complexité minimale. Plusieurs algorithmes ont été proposés en tenant compte des différentes propriétés $P(C, G)$. En général, ces propriétés sont liées à la structure du graphe G et ses propriétés (composante géante, ...). Dans le cadre de la recherche d'information, il est également très connu que le poids d'un document donné dépend des autres documents contenus dans le corpus. Par analogie, nous associons à un noeud donné une sémantique qui peut être calculée à partir des autres noeuds dans le réseau. Cette sémantique est dite sémantique globale. Par conséquent, un noeud peut être représenté d'une manière générale par deux sémantiques : locale et globale.

Définition 7.3 *La **sémantique** d'un noeud v est une paire $\langle l(v), g(v) \rangle$ tel que l et g sont deux fonctions qui retournent respectivement la sémantique locale et la sémantique globale de v.*

Considérons dans le reste de ce chapitre que la sémantique locale d'un noeud est représentée par un vecteur de mots et sa sémantique globale est un mélange de thèmes. Nous considérons aussi que les noeuds appartiennent à des catégories prédéfinies. Dans ce chapitre, nous visons à répondre à ces trois questions fondamentales :

- Comment peut-on évaluer l'homogénéité des communautés détectées ?
- Comment peut-on évaluer la cohérence sémantique d'une communauté en utilisant la modélisation thématique ?
- Comment peut-on évaluer sémantiquement les algorithmes de détection de communautés ?

7.3 Algorithmes de détection de communautés

La détection de communautés a attiré beaucoup d'attention ces dernières années et plusieurs méthodes ont été proposées. Cependant, la plupart de ces méthodes ou algorithmes considèrent uniquement les données structurelles des réseaux indépendamment de la sémantique des noeuds. Plusieurs mesures, liées uniquement à la topologie du réseau, ont été utilisées comme la performance, la conductance [Brandes & Erlebach, 2005] et la

modularité [Newman & Girvan, 2004]. Une étude de synthèse sur ces mesures est faite par [Brandes et al., 2008]. Le problème de la détection des communautés s'apparente à des problèmes traités dans d'autres domaines, comme le regroupement de données ou le partitionnement de graphes. En effet, le partitionnement de graphes consiste à regrouper les noeuds d'un graphe en un nombre généralement prédéterminé de sous-groupes homogènes en minimisant le nombre de liens entre les différents groupes alors que la détection de communautés effectue la même opération avec ou sans exiger la connaissance à priori du nombre de communautés. Avec le nombre important d'algorithmes de détection de communautés existants, il est impossible d'évaluer toutes les méthodes proposées dans la littérature. Pour un état de l'art plus complet des méthodes existantes, nous invitons le lecteur intéressé à se reporter au travail décrit dans [Fortunato, 2010]. Nous présentons dans cette section, quelques algorithmes qui ont reçu le plus d'attention par la communauté scientifique et reconnus comme efficaces. Nous utilisons dans nos expérimentations, les algorithmes présentés ci-dessous pour valider la méthode d'évaluation proposée dans ce chapitre (voir section 7.5).

Louvain [Blondel et al., 2008] : l'algorithme Louvain adopte une méthode agglomérative hiérarchique. Cette méthode est constituée de deux phases qui sont exécutées de manière itérative. Initialement, chaque noeud représente une communauté. Durant la première phase, le gain de modularité est calculé en déplaçant chaque noeud dans la communauté voisine pour laquelle ce gain est maximum (positif). Le noeud reste dans sa communauté d'origine si aucun gain positif n'est possible. Cette phase est répétée, de façon séquentielle, pour tous les noeuds jusqu'à ce qu'aucun noeud ne soit déplacé. Ensuite, pendant la deuxième phase, un nouveau réseau est construit entre les communautés identifiées pendant la première phase. L'algorithme reprend la première phase sur le réseau ainsi construit jusqu'à ce que la modularité n'augmente plus.

Multi Step Greedy (MSG) [Schuetz & Caflisch, 2008] : MSG est une technique qui étend l'algorithme Greedy de Newman [Newman, 2004a] en promouvant plusieurs fusions à chaque étape. Plusieurs noeuds changent de communauté au même moment. Ils y ont ajouté un raffinement, appelé vertex mover, qui s'inspire de l'opérateur de déplacement de noeud de l'algorithme de Kernighan-Lin [Kernighan & Lin, 1970]. Chaque noeud est examiné à tour de rôle dans un ordre prédéterminé qui peut être aléatoire ou par exemple fixé par le degré croissant ou décroissant. L'examen d'un noeud consiste à évaluer son déplacement vers chaque communauté voisine, telle qu'il existe au moins un noeud de cette communauté partageant une arête avec le noeud examiné. Parmi tous ces déplacements possibles, celui qui engendre la plus grande augmentation de la modularité est exécuté. Si aucun déplacement n'engendre une variation positive de la modularité, le noeud reste dans sa communauté et l'algorithme passe au suivant. Pour obtenir un opti-

mum local selon cet opérateur, l'algorithme s'arrête lorsque, en ayant examiné tous les noeuds, aucun n'est déplacé. L'algorithme peut s'arrêter également quand la variation de modularité générée par un examen complet des noeuds est inférieure à un seuil fixé par avance, qui mesure donc une précision décimale souhaitée pour la modularité.

Greedy Clique Expansion (GCE) [Lee et al., 2010] : Lee et al. proposent d'utiliser les cliques maximales comme graines. Les communautés sont étendues en optimisant la force de communauté. Etant donné le nombre très important de graines initiales, un mécanisme est proposé pour fusionner les communautés devenues semblables (pourcentage de noeuds en commun supérieur à un paramètre).

WalkTrap [Pons & Latapy, 2005] : l'algorithme Walktrap adopte une approche hiérarchique agglomérative. Cet algorithme utilise une distance basée sur des marches aléatoires pour identifier les communautés les plus proches. Le but est de construire une structure de communautés en respectant cette distance calculée. La différence entre cet algorithme et les autres approches agglomératives se manifeste à travers le choix de communautés à fusionner à chaque itération. Chaque communauté contient un seul noeud. La première étape consiste à calculer toutes les distances entre les communautés de la partition courante. Ensuite, le choix des communautés à fusionner repose sur les distances calculées entre ces communautés. Deux communautés dont la distance est minimale (i.e qui sont les plus homogènes) seront fusionnées constituant ainsi une seule nouvelle communauté. Chaque fusion entre deux communautés engendre une nouvelle partition. Ce processus est réitéré jusqu'à l'obtention d'une seule communauté. La complexité de walkTrap est en $O(mnlog(n))$ où m est le nombre de liens et n le nombre de noeuds.

Community Overlap Propagation Algorithm (COPRA) : l'algorithme COPRA est proposé pour la première fois par Raghavan et al. [Raghavan et al., 2007]. C'est le premier algorithme qui implante l'idée de propagation de labels. C'est un algorithme itératif où à chaque itération un noeud envoie son label à ses voisins directs, et reçoit ceux de ses voisins. Chaque noeud détermine le label majoritaire qu'il adopte pour l'itération suivante. Ce processus itératif mène à un accord sur un label précis pour chaque groupe de noeuds. Dans [Liu et al., 2015a], une adaptation de cette méthode a été proposée aux cas avec recouvrement. Pour ce faire, un noeud ne peut plus choisir seulement le label le plus courant chez ses voisins, mais de maintenir une liste des labels les plus courants dans son entourage. Un paramètre de l'algorithme fixe le nombre maximum de labels qu'un noeud peut retenir (sans quoi chaque label s'étendrait à l'infini).

Fast Greedy [Clauset et al., 2004] : l'algorithme Fast Greedy, aussi appelé CMN, est un algorithme agglomératif de maximisation de la modularité. Il permet la recherche rapide des communautés dans des réseaux de grande taille. Cette méthode

consiste à regrouper itérativement les noeuds en partant d'une partition où chaque noeud forme une communauté jusqu'à obtenir une seule communauté contenant tous les noeuds. L'algorithme fusionne les paires de communautés correspondantes à la valeur maximale de la modularité. Cet algorithme a une complexité de $O(log^2 n)$ dans un réseau de n noeuds.

GANXIS : l'algorithme Ganxis, aussi appelé Speaker Listener Label Propagation (SLPA) [Xie et al., 2011], est un algorithme de propagation de labels, reprenant les principes de [Raghavan et al., 2007]. Dans SLPA, chaque noeud peut devenir soit speaker ou listener à tour de rôle. En tant que listener, le noeud reçoit des informations de ses voisins. Quand le noeud est speaker, il propage son label pendant que les autres écoutent, ce jusqu'à stabilisation, qui peut prendre un temps arbitrairement long comme dans la plupart des algorithmes de propagation de label. L'un des avantages de SLPA (ou Ganxis) est qu'il ne nécessite aucune connaissance sur le nombre de communautés. Il s'est avéré que SLPA produit des résultats significatifs sur des réseaux sociaux et génétiques. Néanmoins, la complexité temporelle de cet algorithme est $O(m)$.

7.4 Méthode proposée pour l'évaluation et le classement des algorithmes de détection de communautés

Dans cette section, nous proposons une nouvelle méthode permettant d'évaluer les algorithmes de détection de communautées en combinant des mesures classiques issues du domaine de la classification non-supervisée et de la modélisation thématique. Nous utilisons plus précisément l'*entropie* et la *pureté* [Zhao & Karypis, 2001, Mandhani et al., 2003, Ma et al., 2008] pour évaluer la qualité des communautés détectées. Ensuite, nous nous basons sur la divergence de *Kullback Leibler (KL)* [Steyvers & Griffiths, 2007] pour introduire une nouvelle mesure que nous utilisons pour évaluer la cohérence sémantique des communautés détectées.

Dans ce qui suit, nous expliquons en détail comment calculer l'entropie, la pureté et la cohérence sémantique dans notre contexte.

7.4.1 Mesures de qualité des communautés détectées

Nous supposons que nous avons q classes qui représentent les partitions des services web (i.e. domaines d'applications ou catégories), p communautés détectées par chaque algorithme et n le nombre total de services web. Nous définissons ci-dessous les deux mesures d'entropie et de pureté dans notre contexte.

Entropie : l'**entropie** mesure la façon dont les différentes classes sémantiques sont représentées dans chaque communauté. Étant donné une communauté particulière

C_j de taille n_j, l'*entropie* de cette communauté est définie comme suit :

$$E(C_j) = -\frac{1}{log(q)} \sum_{i=1}^{q} \frac{n_j^i}{n_j} log(\frac{n_j^i}{n_j}) \tag{7.1}$$

Où q est le nombre de domaines, et n_j^i est le nombre de services du i^{me} domaine et qui font partie de la comunauté C_j.

L'entropie globale est définie comme la somme pondérée des entropies des communautés individuelles (voir équation 7.2). Une communauté est dite bonne si elle ne contient que des services d'un seul domaine ayant le même interêt. Dans ce cas, l'entropie globale est égal à zéro. On considère donc qu'une valeur faible de l'entropie traduit une bonne méthode de détection de communautés.

$$Entropy = \sum_{j=1}^{p} \frac{n_j}{n} E(C_j) \tag{7.2}$$

Pureté : la **pureté** évalue la cohérence d'une communauté. Il s'agit du degré auquel une communauté contient des services d'un seul domaine. La pureté d'une communauté C_j est définie comme suit :

$$P(C_j) = \frac{1}{n_j} max_i(n_j^i) \tag{7.3}$$

Où $max_i(n_j^i)$ est le nombre de services du domaine dominant dans la communauté C_j et n_j^i représente le nombre de services du i-ème domaine faisant partie de cette communauté.

La mesure de la pureté représente la fraction de la taille globale de la communauté dont le plus grand domaine des services lui est affecté. Ainsi, la pureté globale est la somme pondérée des puretés de communautés individuelles (voir l'équation 7.4). En général, les grandes valeurs de la pureté traduisent une bonne méthode de détection de communautés.

$$Purity = \sum_{i=1}^{f} \frac{n_i}{n} P(C_i) \tag{7.4}$$

La mesure d'entropie est plus complète que la pureté parce qu'elle tient compte de la répartition globale de tous les domaines dans une communauté donnée plutôt que de ne considérer que le nombre de services existants et le domaine dominant. Contrairement à la mesure de la pureté, pour un algorithme idéal dont les services d'une communauté appartiennent seulement à un seul domaine, l'entropie de la communauté sera 0.

7.4.2 Mesure de divergence sémantique des communautés

Comme mentionné auparavant, nous utilisons dans le cadre de cette thèse les modèles thématiques probabilistes, plus précisément le modèle probabiliste à thèmes corrélés (CTM) [Blei & Lafferty, 2007], pour extraire un ensemble de thèmes à partir des descriptions de services web. Dans ce chapitre, nous utilisons ces thèmes comme critère de base pour le calcul de la divergence sémantique de chaque communauté détectée. Dans notre contexte, les thèmes extraits sont utilisés comme techniques efficaces de réduction des dimensions en capturant des relations sémantiques entre mots-thèmes et thèmes-services, exprimées sous forme de distributions de probabilités (voir chapitre 2, section 2.3, page 23).

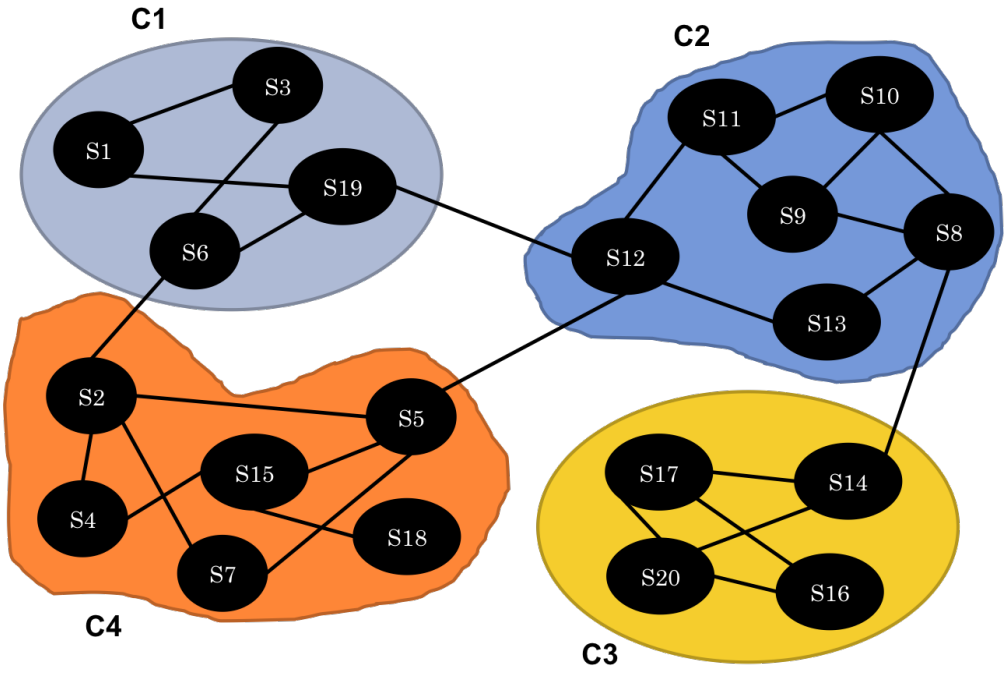


Figure 7.1 – Un exemple de réseau de services web comportant 4 communautés.

Pour calculer la divergence sémantique des communautés détectées, nous nous basons sur les distributions de probabilité apprises. Plus précisément, nous utilisons la distribution de probabilité sur les thèmes pour chaque service web. Ainsi, nous pouvons assigner à chaque noeud (service web) dans le réseaux de services un modèle θ_x qui représente la distribution de probabilité sur les thèmes. Il existe de nombreuses mesures pour calculer la distance entre la sémantique de deux distributions θ_x et θ_y (i.e. dans notre cas ; θ_x et θ_y sont respectivement les distributions de probabilité sur les thèmes pour les noeuds x et y). Nous rappelons que nous utilisons dans le cadre de cette thèse, la divergence de *Kullback Leibler (KLD)* [Kullback & Leibler, 1951] pour calculer la distance sémantique entre deux

distributions (voir chapitre 5, section 5.4.2, page 105). L'équation 5.4 définit la divergence de *Kullback Leibler (KLD)* entre deux distributions de probabilité θ_x et θ_y (voir page 105). Nous mesurons la divergence sémantique pour chaque paire de sommets correspondant à une arête dans le réseau de services, en utilisant la divergence $SymmetricKLD$ définie par l'équation 5.5 en se basant sur la divergence KLD. En effet, KLD est une mesure non symétrique entre deux distributions de probabilité, tandis que $SymmetricKLD$ obtient la symétrie en additionnant les deux divergences KLD comme définie dans l'équation 5.5 (voir page 105).

Pour chaque communauté, nous calculons la divergence sémantique, dénotée CSD (Community Semantic Divergence), comme étant la moyenne des divergences sémantiques de ses noeuds connectés. L'équation 7.5 définit la divergence sémantique d'une communauté. Plus la divergence sémantique est petite, plus la communauté est sémantiquement cohérente.

$$CSDivergence(C_i) = \frac{1}{|E_{C_i}|} \sum_{(x,y)\in E_{C_i}} SymmetricKL(\theta_x, \theta_y) \tag{7.5}$$

où E_{C_i} est l'ensemble des arêtes d'une communauté C_i.

Prenons par exemple la communuaté C_1 de l'exemple présenté dans la figure 7.1. La divergence sémantique de cette communauté est la moyenne des divergences sémantiques de ces quatres arêtes (i.e. (s_1, s_3), (s_1, s_{19}), (s_3, s_6) et (s_6, s_{19})), comme illustré par l'équation 7.6.

$$CSDivergence(C_1) = \frac{1}{|4|}(SKL(\theta_{s_1}, \theta_{s_3}) + SKL(\theta_{s_1}, \theta_{s_{19}}) + SKL(\theta_{s_3}, \theta_{s_6}) + SKL(\theta_{s_6}, \theta_{s_{19}})) \tag{7.6}$$

Où $SKL(\theta_x, \theta_y) = SymmetricKL(\theta_x, \theta_y)$

7.4.3 Classement des algorithmes de détection de communautés

Nous proposons dans cette section, une nouvelle méthode d'évaluation et de classement des algorithmes de détection de communautés. Nous nous basons sur les trois mesures décrites ci-dessus (la pureté, l'entropie et la divergence sémantique) pour évaluer respectivement la qualité et la cohérence sémantiques des communautés produites par les systèmes de détection de communautés. Nous pouvons par conséquent, évaluer et classer ces systèmes.

Dans notre contexte, nous supposons qu'un ensemble de communautés de services web sont produites par un ou plusieurs systèmes de détection de communautés (par exemple les algorithmes présentés dans la section 7.3). Une fois les communautés des services web détectées, nous pouvons les évaluer ainsi que les algorithmes considérés en suivant les

trois principales étapes suivantes :

1. **Évaluation de la qualité des communautés :** nous évaluons premièrement, les communautés détectées par chaque algorithme en considérant les deux mesures définies précédemment : la pureté et l'entropie. Une communauté est considérée bonne si la valeur obtenue pour la pureté est élevée et celle de l'entropie est petite. Pour évaluer la qualité des communautés produites par un algorithme, nous introduisons une nouvelle mesure, dénoté *GCQuality* (Global Community Quality), en combinant la pureté et l'entropie globales obtenues par chaque algorithme. Chaque algorithme est caractérisé par un score *GCQuality* comme définit dans l'équation 7.7. Les valeurs du score *GCQuality* sont dans l'intervalle [0,1]. Des valeurs élevées du score *GCQuality* traduisent un bon algorithme de détection de communautés. Chaque communauté peut être caractérisé par ce score en considérant la pureté et l'entropie obtenues pour cette communauté. Aussi, plus ce score est élevé, plus la communauté est bonne.

$$GCQuality = Purity * (1 - Entropy) \tag{7.7}$$

2. **Évaluation de la cohérence sémantique des communautés :** la deuxième phase de notre méthode consiste à évaluer la cohérence sémantique des communautés détectées. Cela nous permet également de classer les différents algorithmes étudiés. Dans un premier temps, nous calculons la divergence sémantique globale *GSDivergence* (Global Semantic Divergence), de chaque algorithme comme étant la moyenne des divergences sémantiques de toutes les communautés détectées par cet algorithme comme décrit dans l'équation 7.8. Une faible valeur de la divergence sémantique globale indique que les communautés produites sont sémantiquement cohérentes. Cela montre que la méthode de détection de communautés est efficace.

$$GSDivergence = \frac{1}{p} \sum_{i=1}^{p} CSDivergence(C_i) \tag{7.8}$$

Où p représente le nombre de communautés détectées par un algorithme de détection de communautés.

Notons que les valeurs de cette mesure n'appartiennent pas à l'intervalle [0,1]. Nous pouvons obtenir des valeurs très élevées de la divergence sémantique globale lorsque un algorithme détecte plusieurs communautés non cohérentes sémantiquement. Afin d'obtenir des valeurs de probabilité appartenant à l'intervalle [0,1], nous introduisons une nouvelle mesure que nous appelons *GSCoherence* (Global Semantic Coherence) permettant d'évaluer la cohérence sémantique globale des communautés détectées. Pour cela, nous divisons la divergence sémantique globale *GSDivergence* par la divergence sémantique maximale obtenue pour les communautés détectées par un

algorithme. Ensuite, nous soustrayons le résultat obtenu de 1 pour avoir la cohérence sémantique globale pour chaque algorithme comme décrit dans l'équation 7.9. De faibles valeurs de $GSCoherence$ traduisent que les communautés détectées sont sémantiquement cohérentes. Ce qui traduit ainsi une bonne méthode de détection de communautés.

$$GSCoherence = 1 - \frac{GSDivergence}{\max_{(c_j \in \mathcal{C})} CSDivergence(c_j)} \tag{7.9}$$

Où $\mathcal{C}$ représente l'ensemble des communutés produites par un algorithme.

3. Évaluation et classement des algorithmes de détection de communautés : la troisième et dernière étape de notre méthode consiste à calculer un score global permettant de classer les algorithmes de détection de communautés en combinant les deux mesures introduites précédemment $GCQuality$ (mesure de la qualité globale) et $GSCoherence$ (mesure de la cohérence sémantique globale). Chacune de ces mesures nous permet d'évaluer et classer les communautés produites par chaque algorithme et ainsi de classer les différents algorithmes étudiés. Nous considérons que ces deux classements sont cohérents si les valeurs des deux mesures $GCQuality$ et $GSCoherence$ sont élevées. Par conséquent, les communautés détectées sont considérées cohérentes sémantiquement. Dans notre approche, nous calculons pour chaque algorithme un score global, que nous appelons $GCRankingScore$ (Global Community Ranking Score), en combinant les deux mesures $GCQuality$ et $GSCoherence$ comme décrit dans l'équation 7.10. Ce score permet de classer les différents algorithmes de détection de communautés. Il permet également d'évaluer la qualité et la cohérence sémantique globales des communautés détectées par chaque algorithme. Des valeurs élevées du score GCRankingScore traduisent un bon algorithme de détection de communautés.

$$GCRankingScore = \alpha * GCQuality + (1 - \alpha) * GSCoherence \tag{7.10}$$

Où α représente un paramètre d'ajustement ($\alpha \in [0, 1]$).

7.5 Expérimentation et évaluation

Dans cette section, nous présentons les différentes expérimentations réalisées et les résultats d'évaluation obtenus pour l'évaluation des algorithmes de détection de communautés en considérant les mesures introduites dans ce chapitre.

7.5.1 Collection de services web et protocole d'expérimentation

Comme pour les chapitres précédents, nous rappelons que nos expérimentations ont été réalisées sur des services web réels obtenus à partir de la collection de test publique appelée *SAWSDL-TC3*[1](voir chapitre 4, section 4.5.1, page 85). Nous rappelons que cette collection contient 1088 documents WSDL annotés sémantiquement appartenant aux neuf différents domaines d'applications, à savoir : Communication, Education, Economy, Food, Geography, Medical, Military, Travel, Simulation. Le tableau 4.2 indique le nombre de services utilisés pour chaque domaine (voir chapitre 4, page 86).

Nous présentons dans cette section, les différentes étapes du protocole expérimental de l'évaluation réalisée dans ce chapitre :

1. **Traitement de documents WSDL :** le corpus de services web (i.e. documents WSDL) a été traité en appliquant un ensemble de techniques d'extraction d'information présentées dans le chapitre 2 (voir section 2.3.1, page 23). L'objectif de ce traitement consiste à identifier les descriptions textuelles des services (i.e. des mots potentiels) qui décrivent la sémantique de leurs fonctionnalités. Le processus du traitement de documents WSDL se compose de plusieurs étapes : *analyse et extraction d'information, tokénisation, suppression des mots vides, lemmatisation, pondération des termes et construction de la matrice transactionnelle des services (i.e. matrice STM).*
2. **Extraction de thèmes :** apprentissage du modèle probabiliste à thèmes corrélés CTM et extraction d'un ensemble de thèmes à partir des descriptions de services. Les données textuelles observées produites dans l'étape précédente sont représentées dans la matrice *STM* que nous utilisons comme donnée d'entraînement pour construire le modèle thématique probabiliste CTM. Nous rappelons que pour construire notre modèle probabiliste, le nombre de thèmes doit être choisi avant la phase d'entraînement. Nous avons montré précédemment dans le chapitre 4 que l'on obtient une meilleure performance du modèle avec le nombre de thèmes optimal $k = 90$ (voir section 4.5.2, page 86). Nous utilisons donc, pour ces expérimentations, le modèle généré avec 90 thèmes.
3. **Construction de réseaux de services :** nous utilisons pour cette expérimentation le réseau de similitude de services décrit dans le chapitre 4. Plus précisément, nous construisons un réseau de services web qui représente la similarité entre les différents services. Nous nous basons sur les représentations vectorielles définies par les scores TF-IDF et la mesure Cosine pour calculer la similarité entre deux services. Une arête est crée dans le réseau si la similarité entre deux services est supérieure à un seuil donné. Dans cette expérimentation nous avons utilisé le réseau de similitude construit avec le seuil = 0.5 (voir chapitre 4, tableau 4.4, page 92).

1. http://www.semwebcentral.org/projects/sawsdl-tc

Algorithmes	NbrCom	Nombre de services de chaque communauté								
		C1	C2	C3	C4	C5	C6	C7	C8	C9
1 CMN	3	493	524	33	-	-	-	-	-	-
2 LOUVAIN	8	239	101	179	121	65	26	47	270	-
3 MSG	3	527	239	282	-	-	-	-	-	-
4 GCE	7	161	386	61	140	94	30	58	-	-
5 COPRA	2	529	521	-	-	-	-	-	-	-
6 WALKTRAP	6	30	189	138	171	113	405	-	-	-
7 GANXIS	9	764	438	364	310	214	206	94	36	4

Table 7.1 – Le nombre de communautés sélectionnées *NbrCom* pour chaque algorithme ainsi que le nombre de services dans chaque communauté.

4. **Détection de communautés de services :** après avoir construit le réseau de services web, nous avons appliqué les algorithmes de détection de communautés présentés dans la section7.3.

5. **Évaluation des communautés détectées et classement des algorithmes :** nous évaluons chaque communauté détectée en calculant les trois mesures décrites dans la section 7.4 : la pureté, l'entropie et la divergence sémantique. Ensuite, nous classons les différents algorithmes sélectionnés en calculant le score global 7.10 combinant les deux mesures introduites précédemment *GCQuality* (mesure de la qualité globale, voir équation 7.7) et *GSCoherence* (mesure de la cohérence sémantique globale, voir équation 7.9)

Dans les sections suivantes, nous discutons les résultats obtenus pour l'évaluation des communautés détectées ainsi que le classement des algorithmes sélectionnés.

7.5.2 Nombre et taille des communautés détectées

Une fois les algorithmes sélectionnés appliqués, chaque algorithme a produit un ensemble de communautés de services web. En analysant le résultat de chaque algorithme, nous avons remarqué que quelques algorithmes détectent plusieurs communautés. Certaines communautés sont trop petites car ayant moins de 3 noeuds (i.e. services). Pour cette expérimentation, nous ne considérons que les communautés ayant au moins trois noeuds. Le tableau 7.1 montre le nombre de communautés considérées pour chaque algorithme ainsi que le nombre de services de chaque communauté. Les communautés sont nommées Ci avec $i \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Nous remarquons que les trois algorithmes CMN, MSG et COPRA détectent très peu de communautés qui sont respectivement les communautés 1, 2 et 3. La taille de ces communautés est assez grande sauf la communauté $C3$ de CMN qui contient 33 services. Les autres algorithmes détectent un nombre élevé considérable de communautés par rapport au nombre de domaines de services web.

		Domaines d'applications							
Algo		**Com**	**Eco**	**Edu**	**Foo**	**Geo**	**Medi**	**Tra**	**Mil**
1	C1	**11.97**	**68.36**	**15.01**	**2.23**	**0.2**	**1.83**	**0.41**	0.0
	C2	0.0	**2.48**	**36.07**	**0.38**	**11.26**	**11.83**	**30.34**	**7.63**
	C3	0.0	**12.12**	0.0	**84.85**	0.0	**3.03**	0.0	0.0
2	C1	**24.69**	**51.88**	**21.76**	0.0	0.0	**1.67**	0.0	0.0
	C2	0.0	**48.51**	**15.84**	**35.64**	0.0	0.0	0.0	0.0
	C3	0.0	**93.3**	**4.47**	**1.12**	0.0	**0.56**	**0.56**	0.0
	C4	0.0	**5.79**	**60.33**	0.0	0.0	0.0	**1.65**	**32.23**
	C5	0.0	**4.62**	**4.62**	**1.54**	**70.77**	**1.54**	**16.92**	0.0
	C6	0.0	0.0	**3.85**	**3.85**	**3.85**	**88.46**	0.0	0.0
	C7	0.0	**2.13**	**2.13**	0.0	0.0	**2.13**	**93.62**	0.0
	C8	0.0	**1.11**	**40.74**	**0.37**	**4.81**	**14.81**	**38.15**	0.0
3	C1	**11.01**	**63.76**	**13.09**	**7.21**	0.0	**4.93**	0.0	0.0
	C2	**0.42**	**3.35**	**38.91**	**0.42**	0.0	**17.57**	**22.59**	**16.74**
	C3	0.0	**3.55**	**35.82**	**0.35**	**21.28**	**1.06**	**37.94**	0.0
4	C1	**7.45**	**55.9**	**10.56**	**25.47**	0.0	0.0	**0.62**	0.0
	C2	0.0	**3.89**	**32.38**	**0.26**	**15.54**	**17.36**	**30.57**	0.0
	C3	**9.84**	**78.69**	**8.2**	**3.28**	0.0	0.0	0.0	0.0
	C4	0.0	**2.86**	**67.86**	0.0	0.0	0.0	**0.71**	**28.57**
	C5	**62.77**	**35.11**	**1.06**	**1.06**	0.0	0.0	0.0	0.0
	C6	0.0	**93.33**	**3.33**	**3.33**	0.0	0.0	0.0	0.0
	C7	0.0	0.0	**1.72**	0.0	0.0	0.0	**98.28**	0.0
5	C1	**11.15**	**65.97**	**14.18**	**7.37**	0.0	**1.13**	**0.19**	0.0
	C2	0.0	**0.96**	**36.08**	**0.38**	**11.52**	**12.67**	**30.71**	**7.68**
6	C1	0.0	**13.33**	0.0	**86.67**	0.0	0.0	0.0	0.0
	C2	**28.57**	**59.26**	**6.35**	**5.82**	0.0	0.0	0.0	0.0
	C3	**2.9**	**67.39**	**26.81**	0.0	0.0	**2.9**	0.0	0.0
	C4	0.0	**79.53**	**19.3**	0.0	0.0	**0.58**	**0.58**	0.0
	C5	0.0	**3.54**	**61.06**	0.0	0.0	0.0	0.0	**35.4**
	C6	0.0	**1.23**	**27.65**	**0.74**	**14.81**	**16.3**	**39.26**	0.0
7	C1	0.0	**28.14**	**25.92**	**4.06**	**7.85**	**8.38**	**20.42**	**5.24**
	C2	**11.87**	**57.08**	**17.35**	**8.9**	0.0	**4.34**	0.0	**0.46**
	C3	0.0	**65.66**	**28.57**	**2.2**	**0.27**	**2.2**	**0.82**	**0.27**
	C4	**2.26**	**52.26**	**24.52**	**11.29**	0.0	**9.35**	0.0	**0.32**
	C5	**27.57**	**68.69**	**3.27**	**0.47**	0.0	0.0	0.0	0.0
	C6	0.0	0.0	**37.86**	0.0	**22.82**	**7.28**	**13.11**	**18.93**
	C7	0.0	**24.47**	**39.36**	**5.32**	**9.57**	0.0	**21.28**	0.0
	C8	0.0	0.0	0.0	0.0	0.0	0.0	**100.0**	0.0
	C9	0.0	0.0	0.0	0.0	0.0	**100.0**	0.0	0.0

Table 7.2 – Répartition (en pourcentage %) de services de chaque communauté sur les domaines d'applications (Com : Communication, Eco : Economy, Edu : Education, Geo : Geography, Medi : Medical, Tra : Travel, Mil : Military)

		Communautés détectées								
Algo	**Mesures**	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**
1	Purity	0.68	0.36	0.85	-	-	-	-	-	-
	Entropy	0.48	0.74	0.24	-	-	-	-	-	-
	GCQuality	0.35	0.09	**0.65**	-	-	-	-	-	-
2	Purity	0.52	0.49	0.93	0.6	0.71	0.88	0.94	0.41	-
	Entropy	0.52	0.49	0.15	0.43	0.46	0.23	0.15	0.59	-
	GCQuality	0.25	0.25	**0.79**	0.34	0.38	**0.68**	**0.8**	0.17	-
3	Purity	0.64	0.39	0.38	-	-	-	-	-	-
	Entropy	0.55	0.71	0.6	-	-	-	-	-	-
	GCQuality	0.29	0.11	0.15	-	-	-	-	-	-
4	Purity	0.56	0.32	0.79	0.68	0.63	0.93	0.98	-	-
	Entropy	0.55	0.7	0.35	0.36	0.36	0.14	0.04	-	-
	GCQuality	0.25	0.1	0.51	0.44	0.4	**0.8**	**0.94**	-	-
5	Purity	0.66	0.36	-	-	-	-	-	-	-
	Entropy	0.51	0.72	-	-	-	-	-	-	-
	GCQuality	0.32	0.10	-	-	-	-	-	-	-
6	Purity	0.87	0.59	0.67	0.8	0.61	0.39	-	-	-
	Entropy	0.19	0.49	0.4	0.27	0.38	0.67	-	-	-
	GCQuality	**0.7**	0.3	0.4	0.58	0.38	0.13	-	-	-
7	Purity	0.28	0.57	0.66	0.52	0.69	0.38	0.39	1.0	1.00
	Entropy	0.83	0.6	0.42	0.6	0.36	0.71	0.68	0.0	0.00
	GCQuality	0.05	0.23	0.38	0.21	0.44	0.11	0.12	**1.00**	**1.00**

Table 7.3 – Evaluation de la qualité des communautés détectées par les algorithmes sélectionnés.

Notons que notre collection de test contient neuf domaines d'applications (voir tableau 4.2, page 86). Le tableau 7.2 montre la répartition (en pourcentage %) des services de chaque communauté sur les différents domaines d'applications. Nous remarquons que les domaines Economy et Education sont présents presque dans toutes les communautés détectées. En effet, comme illustré dans le tableau 7.2, ces deux domaines regroupent un nombre important de services (358 services pour Economy et 285 pour Education).

7.5.3 Evaluation de la qualité des communautés détectées

Pour évaluer la qualité des communautés détectées, nous nous basons sur les deux mesures classiques ; la pureté et l'entropie. Pour chaque algorithme, nous avons calculé la pureté et l'entropie des communautés détectées. Nous avons également calculé le score *GCQuality* (voir équation 7.7) qui combine la pureté et l'entropie obtenues pour chaque communauté. Le tableau 7.3 représente les valeurs obtenues pour la pureté, l'entropie et le score *GCQuality* pour toutes les communautés détectées par chaque algorithme. Une communauté est considérée bonne si la valeur obtenue pour la pureté est élevée et

celle de l'entropie est petite. Par conséquent, des valeurs élevées du score *GCQuality* indiquent une bonne communauté de services web.

Les valeurs de la pureté et l'entropie globales ainsi que le score global combinant ces deux mesures sont représentées dans la figure 7.2. Les communautés détectées par CMN, MSG et COPRA ont une entropie similaire (très grandes communautés), mais CMN est mieux par rapport aux MSG et COPRA du point de vue de la pureté. Les communautés détectées par l'algorithme Louvain sont meilleures que celles détectées par les algorithmes CGE, Walktrap et Ganxis.

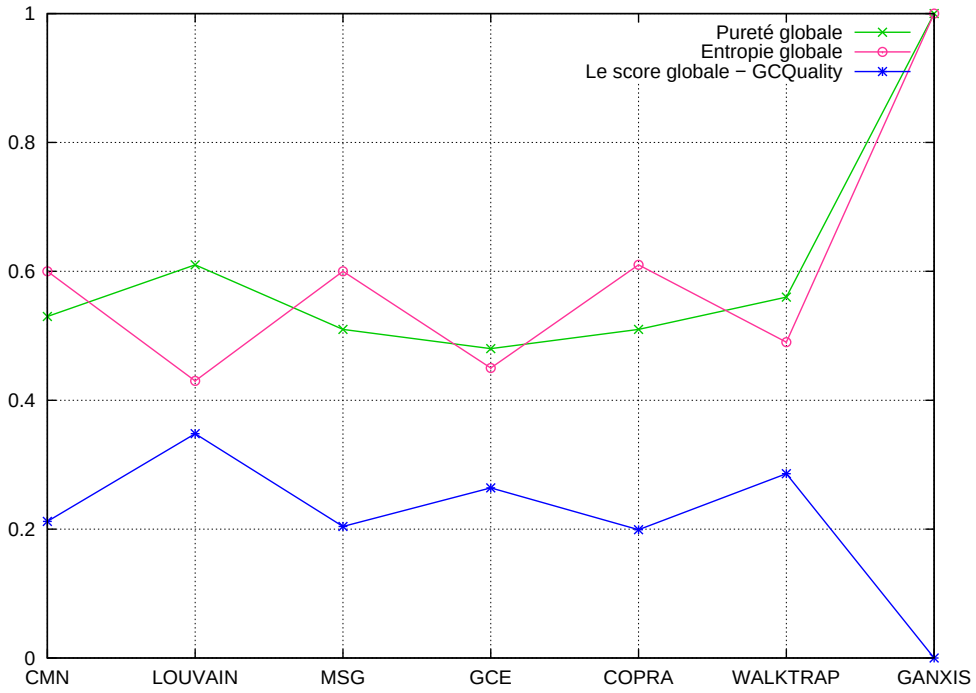


Figure 7.2 – Pureté et entropie globales pour tous les algorithmes.

7.5.4 Évaluation de la cohérence sémantique des communautés détectées

Nous allons maintenant évaluer la cohérence sémantique des communautés détectées en considérant la divergence sémantique *CSDivergence* de leurs noeuds connectés. Pour chaque algorithme, nous calculons la divergence sémantique pour toutes les communautés détectées. Comme indiqué dans l'équation 7.5, la divergence sémantique d'une communauté est calculée comme étant la moyenne des divergences sémantiques de ses noeuds

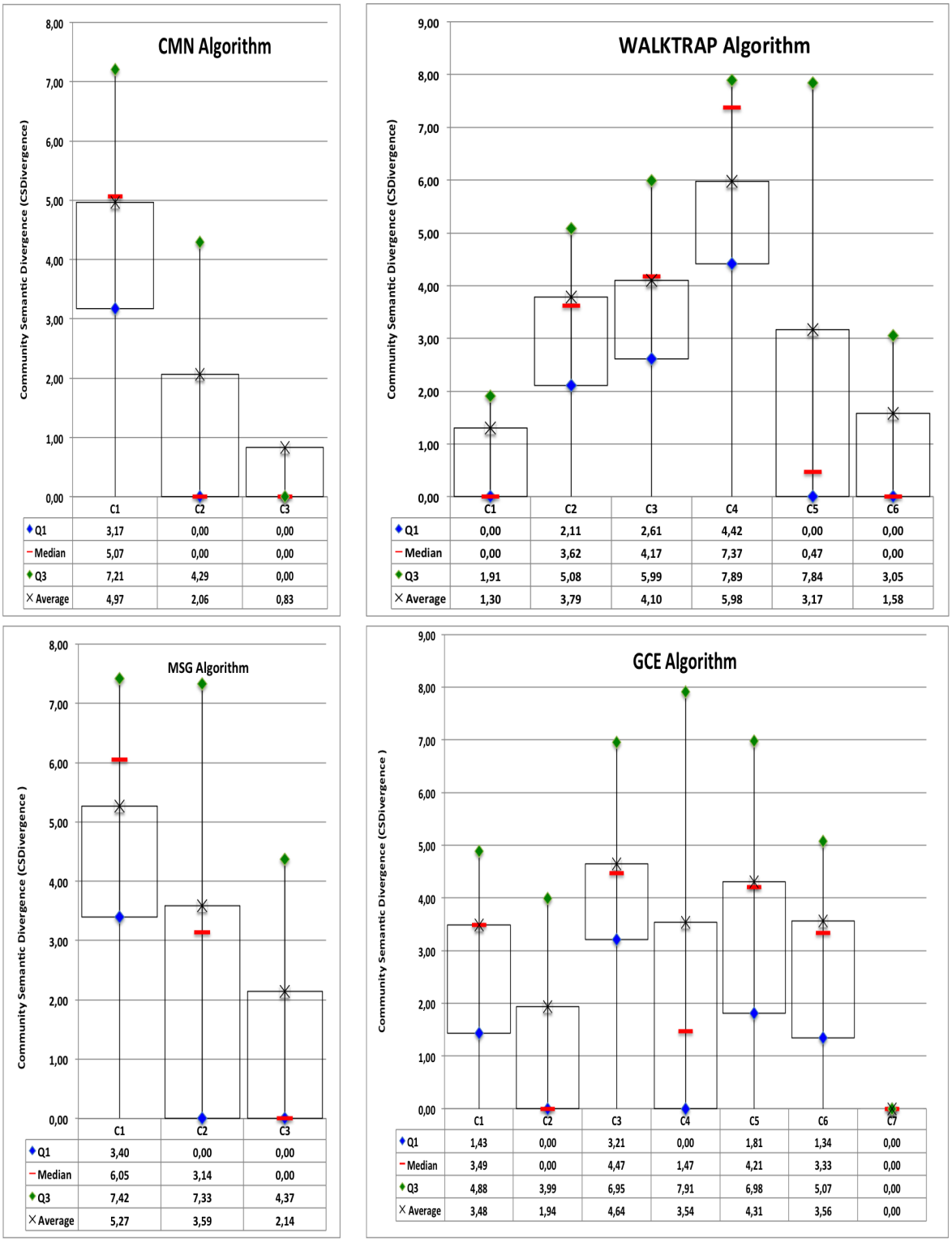


Figure 7.3 – La divergence sémantique des communautés détectées par les algorithmes suivants : CMN, Walktrap, MSG et GCE.

	Communautés détectées								
Algo	C1	C2	C3	C4	C5	C6	C7	C8	C9
1	4.97	2.06	0.83	-	-	-	-	-	-
2	4.35	3.08	5.81	3.35	0.25	0.00	0.14	2.10	-
3	5.27	3.59	2.14	-	-	-	-	-	-
4	3.48	1.94	4.64	3.54	4.31	3.56	0.00	-	-
5	4.95	2.01	-	-	-	-	-	-	-
6	1.30	3.79	4.10	5.98	3.17	1.58	-	-	-
7	3.32	4.31	5.45	3.77	4.40	1.66	3.68	0.00	0.00

Table 7.4 – Comparaison des valeurs de la divergence sémantique CSDivergence obtenues pour les différentes communautés détectées par chaque algorithme.

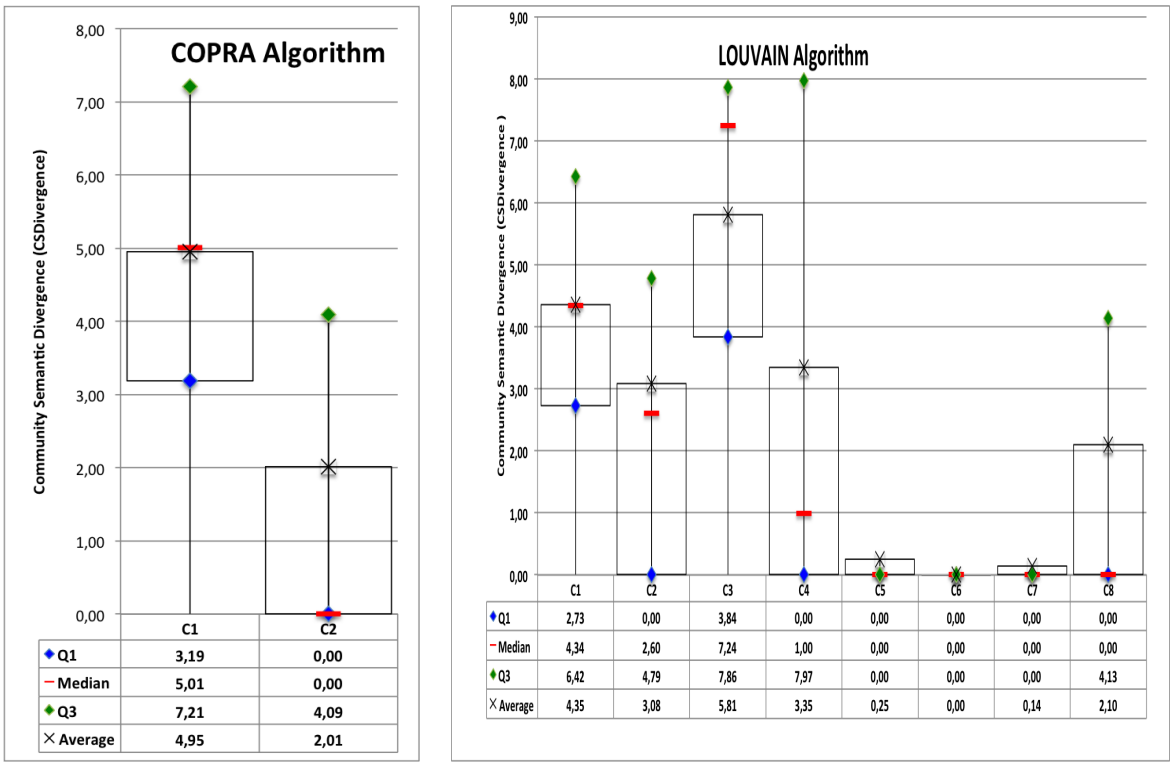
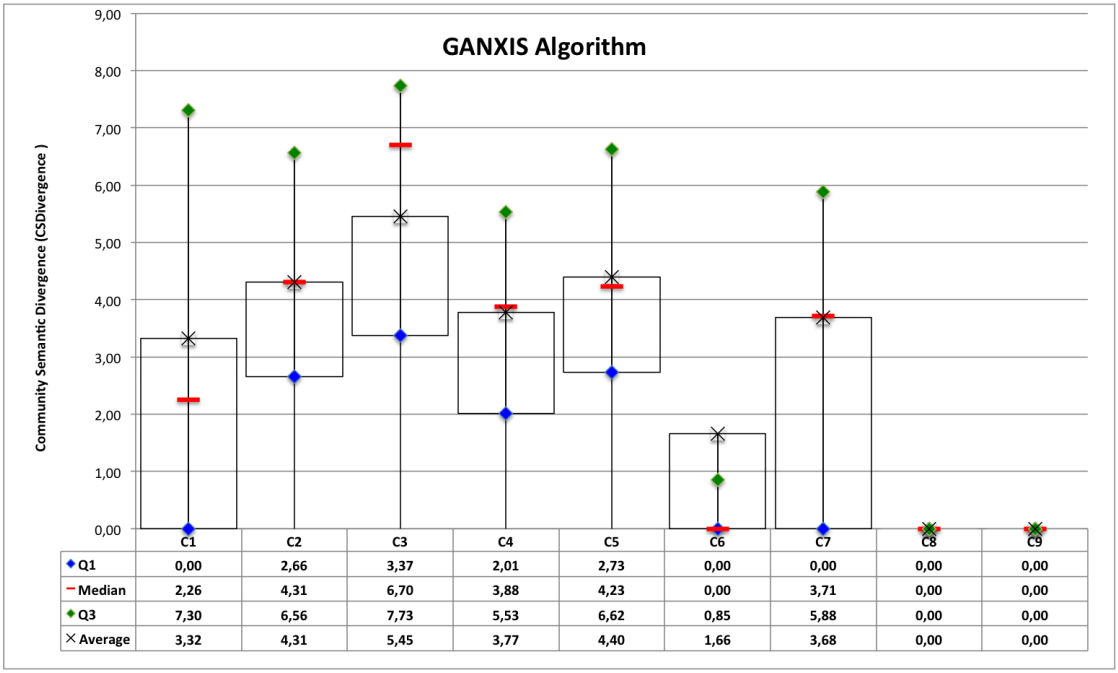


Figure 7.4 – La divergence sémantique des communautés détectées par COPRA et Louvain.



	C1	C2	C3	C4	C5	C6	C7	C8	C9
Q1	0,00	2,66	3,37	2,01	2,73	0,00	0,00	0,00	0,00
Median	2,26	4,31	6,70	3,88	4,23	0,00	3,71	0,00	0,00
Q3	7,30	6,56	7,73	5,53	6,62	0,85	5,88	0,00	0,00
Average	3,32	4,31	5,45	3,77	4,40	1,66	3,68	0,00	0,00

Figure 7.5 – La divergence sémantique des communautés détectées par l'algorithme GANXIS

connectés. Les figures 7.3, 7.4, 7.5 représentent respectivement les valeurs de la divergence sémantique obtenues pour les différents algorithmes considérés. Comme nous remarquons à partir de ces figures, nous avons représenté ces résultats sous forme de boîtes à moustaches (traduction de Box & Whiskers Plot). La représentation graphique des boîtes à moustaches a été proposée par Tukey [Tukey, 1977] pour représenter schématiquement la distribution d'une variable ou d'une mesure. Dans ces figures, Q1 indique la valeur du premier quartile, Median la valeur médiane, Q3 la valeur du troisième quartile et Average la valeur moyenne des divergences sémantiques des communautés détectées. Nous remarquons à partir de ces résultats, que les communautés détectées par les algorithmes CMN, MSG et COPRA sont assez grandes et ne sont pas sémantiquement homogènes sauf les plus petites, comme par exemple, la communauté 3 détectée par l'algorithme CMN qui contient seulement 33 noeuds. Cette situation est différente pour le cas de certains algorithmes qui détectent de nombreuse communautés. Par exemple, 50 % des communautés détectées par Louvain ont une taille moyenne (avec une plus petite, la communauté C6) et sont sémantiquement cohérentes. En effet, comme indiqué dans le tableau 7.2 la plupart des services de ces communautés appartiennent à un domaine spécifique (70.77% Geography pour C5, 88.46% Medical pour C6, 93.62% Travel pour C7).

D'une manière générale, nous remarquons que les communautés détectées par les

Mesures	CMN	LOUVAIN	MSG	GCE	COPRA	WALKTRAP	GANXIS
Purity	0.530	0.610	0.510	0.480	0.510	0.560	1.000
Entropy	0.600	0.430	0.600	0.450	0.610	0.490	1.000
GCQuality	0.212	0.348	0.204	0.264	0.199	0.286	0.000
GSDivergnce	2.620	2.415	2.754	4.413	4.079	3.136	2.955
GSCoherence	0.473	0.445	0.367	0.241	0.298	0.460	0.458

Table 7.5 – Comparaison des valeurs obtenues pour les différentes mesures de classement pour tous les algorithmes

différents algorithmes sont soit petites et homogènes ou grandes et divergentes sémantiquement. Le tableau 7.4 montre les valeurs moyennes de la divergence sémantique CSDivergence obtenues pour les différentes communautés détectées par chaque algorithme. Par exemple l'algorithme Ganxis détecte 3 petites communautés homogènes et 6 grandes communautés très divergentes.

7.5.5 Classement des algorithmes de détection de communautés

Nous avons évalué les algorithmes de détection de communautés selon deux critères. le premier critère se base sur les catégories ou domaines des services web. Le second critère se focalise sur la cohérence sémantique en considérant l'aspect thématique des services web.

Comme décrit dans la section 7.4.3, nous avons introduit une mesure d'évaluation pour chacun de ces critères qui nous permet de classer les algorithmes de détection de communautés. Plus précisément, un algorithme est dit bon si ces deux classements sont cohérents : si les communautés détectées ont une bonne qualité et sont sémantiquement cohérentes. Autrement dit, des valeurs élevées des deux mesures introduites précédemment *GCQuality* (voir équation 7.7) et *GSCoherence* (voir équation 7.10) traduisent un bon algorithme de détection de communautés. Dans le tableau 7.5, nous présentons les valeurs obtenues pour les différentes mesures permettant de classer les différents algorithmes de détection de communautés. En considérant les valeurs de *GCQuality*, les algorithmes peuvent être classés comme suit : Louvain, Walktrap, GCE, CMN, MSG, Copra, Ganxis. En effet, Louvain, Walktrap et GCE ont obtenu les plus fortes valeurs de *GCQuality*. Ceci s'explique par le fait que la plupart des communautés détectées par ces algorithmes ont tendance à regrouper des services provenant des domaines spécifiques (voir tableau 7.2). Ainsi, la notion de communauté reste plus riche que celle de domaine.

Le classement que nous avons trouvé en se basant sur la qualité est en accord avec l'étude menée par [Labatut, 2012]. Les auteurs de ce travail ont proposé une nouvelle mesure en modifiant la pureté. Le but était d'obtenir un outil permettant une discrimination plus pertinente des algorithmes de détection de communautés. L'ordonnancement obtenu par notre mesure pour les algorithmes ; Louvain, Walktrap, CMN et Copra est

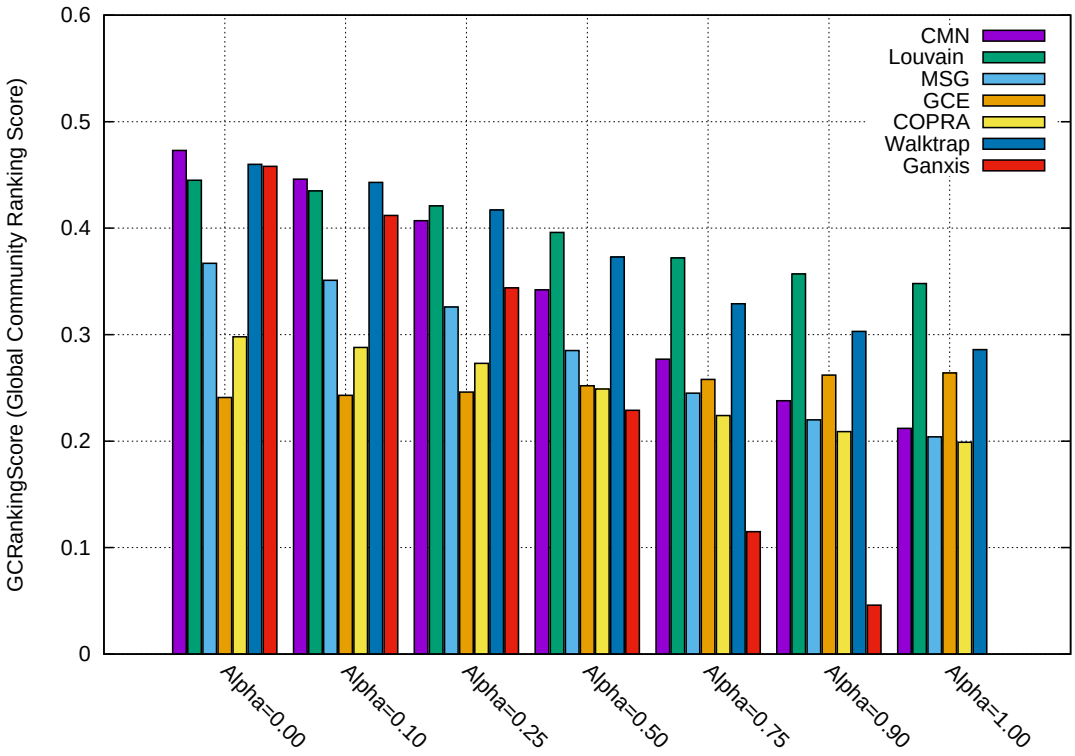


Figure 7.6 – Comparaison des valeurs obtenues du score GCRankingScore, avec différentes valeurs du paramètre α, pour tous les algorithmes.

identique à celui donné par cette étude. Le classement de cette dernière était à son tour cohérent avec une analyse approfondie de la topologie des structures de communautés estimées, qui avait été éffectuée par [Orman et al., 2012]. En se basant seulement sur les valeurs de la $GSCoherence$, nous pouvons avoir le classement suivant : CMN, Walktrap, Ganxis, Louvain, MSG, Copra, GCE.

Nous avons également calculé pour chaque algorithme un score global $GCRankingScore$ combinant les deux mesures $GCQuality$ et $GSCoherence$ comme décrit dans l'équation 7.10. Ce score permet de classer les différents algorithmes de détection de communautés. Nous rappelons qu'un score élevé $GCRankingScore$ traduit un bon algorithme de détection de communautés. La figure 7.6 représente les différentes valeurs obtenues pour le score $GCRankingScore$ tout en variant le paramètre d'ajustement α, pour tous les algorithmes. Le score $GCRankingScore$ évalue également la qualité et la cohérence sémantique globales des communautés détectées par chaque algorithme. Pour $\alpha = 1$, le score $GCRankingScore$ évalue que la qualité globale des communautés détectées par chaque algorithme. Pour $\alpha = 0$, nous évaluons que la cohérence sémantique globale des communautés détectées. Ainsi, si α est entre 0 et 0.50, notre approche favorise plus les communautés qui sont sémantiquement cohérentes. Quand α est entre 0.50 et 1, le critère

α	**Classement**	α	**Classement**	α	**Classement**
0.00	1. CMN 2. Walktrap 3. Ganxis 4. Louvain 5. MSG 6. COPRA 7. GCE	0.10	1. CMN 2. Walktrap 3. Louvain 4. Ganxis 5. MSG 6. COPRA 7. GCE	0.25	1. Louvain 2. Walktrap 3. CMN 4. Ganxis 5. MSG 6. COPRA 7. GCE
0.50	1. Louvain 2. Walktrap 3. CMN 4. MSG 5. GCE 6. COPRA 7. Ganxis	0.75	1. Louvain 2. Walktrap 3. CMN 4. GCE 5. MSG 6. COPRA 7. Ganxis	0.90	1. Louvain 2. Walktrap 3. GCE 4. CMN 5. MSG 6. COPRA 7. Ganxis
1.00	1. Louvain 2. Walktrap 3. GCE 4. CMN 5. MSG 6. COPRA 7. Ganxis	-	-	-	-

Table 7.6 – Classement des algorithmes selon les différentes valeurs du paramètre α.

de la qualité s'avère plus favorisé. $\alpha = 0.50$ permet de sélectionner des communautés ayant autant de qualité que de cohérence sémantique (voir figure 7.6). Le tableau 7.6 représente également les différents classements obtenus des algorithmes en fonction de différentes valeurs du paramètre α.

7.6 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle méthode pour classer les algorithmes de détection de communautés en combinant des mesures classiques telles que la pureté et l'entropie avec la modélisation thématiques des services. Tout d'abord, nous avons classé les communautés détectées par chaque algorithme selon chacun des critères. Les communautés dont la pureté est élevée et la valeur de l'entropie est petite sont considérées bonnes. Ensuite, nous avons classé les communautées de chaque algorithme selon leur divergence sémantique. Plus la divergence sémantique est petite, plus la communauté est sémantiquement cohérente. Nous avons également calculé un score global du classement pour chaque communauté en combinant le score de la cohérence sémantique

avec celui basé sur les mesures classiques (pureté et entropie). Notre approche permet donc d'évaluer la qualité et l'homogénéité sémantique des communautés détectées ainsi que le classement des algorithmes de détection de communautés.

Dans le chapitre suivant, nous allons conclure ce document par un résumé des travaux présentés dans cette thèse et par quelques perspectives.

8

Conclusion générale et perspectives

Ce chapitre conclue cette thèse en récapitulant les contributions et en évoquant également plusieurs directions de travaux futurs.

8.1 Conclusion

L'objectif principal de ce travail de thèse était de développer des méthodes basées sur la modélisation thématique probabiliste, l'analyse de concepts formels et les motifs fréquents, permettant d'améliorer la qualité des services web découverts.

Dans un premier temps, nous avons proposé une méthode pour construire une structure de services web sous forme de réseaux (i.e. réseaux d'interaction et réseaux de similitude). Dans le cas de réseaux d'interaction, nous avons considéré trois niveaux de granularité (paramètres, opérations et services). Nous avons dans un premier temps construit le réseau d'interaction d'opérations en calculant des treillis de concepts formels pour déduire ensuite le réseau d'interaction de services. Les réseaux d'interaction d'opérations se distinguent en fonction du mode d'invocation (totale ou partielle). Les réseaux d'interaction sont généralement utilisés pour identifier les liens de compositions entre les services web. Nous avons proposé également deux modèles de réseaux de similitude de services web qui se basent respectivement sur les descriptions syntaxiques et sémantiques des services web. Le modèle de réseau de similitude est conçu sur la base de la similitude entre les services web et est lié généralement à la classification des services web. Les réseaux de similitude peuvent être utilisés également pour identifier et construire des communautés de services web. Afin d'identifier la similarité entre deux services, nous avons utilisé quelques mesures classiques basées sur le calcul de la distance et/ou la similarité entre deux distributions de probabilités ou vecteurs de comptes.

Ensuite, nous avons proposé une méthode de découverte de services permettant de diversifier les résultats de la découverte tout en maintenant la qualité des services découverts. Cette méthode est à la fois basée sur la pertinence (similarité sémantique), la diversité et la densité des services pour minimiser la redondance dans la liste renvoyée à

l'utilisateur. Dans le cas de requêtes complexes ou ambiguës, il est nécessaire de combiner plusieurs services pour satisfaire ce genre de requêtes. Dans ce contexte, nous avons exploité le réseau d'interaction de services web construit et la notion de diversité dans les graphes pour identifier les services web qui sont susceptibles d'être composables. Notre méthode permet également l'optimisation et la diversification des liens de compositions identifiés. Les résultats des expérimentations effectuées ont montré que notre système de découverte est plus performant par rapport aux autres systèmes de la littérature considérés dans notre étude.

Puis, nous avons proposé un système de recommandation hybride basé sur le contenu et le filtrage collaboratif. L'originalité de la méthode proposée vient de la combinaison des modèles thématiques probabilistes et des motifs fréquents pour capturer la sémantique commune maximale d'un ensemble de services. L'objectif principal de notre méthode est d'identifier un ensemble de services web qui sont sémantiquement très liés par rapport à un service donné et ayant de bonnes qualités de services. Pour cela, nous avons introduit la notion de motifs sémantiques. Ces motifs sémantiques correspondent à des motifs fréquents maximaux de thèmes qui sont utilisés ensuite pour extraire un ensemble de motifs de services. Dans notre approche, nous avons construit un treillis de concepts fréquents pour calculer les motifs sémantiques et les ensembles correspondants de services. Afin d'effectuer des recherches rapides par le système de recommandation en se basant sur des index, nous avons proposé de stocker les ensembles de services résultats dans une structure spéciale, appelée MFI-tree. Nous avons également introduit une nouvelle mesure de classement permettant de mesurer le degré de pertinence des services à recommander par rapport à un service demandé. Cette mesure combine la similarité sémantique et un ensemble d'attributs de la qualité de services. Les résultats des évaluations effectuées ont montré que notre système de recommandation obtient de meilleurs résultats par rapport aux autres méthodes de la littérature considérés.

Enfin, nous avons étudié la structure communautaire des réseaux de services au lieu de considérer seulement des services individuels pour la recommandation. Dans ce contexte, nous avons proposé une nouvelle méthode qui combine la sémantique et la topologie dans les réseaux afin d'évaluer la qualité et la cohérence sémantique des communautés détectées. D'abord, nous avons évalué la qualité d'une communauté en utilisant les mesures de pureté et d'entropie. Ensuite, la cohérence sémantique d'une communauté a été évaluée en introduisant une nouvelle mesure appelée la divergence sémantique des communautés détectées. Enfin, pour évaluer et classer les algorithmes de détection de communautés, nous avons introduit une nouvelle mesure qui combine le score de la cohérence sémantique (basé sur la divergence sémantique) avec celui basé sur les mesures classiques (pureté et entropie). Plusieurs expérimentations ont été réalisées pour évaluer les algorithmes de détection de communautés en considérant les différentes mesures introduites. Les résultats obtenus montrent que la majorité des algorithmes évalués

ignorent complètement la sémantique et par conséquent détectent des communautés non cohérentes sémantiquement.

8.2 Perspectives

Dans cette section, nous présentons un ensemble de perspectives futures.

Amélioration du portail de services WS-Portal : nous envisageons de faire évoluer le portail de services web WS-Portal [Aznag et al., 2015] développé au sein de notre équipe en implémentant et intégrant les différentes méthodes proposées dans cette thèse : (1) diversification des résultats de la découverte des services, (2) recommandation de services, (3) découverte de compositions possibles de services et (4) détection de communautés de services. Nous envisageons également de mettre en place un système interactif permettant à l'utilisateur de naviguer dans les résultats de la composition tout en offrant la possibilité de sélectionner une séquence de services et de les invoquer automatiquement. Actuellement, le portail ne permet de gérer que les services web décrits par des documents WSDL. Nous envisageons également d'intégrer dans ce portail, la gestion des services web ou APIs REST (publication, découverte, invocation, ...).

Approximation de motifs fréquents et recommandation de services web : comme nous l'avons déja évoqué dans cette thèse, la découverte des motifs fréquents est une tâche très importante et peut être appliquée à la recommandation de services. Nous envisageons d'étendre la méthode de recommandation de services proposée dans cette thèse en se basant sur l'approximation de motifs fréquents. Nous envisageons donc d'étudier des algorithmes existants de calcul approché de motifs fréquents [Durand & Quafafou, 2014, Durand & Quafafou, 2016] et/ou de développer un nouvel algorithme. Nous avons déjà commencé à expérimenter et évaluer quelques algorithmes d'approximation de motifs fréquents. D'après nos premiers résultats obtenus, une approximation directe des motifs de services n'est pas souhaitable. Il faut d'abord passer par une approximation des motifs de thèmes (i.e. motifs sémantiques). La représentation condensée par motifs fréquents maximaux permet de réduire le nombre de motifs produits. En revanche, le calcul de ce genre de motifs devient difficile pour un gros volume de données car le nombre de motifs produits peut être très élevé dans ce cas. Par conséquent, l'approximation peut être une solution intéressante pour réduire le nombre de motifs.

Découverte de communautés homogènes de services web : les algorithmes de détection de communautés proposés dans la littérature sont nombreux. Cependant, les expérimentations réalisées dans cette thèse (voir chapitre 7) montrent que la majorité des algorithmes évalués détectent des communautés non cohérentes sémantiquement. Pour remédier à cette problématique, nous envisageons de proposer

une nouvelle méthode permettant la découverte de communautés homogènes et cohérentes sémantiquement. Dans cette thèse, nous avons utilisé les thèmes extraits en utilisant le modèle thématique probabiliste CTM pour évaluer la cohérence sémantique des communautés détectées. Nous envisageons d'utiliser aussi l'information sémantique capturée par le biais des thèmes et plus précisément la corrélation des thèmes, pour identifier un ensemble de communautés homogènes. En effet, en plus de l'extraction des thèmes, le modèle CTM permet également la modélisation des corrélations entre les thèmes [Blei & Lafferty, 2007]. L'idée est d'utiliser un algorithme classique de détection de communauté (par exemple l'algorithme Louvain) pour détecter un ensemble de communautés de services en se basant sur la structure topologique dans les réseaux de services. Ensuite, nous pouvons re-classer les services des communautés hétérogènes et de mauvaises qualité, en exploitant la notion de corrélation des thèmes et le mécanisme du clustering hiérarchique [Aznag et al., 2014].

Composition de services web : nous envisageons aussi pour la suite de cette thèse, de réaliser des expérimentations supplémentaires afin d'évaluer d'avantage la méthode d'identification de compositions (temps de réponse, pertinence ...). Comme la méthode d'identification de possibles compositions, présentée dans le chapitre 5 (voir équation 5.9, page 113), se concentre principalement sur le voisinage immédiat dans le réseau d'interaction de services, nous envisageons d'étudier plus en détail le voisinage le plus proches (voir équation 5.10, page 114). Des expérimentations supplémentaires seront réalisés en tenant compte des différentes valeurs du paramètre λ présent respectivement dans les deux équations 5.9 et 5.10. Nous envisageons également de proposer une méthode permettant de construire un réseau d'interaction de services REST et d'identifier de possibles compositions des services ou APIs REST.

Bibliographie

[A. Abdullah, 2016] A. Abdullah, X. L. (2016). An efficient similarity-based model for web service recommendation. *International Journal of Services Computing (IJSC)*, 4(3), 15–28. (Cité en page 52.)

[Agrawal et al., 1993] Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining Association Rules between Sets of Items in Large Database. *ACM SIGMOD International Conference on Management of Data*, (pp. 207–216). (Cité en page 39.)

[Agrawal & Srikant, 1994] Agrawal, R. & Srikant, R. (1994). Fast Algorithms for Mining Association Rules. In *Proceedings of the 20th VLDB Conference* Santiago, Chile. (Cité en page 64.)

[Albert & Barabási, 2002] Albert, R. & Barabási, A.-L. (2002). Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74(1), 47–97. (Cité en page 110.)

[Andrews et al., 2003] Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., & Weerawarana, S. (2003). *BPEL4WS, Business Process Execution Language for Web Services Version 1.1.* IBM. (Cité en page 61.)

[Atkinson et al., 2007] Atkinson, C., Bostan, P., Hummel, O., & Stoll, D. (2007). A practical approach to web service discovery and retrieval. In *ICWS* (pp. 241–248). : IEEE Computer Society. (Cité en page 50.)

[Azmeh et al., 2010] Azmeh, Z., Huchard, M., Tibermacine, C., Urtado, C., & Vauttier, S. (2010). Using concept lattices to support web service compositions with backup services. In *Internet and Web Applications and Services (ICIW), 2010 Fifth International Conference on* (pp. 363–368). (Cité en page 53.)

[Aznag, 2015] Aznag, M. (Juillet 2015). *Modélisation thématique probabiliste des services web.* Thèse de doctorat, Université Aix-Marseille. (Cité en pages 5, 25, 27, 29 et 136.)

[Aznag et al., 2013a] Aznag, M., Quafafou, M., & Jarir, Z. (2013a). Correlated topic model for web services ranking. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 4(6), 283–291. (Cité en pages 53, 54 et 101.)

[Aznag et al., 2014] Aznag, M., Quafafou, M., & Jarir, Z. (2014). Leveraging formal concept analysis with topic correlation for service clustering and discovery. In *21th IEEE International Conference on Web Services (ICWS 2014)*. (Cité en pages 6, 52, 53, 54, 101, 129 et 176.)

[Aznag et al., 2015] Aznag, M., Quafafou, M., & Jarir, Z. (2015). *WS-Portal an Enriched Web Services Search Engine*, (pp. 409–412). Springer International Publishing : Cham. (Cité en pages 97, 136 et 175.)

[Aznag et al., 2013b] Aznag, M., Quafafou, M., Rochd, E. M., & Jarir, Z. (2013b). Probabilistic topic models for web services clustering and discovery. In W. L. K.-K. Lau & E. P. (Eds.) (Eds.), *European Conference on Service-Oriented and Cloud Computing (ESOCC 2013)*, LNCS 8135 (pp. 19–33). (Cité en pages 6, 27, 52, 53 et 54.)

[Bache et al., 2013] Bache, K., Newman, D., & Smyth, P. (2013). Text-based measures of document diversity. In I. S. Dhillon, Y. Koren, R. Ghani, T. E. Senator, P. Bradley, R. Parekh, J. He, R. L. Grossman, & R. Uthurusamy (Eds.), *KDD* (pp. 23–31). : ACM. (Cité en pages 6, 100 et 103.)

[Bansal et al., 2005] Bansal, A., Kona, S., Simon, L., & Hite, T. D. (2005). A universal service-semantics description language. In *ECOWS* (pp. 214–225). : IEEE Computer Society. (Cité en page 50.)

[Battle & Benson, 2008] Battle, R. & Benson, E. (2008). Bridging the semantic web and web 2.0 with representational state transfer (rest). *J. Web Sem.*, 6(1), 61–69. (Cité en page 50.)

[Benatallah et al., 2005] Benatallah, B., Dijkman, R., Dumas, M., & Maamar, Z. (2005). Service composition : concepts, techniques, tools and trends. In Z. Stojanovic & A. Dahanayake (Eds.), *Service-Oriented Software System Engineering : Challenges and Practices* (pp. 48–66). Hershey : Idea Group Publishing. (Cité en pages 4 et 60.)

[Bentahar et al., 2007] Bentahar, J., Maamar, Z., Benslimane, D., & Thiran, P. (2007). An argumentation framework for communities of web services. *IEEE Intelligent Systems*, 22(6), 75–83. (Cité en page 62.)

[Bentahar et al., 2008] Bentahar, J., Maamar, Z., Wan, W., Benslimane, D., Thiran, P., & Subramanian, S. (2008). Agent-based communities of web services : an argumentation-driven approach. *Service Oriented Computing and Applications*, 2(4), 219–238. (Cité en page 62.)

[Birkhoff, 1967] Birkhoff, G. (1967). Lattice theory. In *Colloquium Publications*, volume 25. Amer. Math. Soc., 3. edition. (Cité en pages 7, 31, 37 et 43.)

[Blake & Nowlan, 2007] Blake, M. B. & Nowlan, M. F. (2007). A web service recommender system using enhanced syntactical matching. In *ICWS* (pp. 575–582). : IEEE Computer Society. (Cité en page 56.)

[Blei & Jordan, 2003] Blei, D. & Jordan, M. (2003). Modeling annotated data. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 127–134). : ACM Press. (Cité en page 27.)

[Blei & Lafferty, 2007] Blei, D. & Lafferty, J. (2007). A correlated topic model of science. *Annals of Applied Statistics*, 1, 17–35. (Cité en pages 5, 27, 29, 30, 31, 53, 64, 76, 128, 157 et 176.)

[Blei et al., 2003] Blei, D., Ng, A., & Jordan, M. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3, 993–1022. (Cité en pages 27, 29 et 87.)

[Blondel et al., 2008] Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics : Theory and Experiment*, 2008(10), P10008. (Cité en page 153.)

[Blum, 2004] Blum, A. (2004). Uddi as an extended web services registry : Versioning, quality of service, and more. *White paper, SOA World magazine*, 4(6). (Cité en page 23.)

[Bordat, 1986] Bordat, J. P. (1986). Calcul pratique du treillis de galois d'une correspondance. *Informatiques et Sciences Humaines*, 96, 31–47. (Cité en page 66.)

[Boulicaut & Bykowski, 2000] Boulicaut, J. F. & Bykowski, A. (2000). Frequent Closures as Concise Representation for Binary Data Mining. In Springer-Verlag (Ed.), *Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'00)* (pp. 18–20). Kyoto, Japan. (Cité en pages 42 et 43.)

[Boulicaut et al., 2003] Boulicaut, J.-F., Bykowski, A., & Rigotti, C. (2003). Free-sets : a condensed representation of boolean data for the approximation of frequency queries. *Data Mining and Knowledge Discovery*, 7(1), 5–22. (Cité en page 45.)

[Brandes & Erlebach, 2005] Brandes, U. & Erlebach, T. (2005). *Network Analysis : Methodological Foundations*. Springer. (Cité en page 152.)

[Brandes et al., 2008] Brandes, U., Gaertler, M., & Wagner, D. (2008). Engineering graph clustering : Models and experimental evaluation. *ACM Journal of Experimental Algorithmics*, 12(1.1). (Cité en page 153.)

[Burdick et al., 2005] Burdick, D., Calimlim, M., Flannick, J., Gehrke, J., & Yiu, T. (2005). Mafia : A maximal frequent itemset algorithm. *IEEE Trans. Knowl. Data Eng.*, 17(11), 1490–1504. (Cité en page 42.)

[Burdick et al., 2001] Burdick, D., Calimlim, M., & Gehrke, J. (2001). Mafia : A maximal frequent itemset algorithm for transactional databases. In *In ICDE* (pp. 443–452). (Cité en page 65.)

[Burstein et al., 2004] Burstein, M., Hobbs, J., Lassila, O., Mcdermott, D., Mcilraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., & Sycara, K. (2004). Owl-s : Semantic markup for web services. Website. (Cité en page 61.)

[Calders & Goethals, 2007] Calders, T. & Goethals, B. (2007). Non-derivable itemset mining. *Data Mining and Knowledge Discovery*, 14(1), 171–206. (Cité en page 42.)

[Carbonell & Goldstein, 1998] Carbonell, J. G. & Goldstein, J. (1998). The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Research and Development in Information Retrieval* (pp. 335–336). (Cité en pages 54, 112 et 117.)

[Carpineto & Romano, 1996] Carpineto, C. & Romano, G. (1996). A lattice conceptual clustering system and its application to browsing retrieval. *Machine Learning*, 24(2), 95–122. (Cité en page 66.)

[Cassar et al., 2013] Cassar, G., Barnaghi, P., & Moessner, K. (2013). Probabilistic matchmaking methods for automated service discovery. *IEEE Transactions on Services Computing*, PP(99), 1–1. (Cité en pages 52 et 53.)

[Chein, 1969] Chein, M. (1969). Algorithme de recherche des sous-matrices premières d'une matrice. *Bull. Math. Soc. Sc. Math. de Roumanie*, 1(13), 21–25. (Cité en page 66.)

[Chen et al., 2013] Chen, L., Wang, Y., Yu, Q., Zheng, Z., & Wu, J. (2013). Wt-lda : User tagging augmented lda for web service clustering. In S. Basu, C. Pautasso, L. Zhang, & X. Fu (Eds.), *ICSOC*, volume 8274 of *Lecture Notes in Computer Science* (pp. 162–176). : Springer. (Cité en page 53.)

[Cheng et al., 2015] Cheng, J., Liu, C., Zhou, M., Zeng, Q., & Ylä-Jääski, A. (2015). Automatic composition of semantic web services based on fuzzy predicate petri nets. *IEEE Trans. Automation Science and Engineering*, 12(2), 680–689. (Cité en pages 6 et 61.)

[Cherifi et al., 2010] Cherifi, C., Labatut, V., & Santucci, J. F. (2010). Benefits of semantics on web service composition from a complex network perspective. In F. Zavoral, J. Yaghob, P. Pichappan, & E. El-Qawasmeh (Eds.), *NDT (2)*, volume 88 of *Communications in Computer and Information Science* (pp. 80–90). : Springer. (Cité en page 59.)

[Cherifi et al., 2013a] Cherifi, C., Labatut, V., & Santucci, J. F. (2013a). On flexible web services composition networks. *CoRR*, abs/1305.0688. (Cité en pages 59 et 61.)

[Cherifi et al., 2013b] Cherifi, C., Labatut, V., & Santucci, J. F. (2013b). Topological properties of web services similarity networks. *CoRR*, abs/1305.0196. (Cité en page 59.)

[Cherifi et al., 2011] Cherifi, C., Rivierre, Y., & Santucci, J.-F. (2011). WS-NEXT, a Web Services Network Extractor Toolkit. In A.-D. Ali (Ed.), *The 5th International Conference on Information Technology*, ISBN 9957-8583-0-0 (pp. 91– 96). amman, Jordan : Al-Zaytoonah University of Jordan. (Cité en pages 59 et 61.)

[Chollet et al., 2011] Chollet, S., Lestideau, V., Lalanda, P., Maurel, Y., Colomb, P., & Raynaud, O. (2011). Building fca-based decision trees for the selection of heterogeneous services. In H.-A. Jacobsen, Y. Wang, & P. Hung (Eds.), *IEEE SCC* (pp. 616–623). : IEEE. (Cité en pages 52 et 53.)

[Christensen et al., 2001] Christensen, E., Meredith, F. C. G., & Weerawarana, S. (2001). *Web Services Description Language (WSDL) 1.1, http ://www.w3.org/TR/wsdl*. Technical report, The World Wide Web Consortium (W3C). (Cité en page 48.)

[Chu et al., 2016] Chu, V. W., Wong, R. K., Chen, F., & Chi, C. H. (2016). Service selection based on dynamic qos networks. In *2016 IEEE International Conference on Services Computing (SCC)* (pp. 98–105). (Cité en page 58.)

[Clauset et al., 2004] Clauset, A., Newman, M. E. J., & Moore, C. (2004). Finding community structure in very large networks. *Physical Review E*, 70, 066111. (Cité en page 154.)

[Cole & Eklund, 2001] Cole, R. J. & Eklund, P. W. (2001). Browsing semi-structured web texts using formal concept analysis. In H. S. Delugach & G. Stumme (Eds.), *Proceedings of the 9th International Conference on Conceptual Structures (ICCS 2001)*, volume 2120 of *Lecture Notes in Computer Science* (pp. 319–332). : Springer. (Cité en page 34.)

[de Bruijn & Lausen, 2005] de Bruijn, J. & Lausen, H. (June 2005). *Web Service Modeling Language (WSML). http ://www.w3.org/Submission/WSML/.* Technical report, W3C Member Submission. (Cité en page 49.)

[Dekar & Kheddouci, 2008] Dekar, L. & Kheddouci, H. (2008). A graph b-coloring based method for composition-oriented web services classification. In A. An, S. Matwin, Z. W. Ras, & D. Slezak (Eds.), *ISMIS*, volume 4994 of *Lecture Notes in Computer Science* (pp. 599–604). : Springer. (Cité en page 58.)

[Delugach & Stumme, 2001] Delugach, H. & Stumme, G., Eds. (2001). *Conceptual Structures – Broadening the Base. Proc. 9th International Conference on Conceptual Structures*, volume 2120 of *LNAI*, Heidelberg. Springer. (Cité en page 66.)

[Dempster et al., 1977] Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistcal Society*, 39(B), 1–38. (Cité en page 31.)

[Driss et al., 2010] Driss, M., Moha, N., Jamoussi, Y., Jézéquel, J.-M., & Ghézala, H. H. B. (2010). A requirement-centric approach to web service modeling, discovery, and selection. In P. P. Maglio, M. Weske, J. Yang, & M. Fantinato (Eds.), *ICSOC*, volume 6470 of *Lecture Notes in Computer Science* (pp. 258–272). (Cité en page 53.)

[Durand & Quafafou, 2014] Durand, N. & Quafafou, M. (2014). Approximation of Frequent Itemset Border by Computing Approximate Minimal Hypergraph Transversals. In *Proceedings of the 16th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2014)* (pp. 357–368). Munich, Germany. (Cité en page 175.)

[Durand & Quafafou, 2016] Durand, N. & Quafafou, M. (2016). Frequent Itemset Border Approximation by Dualization. *Transactions on Large-Scale Data- and Knowledge-Centered Systems (TLDKS)*, 26, 32–60. (Cité en page 175.)

[Elgazzar et al., 2010] Elgazzar, K., Hassan, A. E., & Martin, P. (2010). Clustering wsdl documents to bootstrap the discovery of web services. In *IEEE International Conference on Web Services (ICWS)* (pp. 147–154). : IEEE Computer Society. (Cité en page 52.)

[Fang et al., 2006] Fang, W., Moreau, L., Ananthakrishnan, R., Wilde, M., & Foster, I. T. (2006). Exposing uddi service descriptions and their metadata annotations as ws-resources. In *GRID* (pp. 128–135). : IEEE. (Cité en page 52.)

[Farrell & Lausen, 2007] Farrell, J. & Lausen, H. (August 2007). *Semantic Annotations for WSDL and XML Schema. http ://www.w3.org/TR/sawsdl/.* Technical report, W3C Recommendation. (Cité en page 50.)

[Fensel & Bussler, 2002] Fensel, D. & Bussler, C. (2002). *The Web Service Modeling Framework (WSMF).* Technical report, Vrije Universiteit Amsterdam. (Cité en page 50.)

[Fethallah et al., 2010] Fethallah, H., Chikh, A., & Belabed, A. (2010). Automated discovery of web services : an interface matching approach based on similarity measure. In A. Alnsour & S. Aljawarneh (Eds.), *ISWSA* (pp.1̃3). : ACM. (Cité en pages 52 et 56.)

[Fielding, 2000] Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures.* PhD thesis, Thèse de doctorat, University of California, Irvine. (Cité en page 49.)

[Fortunato, 2010] Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, 486, 75–174. (Cité en pages 5, 62, 151 et 153.)

[Ganter, 1984] Ganter, B. (1984). *Two basic algorithms in concept analysis.* FB4-Preprint 831, TH Darmstadt. (Cité en page 66.)

[Ganter & Wille, 1999] Ganter, B. & Wille, R. (1999). *Formal Concept Analysis : Mathematical Foundations.* Berlin/Heidelberg : Springer. (Cité en pages 31, 34 et 43.)

[Gekas & Fasli, 2007] Gekas, J. & Fasli, M. (2007). Employing graph network analysis for web service composition. *IJITWE*, 2(4), 21–40. (Cité en page 58.)

[Gibbs & Su, 2002] Gibbs, A. L. & Su, F. E. (2002). On choosing and bounding probability metrics. *International Statistical Review*, 70, 419–435. (Cité en page 78.)

[Goldberg et al., 1992] Goldberg, D., Nicols, D., Oki, B., & Terry, D. (1992). Using Collaborative Filtering to Weave an Information Tapestry. *Communications of the ACM*, 35(12), 61–70. (Cité en page 55.)

[Gollapudi & Sharma, 2009] Gollapudi, S. & Sharma, A. (2009). An axiomatic approach for result diversification. In J. Quemada, G. León, Y. S. Maarek, & W. Nejdl (Eds.), *WWW* (pp. 381–390). : ACM. (Cité en page 54.)

[GOMADAM et al., 2010] GOMADAM, K., RANABAHU, A., & SHETH, A. (2010). Towards a restful service ecosystem : Perspectives and challenges. In *4th IEEE International Conference on Digital Ecosystems and Technologies.* (Cité en pages xv, 50 et 51.)

[Gouda & Zaki, 2005] Gouda, K. & Zaki, M. J. (2005). Genmax : An efficient algorithm for mining maximal frequent itemsets. *Data Min. Knowl. Discov.*, 11(3), 223–242. (Cité en page 65.)

[Grahne & Zhu, 2003] Grahne, G. & Zhu, J. (2003). Efficiently using prefix-trees in mining frequent itemsets. In B. Goethals & M. J. Zaki (Eds.), *FIMI*, volume 90 of *CEUR Workshop Proceedings* : CEUR-WS.org. (Cité en page 65.)

[Grahne & Zhu, 2005] Grahne, G. & Zhu, J. (2005). Fast algorithms for frequent itemset mining using fp-trees. *IEEE Trans. Knowl. Data Eng.*, 17(10), 1347–1362. (Cité en pages 8, 65, 125 et 132.)

[Haas, 2005] Haas, H. (2005). Reconciling web services and rest services (keynote address). In *Proc. of the 3rd IEEE European Conference on Web Services (ECOWS 2005), Växjö, Sweden*. (Cité en page 49.)

[Han et al., 2000] Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. In *Proc. ACM SIGMOD Int'l Conf. on Management of Data* (pp. 1–12). (Cité en page 64.)

[Hassen et al., 2008] Hassen, R. R., Nourine, L., & Toumani, F. (2008). Protocol-based web service composition. In A. Bouguettaya, I. Krüger, & T. Margaria (Eds.), *ICSOC*, volume 5364 of *Lecture Notes in Computer Science* (pp. 38–53). (Cité en page 60.)

[Hofmann, 1999] Hofmann, T. (1999). Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence* (pp. 289–296). : Morgan Kaufmann Publishers Inc. (Cité en pages 27 et 53.)

[Hofmann, 2003] Hofmann, T. (2003). Collaborative filtering via gaussian probabilistic latent semantic analysis. In *SIGIR* (pp. 259–266). : ACM. (Cité en page 27.)

[Hofmann & Puzicha, 1998] Hofmann, T. & Puzicha, J. (1998). *Unsupervised Learning from Dyadic Data*. Technical Report TR-98-042, International Computer Science Insitute, Berkeley, CA. (Cité en page 27.)

[Hu, 2003] Hu, M. (2003). Web services composition, partition, and quality of service in distributed system integration and re-engineering. In *XML Conference* (pp. 1–9). : Citeseer. (Cité en page 60.)

[Ian Horrocks, 2004] Ian Horrocks, Peter F. Patel-Schneider, H. B. S. T. B. G. M. D. (May 2004). *SWRL : A Semantic Web Rule Language Combining OWL and RuleML. http ://www.w3.org/Submission/SWRL/.* Technical report, W3C Member Submission. (Cité en page 50.)

[Iwata et al., 2008] Iwata, T., Yamada, T., & Ueda, N. (2008). Probabilistic latent semantic visualization : topic model for visualizing documents. In Y. Li, B. L. 0001, & S. Sarawagi (Eds.), *KDD* (pp. 363–371). : ACM. (Cité en page 27.)

[Jones, 2005] Jones, S. (2005). Toward an acceptable definition of service. *IEEE Software*, 22(3), 87–93. (Cité en page 3.)

[Järvelin, 2002] Järvelin, K. (2002). Cumulated gain-based evaluation of ir techniques. volume 20 (pp. 2002). (Cité en page 116.)

[Kang et al., 2016] Kang, G., Tang, M., Liu, J., Liu, X. F., & Cao, B. (2016). Diversifying web service recommendation results via exploring service usage history. *IEEE Trans. Services Computing*, 9(4), 566–579. (Cité en page 55.)

[Kantor, 2009] Kantor, P. B. (2009). *Recommender systems handbook*. New York ; London : Springer. (Cité en page 55.)

[Karim, 2014] Karim, J. (2014). Hybrid system for personalized recommendations. In *2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS)* (pp. 1–6). (Cité en page 57.)

[Kavantzas et al., 2004] Kavantzas, N., Burdett, D., Ritzinger, G., Fletcher, T., & Lafon, Y. (2004). Web Services Choreography Description Language, Version 1.0. W3C Working Draft 17-12-04. (Cité en page 61.)

[Keller et al., 2005] Keller, U., Lara, R., Lausen, H., Polleres, A., Predoiu, L., & Toma, O. (2005). *Semantic Web Service Discovery*. Technical Report WSMX Deliverable D10, DERI Innsbruck. (Cité en page 3.)

[Kernighan & Lin, 1970] Kernighan, B. & Lin, S. (1970). An Efficient Heuristic Procedure for Partitioning Graphs. *The Bell Systems Technical Journal*, 49(2). (Cité en page 153.)

[Kil et al., 2009] Kil, H., Oh, S.-C., Elmacioglu, E., Nam, W., & Lee, D. (2009). Graph theoretic topological analysis of web service networks. *World Wide Web*, 12(3), 321–343. (Cité en page 58.)

[Klusch, 2014] Klusch, M. (2014). Service discovery. In R. Alhajj & J. Rokne (Eds.), *Encyclopedia of Social Network Analysis and Mining* (pp. 1707–1717). Springer New York. (Cité en page 51.)

[Klusch & Kapahnke, 2012] Klusch, M. & Kapahnke, P. (2012). The isem matchmaker : A flexible approach for adaptive hybrid semantic service selection. *J. Web Sem.*, 15, 1–14. (Cité en page 52.)

[Klusch et al., 2009] Klusch, M., Kapahnke, P., & Zinnikus, I. (2009). Hybrid adaptive web service selection with sawsdl-mx and wsdl-analyzer. In *Proceedings of the 6th European Semantic Web Conference on The Semantic Web (ESWC'09)* (pp. 550–564). Berlin, Heidelberg : Springer-Verlag. (Cité en page 117.)

[Klusch et al., 2010] Klusch, M., Kapahnke, P., & Zinnikus, I. (2010). Adaptive hybrid semantic selection of sawsdl services with sawsdl-mx2. *Int. J. Semantic Web Inf. Syst.*, 6(4), 1–26. (Cité en page 52.)

[Klusch & Kaufer, 2009] Klusch, M. & Kaufer, F. (2009). Wsmo-mx : A hybrid semantic web service matchmaker. *Web Intelligence and Agent Systems*, 7(1), 23–42. (Cité en page 52.)

[Kopecky, 2012] Kopecky, J. (2012). *Web Service Automation Supported by Lightweight Semantic Annotations*. PhD thesis, Thèse de doctorat, Faculty of Mathematics, Computer Science and Physics of the University of Innsbruck. (Cité en page 49.)

[Kopecky et al., 2008] Kopecky, J., Vitvar, T., & Fensel, D. (2008). Microwsmo : Semantic description of restful services. *Available : http ://wsmo.org/TR/d38/v0.1/20080219/d38v01*$_2$0080219*.pdf*, 1. (*Citenpage* 50.)

[Kullback & Leibler, 1951] Kullback, S. & Leibler, R. A. (1951). On information and sufficiency. *Ann. Math. Statist.*, 22(1), 79–86. (Cité en pages 105 et 157.)

[Labatut, 2012] Labatut, V. (2012). Une nouvelle mesure pour l'évaluation des méthodes de détection de communautés. In *Actes de 3ième Conférence sur les Modèles et Analyses Réseau : Approches Mathématiques et Informatiques, MARAMI'12*. (Cité en page 169.)

[Lahcen & Kwuida, 2010] Lahcen, B. & Kwuida, L. (2010). Lattice miner : A tool for concept lattice construction and exploration. In *In Suplementary Proceeding of International Conference on Formal concept analysis (ICFCA'10)*. (Cité en page 88.)

[Lausen & Haselwanter, 2007] Lausen, H. & Haselwanter, T. (2007). Finding web services. In *In Proc. of the 1st European Semantic Technology Conference (ESTC*. (Cité en pages 50 et 52.)

[Lécué, 2010] Lécué, F. (2010). Combining collaborative filtering and semantic content-based approaches to recommend web services. In *ICSC* (pp. 200–205). : IEEE Computer Society. (Cité en page 57.)

[Lécué & Delteil, 2007] Lécué, F. & Delteil, A. (2007). Making the difference in semantic web service composition. In *AAAI* (pp. 1383–1388). : AAAI Press. (Cité en page 56.)

[Lee et al., 2010] Lee, C., Reid, F., McDaid, A., & Hurley, N. (2010). Detecting highly overlapping community structure by greedy clique expansion. In *Workshop on Social Network Mining and Analysis*. cite arxiv :1002.1827 Comment : 10 pages, 7 Figures. Implementation source and binaries available at http ://-sites.google.com/site/greedycliqueexpansion/. (Cité en page 154.)

[Lee et al., 2006] Lee, K., Jeon, J., Lee, W., Jeong, S.-H., & Park, S.-W. (May 15, 2006). *QoS for Web Services : Requirements and Possible Approaches*. Technical report, W3C, Web Services Architecture Working Group (2003). (Cité en page 23.)

[Leskovec et al., 2010] Leskovec, J., Lang, K. J., & Mahoney, M. W. (2010). Empirical comparison of algorithms for network community detection. cite arxiv :1004.3539. (Cité en page 151.)

[Li & Yu, 2013] Li, R.-H. & Yu, J. X. (2013). Scalable diversified ranking on large graphs. *IEEE Trans. Knowl. Data Eng.*, 25(9), 2133–2146. (Cité en pages 59, 112 et 113.)

[Liu et al., 2015a] Liu, B., Wang, C., Wang, C., & Wang, Y. (2015a). A novel algorithm for finding overlapping communities in networks based on label propagation. In Y. Tan, Y. Shi, F. B. de Lima Neto, A. F. Gelbukh, S. Das, & A. P. Engelbrecht (Eds.), *ICSI (2)*, volume 9141 of *Lecture Notes in Computer Science* (pp. 325–332). : Springer. (Cité en page 154.)

[Liu et al., 2007] Liu, J., Liu, J., & Chao, L. (2007). Design and implementation of an extended uddi registration center for web service graph. In *ICWS* (pp. 1174–1175). : IEEE Computer Society. (Cité en page 58.)

[Liu et al., 2015b] Liu, L., Xu, L., Wang, Z., & Chen, E. (2015b). Community detection based on structure and content : A content propagation perspective. In C. Aggarwal, Z.-H. Zhou, A. Tuzhilin, H. Xiong, & X. Wu (Eds.), *ICDM* (pp. 271–280). : IEEE Computer Society. (Cité en page 151.)

[Liu & Wong, 2008] Liu, W. & Wong, W. (2008). Discovering homogeneous service communities through web service clustering. In *Proceedings of the AAMAS Workshop on Service-Oriented Computing : Agents, Semantics, and Engineering (SOCASE)* Estoril, Portugal. (Cité en page 63.)

[Liu & Wong, 2009] Liu, W. & Wong, W. (2009). Web service clustering using text mining techniques. *International Journal of Agent-Oriented Software Engineering*, 3(1), 6–26. (Cité en page 52.)

[Liu et al., 2009] Liu, X., Huang, G., & Mei, H. (2009). Discovering homogeneous web service community in the user-centric web environment. *IEEE Trans. Services Computing*, 2(2), 167–181. (Cité en page 63.)

[Lopez-Velasco et al., 2006] Lopez-Velasco, C., Ramos, A. C., Villanova-Oliver, M., Gensel, J., & Martin, H. (2006). Sélection de services web adaptés au contexte d'utilisation. In *INFORSID* (pp. 199–214). (Cité en page 60.)

[Ma et al., 2007] Ma, J., Cao, J., & Zhang, Y. (2007). A probabilistic semantic approach for discovering web services. In C. L. Williamson, M. E. Zurko, P. F. Patel-Schneider, & P. J. Shenoy (Eds.), *WWW* (pp. 1221–1222). : ACM. (Cité en page 52.)

[Ma et al., 2008] Ma, J., Zhang, Y., & He, J. (2008). Efficiently finding web services using a clustering semantic approach. In Q. Z. Sheng, U. Nambiar, A. P. Sheth, B. Srivastava, Z. Maamar, & S. Elnaffar (Eds.), *CSSSIA*, volume 292 of *ACM International Conference Proceeding Series* (pp.5̃). : ACM. (Cité en pages 53 et 155.)

[Maamar et al., 2009] Maamar, Z., Subramanian, S., Thiran, P., Benslimane, D., & Bentahar, J. (2009). An approach to engineer communities of web services : Concepts, architecture, operation, and deployment. *IJEBR*, 5(4), 1–21. (Cité en pages 5, 62 et 150.)

[Maccatrozzo et al., 2014] Maccatrozzo, V., Ceolin, D., Aroyo, L., & Groth, P. (2014). *A Semantic Pattern-Based Recommender*, (pp. 182–187). Springer International Publishing : Cham. (Cité en page 56.)

[Mancoridis et al., 1998] Mancoridis, S., Mitchell, B., Rorres, C., Chen, Y., & Gansner, E. (1998). Using automatic clustering to produce high-level system organizations of source code. In *Proc. 6th Intl. Workshop on Program Comprehension* (pp. 45–53). (Cité en page 151.)

[Mandhani et al., 2003] Mandhani, B., Joshi, S., & Kummamuru, K. (2003). A matrix density based algorithm to hierarchically co-cluster documents and words. In G. Hencsey, B. White, Y.-F. R. Chen, L. Kovács, & S. Lawrence (Eds.), *WWW* (pp. 511–518). : ACM. (Cité en page 155.)

[Mannila & Toivonen, 1997] Mannila, H. & Toivonen, H. (1997). Levelwise Search and Borders of Theories in Knowledge Discovery. *Data Mining and Knowledge Discovery*, 1(3), 241–258. (Cité en pages 39, 41, 64 et 130.)

[Manning et al., 2008] Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press. (Cité en page 116.)

[Martin et al., 2004a] Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., & Sycara, K. (2004a). Owl-s : Semantic markup for web services. Internet [http ://www.w3.org/Submission/2004/SUBM- OWL-S-20041122/]. (Cité en page 49.)

[Martin et al., 2004b] Martin, D. et al. (2004b). Bringing semantics to web services : The OWL-S approach. In J. Cardoso & A. P. Sheth (Eds.), *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*, volume 3387 of *LNCS* Heidelberg, DE : Springer. (Cité en page 60.)

[Medjahed & Bouguettaya, 2005] Medjahed, B. & Bouguettaya, A. (2005). A dynamic foundational architecture for semantic web services. *Distributed and Parallel Databases*, 17(2), 179–206. (Cité en page 62.)

[Mehta et al., 2004] Mehta, B., NiederÈe, C., Stewart, A., Muscogiuri, C., & Neuhold, E. J. (2004). An Architecture for Recommendation Based Service Mediation. In *ICSNW*, volume 3226 of *LNCS* (pp. 250–262). : Springer. (Cité en page 56.)

[Mei et al., 2010] Mei, Q., Guo, J., & Radev, D. R. (2010). Divrank : the interplay of prestige and diversity in information networks. In B. Rao, B. Krishnapuram, A. Tomkins, & Q. Yang (Eds.), *KDD* (pp. 1009–1018). : ACM. (Cité en page 59.)

[Messai, 2009] Messai, N. (2009). *Analyse de concepts formels guidée par des connaissances de domaine : Application à la découverte de ressources génomiques sur le Web*. PhD thesis, Thèse de doctorat, École doctorale IAEM Lorraine. (Cité en page 35.)

[Mier et al., 2015] Mier, P. R., Pedrinaci, C., Lama, M., & Mucientes, M. (2015). An integrated semantic web service discovery and composition framework. *IEEE Transactions on Services Computing*, PP(99), 1–1. (Cité en page 110.)

[Mitchell, 1982] Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence*, 18(2), 203–226. (Cité en page 40.)

[Naim et al., 2016a] Naim, H., Aznag, M., Quafafou, M., & Durand, N. (2016a). Probabilistic approach for diversifying web services discovery and composition. In S. Reiff-Marganiec (Ed.), *ICWS* (pp. 73–80). : IEEE Computer Society. (Cité en pages 6, 7 et 70.)

[Naim et al., 2016b] Naim, H., Aznag, M., Quafafou, M., & Durand, N. (2016b). Semantic divergence based evaluation of web service communities. In J. Zhang, J. A. Miller, & X. Xu (Eds.), *SCC* (pp. 736–743). : IEEE Computer Society. (Cité en pages 6, 9 et 150.)

[Naïm et al., 2016] Naïm, H., Aznag, M., Durand, N., & Quafafou, M. (2016). Semantic pattern mining based web service recommendation. In Q. Z. Sheng, E. Stroulia, S. Tata, & S. Bhiri (Eds.), *ICSOC*, volume 9936 of *Lecture Notes in Computer Science* (pp. 417–432). : Springer. (Cité en pages 6, 8 et 57.)

[Newman, 2004a] Newman, M. E. (2004a). Fast algorithm for detecting community structure in networks. *Physical review E*, 69(6), 066133. (Cité en page 153.)

[Newman, 2004b] Newman, M. E. J. (2004b). Detecting community structure in networks. *European Physical Journal*, B 38, 321–330. (Cité en page 62.)

[Newman & Girvan, 2004] Newman, M. E. J. & Girvan, M. (2004). Finding and evaluating community structure in networks. (Cité en page 153.)

[Nourine & Raynaud, 1999] Nourine, L. & Raynaud, O. (1999). A fast algorithm for building lattices. *Inf. Process. Lett.*, 71(5-6), 199–204. (Cité en page 66.)

[OASIS UDDI Spec, 2004] OASIS UDDI Spec, T. C. (October 2004). *UDDI Version 3.0.2. http ://uddi.org/pubs/uddi-v3.0.2-20041019.htm.* Technical report, OASIS. (Cité en page 21.)

[Oh & Lee, 2009] Oh, S.-C. & Lee, D. (2009). Wsben : A web services discovery and composition benchmark toolkit1. *Int. J. Web Service Res.*, 6(1), 1–19. (Cité en pages 59 et 61.)

[Oh et al., 2008] Oh, S.-C., Lee, D., & Kumara, S. R. T. (2008). Effective web service composition in diverse and large-scale service networks. *IEEE Trans. Services Computing*, 1(1), 15–32. (Cité en pages 58, 59, 61 et 110.)

[Orman et al., 2012] Orman, G. K., Labatut, V., & Cherifi, H. (2012). Comparative evaluation of community detection algorithms : A topological approach. *CoRR*, abs/1206.4987. (Cité en page 170.)

[Palla et al., 2007] Palla, G., Barabasi, A.-l., Vicsek, T., & Hungary, B. (2007). Quantifying social group evolution. 446, 2007. (Cité en page 62.)

[Parsa & Fakhr, 2009] Parsa, S. & Fakhr, K. (2009). Using agent-oriented reasoning engine and sdc graph for optimizing semantic web services discovery. In *Computer Conference, 2009. CSICC 2009. 14th International CSI* (pp. 423–430). (Cité en page 52.)

[Pasquier et al., 1999] Pasquier, N., Bastide, Y., Taouil, R., & Lakhal, L. (1999). Efficient Mining of Association Rules Using Closed Itemset Lattices. *Information Systems*, 24(1), 25–46. (Cité en pages 42, 43, 64, 65 et 131.)

[Pazzani & Billsus, 2007] Pazzani, M. J. & Billsus, D. (2007). Content-based recommendation systems. In *THE ADAPTIVE WEB : METHODS AND STRATEGIES OF WEB PERSONALIZATION. VOLUME 4321 OF LECTURE NOTES IN COMPUTER SCIENCE* (pp. 325–341). : Springer-Verlag. (Cité en page 56.)

[Pedrinaci & Leidig, 2011] Pedrinaci, C. & Leidig, T. (2011). *Linked USDL Core. http ://www.linked-usdl.org/ns/usdl-core*. Technical report, . (Cité en page 50.)

[Peltz, 2003] Peltz, C. (2003). Web services orchestration and choreography. *Computer*, 36(10), 46–52. (Cité en page 61.)

[Pons & Latapy, 2005] Pons, P. & Latapy, M. (2005). Computing communities in large networks using random walks (long version). arXiv :arXiv :physics/0512106v1. (Cité en page 154.)

[Porter, 1980] Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3), 130–137. (Cité en page 25.)

[Qinjiao Mao et al., 2013] Qinjiao Mao, Q., Feng, B., & Pan, S. (2013). Modeling User Interests Using Topic Model. *JATIT*, 48(1), 600–606. (Cité en page 56.)

[Raghavan et al., 2007] Raghavan, U. N., Albert, R., & Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. (Cité en pages 154 et 155.)

[Reihanian et al., 2015] Reihanian, A., Minaei-Bidgoli, B., & Alizadeh, H. (2015). Topic-oriented community detection of rating-based social networks. *Journal of King Saud University - Computer and Information Sciences*, (pp.˜-). (Cité en pages 150 et 151.)

[Rijsbergen, 1979] Rijsbergen, C. V. (1979). *Information Retrieval*. Butterworth-Heinemann Newton, MA, USA. (Cité en page 116.)

[Salton, 1989] Salton, G. (1989). *Automatic Text Processing – The Transformation, Analysis, and Retrieval of Information by Computer*. Addison–Wesley. (Cité en page 26.)

[Schaeffer, 2007] Schaeffer, S. (2007). Graph clustering. *Computer Science Review*, 1(1), 27–64. (Cité en page 62.)

[Schuetz & Caflisch, 2008] Schuetz, P. & Caflisch, A. (2008). Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement. *Phys. Rev. E*, 77, 046112. (Cité en page 153.)

[Schulte et al., 2010] Schulte, S., Lampe, U., Eckert, J., & Steinmetz, R. (2010). Log4sws.kom : Self-adapting semantic web service discovery for sawsdl. In *SERVICES* (pp. 511–518). : IEEE Computer Society. (Cité en page 52.)

[Segev & Sheng, 2012] Segev, A. & Sheng, Q. Z. (2012). Bootstrapping ontologies for web services. *IEEE Trans. Services Computing*, 5(1), 33–44. (Cité en page 56.)

[Sheth et al., 2007] Sheth, A. P., Gomadam, K., & Lathem, J. (2007). Sa-rest : Semantically interoperable and easier-to-use services and mashups. *IEEE Internet Computing*, 11(6), 91–94. (Cité en pages 50 et 56.)

[Soulet, 2006] Soulet, A. (2006). *Un cadre générique de découverte de motifs sous contraintes fondées sur des primitives*. PhD thesis, Thèse de doctorat, Université de Caen Basse-Normandie. (Cité en page 42.)

[Steyvers & Griffiths, 2007] Steyvers, M. & Griffiths, T. (2007). Probabilistic topic models. In T. Landauer, S. D. McNamara, & W. Kintsch (Eds.), *Latent Semantic Analysis : A Road to Meaning* chapter Probabilistic topic models. Laurence Erlbaum. (Cité en pages 9, 27, 28, 30, 64, 87, 150 et 155.)

[Suguna & Sharmila, 2013] Suguna, R. & Sharmila, D. (2013). Article : An efficient web recommendation system using collaborative filtering and pattern discovery algorithms. *International Journal of Computer Applications*, 70(3), 37–44. Full text available. (Non cité.) :article :article :article :article

[Toma et al., 2005] Toma, I., Iqbal, K., Moran, M., Roman, D., Strang, T., & Fensel, D. (2005). An evaluation of discovery approaches in grid and web services environments. In R. Hirschfeld, R. Kowalczyk, A. Polze, & M. Weske (Eds.), *NODe/GSEM*, volume 69 of *LNI* (pp. 233–247). : GI. (Cité en page 3.)

[Tong et al., 2011] Tong, H., He, J., Wen, Z., Konuru, R., & Lin, C.-Y. (2011). Diversified ranking on large graphs : an optimization viewpoint. In C. Apté, J. Ghosh, & P. Smyth (Eds.), *KDD* (pp. 1028–1036). : ACM. (Cité en page 59.)

[Tukey, 1977] Tukey, J. W. (1977). *Exploratory Data Analysis*. Massachusetts : Addison Wesley, 1 edition edition. (Cité en page 168.)

[van der Merwe et al., 2004] van der Merwe, D., Obiedkov, S. A., & Kourie, D. G. (2004). Addintent : A new incremental algorithm for constructing concept lattices. In P. W. Eklund (Ed.), *ICFCA*, volume 2961 of *Lecture Notes in Computer Science* (pp. 372–385). : Springer. (Cité en page 34.)

[Verma & Bharadwaj, 2015] Verma, A. & Bharadwaj, K. K. (2015). Discovering communities in heterogeneous social networks based on non-negative tensor factorization and cluster ensemble approach. In R. Prasath, A. K. Vuppala, & T. Kathirvalavakumar (Eds.), *MIKE*, volume 9468 of *Lecture Notes in Computer Science* (pp. 150–160). : Springer. (Cité en page 63.)

[Wang et al., 2015] Wang, J., Gao, P., Ma, Y., & He, K. (2015). *Common Topic Group Mining for Web Service Discovery*, (pp. 92–107). Springer International Publishing : Cham. (Cité en page 53.)

[Werthner et al., 2007] Werthner, H., Hansen, H. R., & Ricci, F. (2007). Recommender systems. In *HICSS* (pp. 167). : IEEE Computer Society. (Cité en pages 4 et 55.)

[Wille, 1982] Wille, R. (1982). Restructuring lattice theory : an approach based on hierarchies of concepts. In I. Rival (Ed.), *Ordered sets* (pp. 445–470). : Reidel. (Cité en pages 31 et 44.)

[Wu & Khoury, 2012] Wu, C.-S. & Khoury, I. (2012). Tree-based search algorithm for web service composition in saas. In S. Latifi (Ed.), *ITNG* (pp. 132–138). : IEEE Computer Society. (Cité en pages 6, 61 et 62.)

[Wu et al., 2014] Wu, J., Chen, L., Zheng, Z., Lyu, M. R., & Wu, Z. (2014). Clustering web services to facilitate service discovery. *Knowledge and Information Systems*, 38(1), 207–229. (Cité en pages 52 et 53.)

[Xie et al., 2011] Xie, J., Szymanski, B. K., & Liu, X. (2011). Slpa : Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In M. Spiliopoulou, H. Wang, D. J. Cook, J. Pei, W. Wang, O. R. Zaïane, & X. Wu (Eds.), *ICDM Workshops* (pp. 344–349). : IEEE Computer Society. (Cité en page 155.)

[Xu et al., 2008] Xu, G., Zhang, Y., & Yi, X. (2008). Modelling user behaviour for web recommendation using lda model. In *Web Intelligence/IAT Workshops* (pp. 529–532). : IEEE Computer Society. 978-0-7695-3496-1. (Cité en page 56.)

[Xu et al., 2007] Xu, Z., Martin, P., Powley, W., & Zulkernine, F. (2007). Reputation-enhanced qos-based web services discovery. *IEEE International Conference on Web Services (ICWS 2007)*. (Cité en pages 23 et 136.)

[Yao et al., 2015] Yao, L., Sheng, Q. Z., Ngu, A. H. H., Yu, J., & Segev, A. (2015). Unified collaborative and content-based web service recommendation. *IEEE Trans. Services Computing*, 8(3), 453–466. (Cité en page 57.)

[Yao et al., 2013] Yao, L., Sheng, Q. Z., Segev, A., & Yu, J. (2013). Recommending web services via combining collaborative filtering with content-based features. In *ICWS* (pp. 42–49). : IEEE Computer Society. (Cité en page 57.)

[Yu, 2011] Yu, Q. (2011). Place semantics into context : Service community discovery from the wsdl corpus. In G. Kappel, Z. Maamar, & H. R. M. Nezhad (Eds.), *ICSOC*, volume 7084 of *Lecture Notes in Computer Science* (pp. 188–203). : Springer. (Cité en page 63.)

[Yu & Rege, 2010] Yu, Q. & Rege, M. (2010). On service community learning : A co-clustering approach. In *ICWS* (pp. 283–290). : IEEE Computer Society. (Cité en page 63.)

[Zaki & Hsiao, 2005] Zaki, M. & Hsiao, C.-J. (2005). Efficient algorithms for mining closed itemsets and their lattice structure. *TKDE*, 17(4), 462–478. (Cité en pages 8, 66, 125 et 140.)

[Zaki & Hsiao, 2002] Zaki, M. J. & Hsiao, C.-J. (2002). CHARM : An Efficient Algorithm for Closed Itemset Mining. In *Proceedings of the 2nd SIAM International Conference on Data Mining (SDM'02)* Arlington. (Cité en page 65.)

[Zaki et al., 1997] Zaki, M. J., Parthasarathy, S., Ogihara, M., & Li, W. (1997). New algorithms for fast discovery of association rules. In *Proceedings of the 3rd international conference on Knowledge Discovery and Data mining (KDD'97)* (pp. 283–286). : AAAI Press. (Cité en page 64.)

[Zhang et al., 2016] Zhang, N., Wang, J., He, K., & Li, Z. (2016). An approach of service discovery based on service goal clustering. In *2016 IEEE International Conference on Services Computing (SCC)* (pp. 114–121). (Cité en page 52.)

[Zhao & Karypis, 2001] Zhao, Y. & Karypis, G. (2001). Criterion functions for document clustering : Experiments and analysis. (Cité en pages 9, 63, 150 et 155.)

[Zheng et al., 2011a] Zheng, H., Yang, J., Zhao, W., & Bouguettaya, A. (2011a). Qos analysis for web service compositions based on probabilistic qos. In G. Kappel, Z. Maamar, & H. R. M. Nezhad (Eds.), *ICSOC*, volume 7084 of *Lecture Notes in Computer Science* (pp. 47–61). : Springer. (Cité en page 56.)

[Zheng et al., 2009] Zheng, Z., Ma, H., Lyu, M. R., & King, I. (2009). WSRec : A Collaborative Filtering Based Web Service Recommender System. In *ICWS* (pp. 437–444). (Cité en page 55.)

[Zheng et al., 2011b] Zheng, Z., Ma, H., Lyu, M. R., & King, I. (2011b). Qos-aware web service recommendation by collaborative filtering. *IEEE Transactions on Service Computing (TSC)*, 4(2), 140–152. (Cité en pages 23, 55 et 56.)