République Algérienne Démocratique et Populaire
Ministère de l'Enseignement supérieur et de la Recherche Scientifique

UNIVERSITÉ ABOU-BEKR BELKAID - TLEMCEN

THÈSE

Présentée à :

FACULTÉ DES SCIENCES – DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

DOCTORAT EN SCIENCES

Spécialité : Informatique

Par :

M. SMAHI Mohammed Ismail

Sur le thème

Estimation de la QoS dans les services Web par apprentissage profond

Soutenue publiquement le 12 Mars 2022 à Tlemcen devant le jury composé de MM. :

CHIKH Azeddine	Professeur	Université de Tlemcen	Président
BENMAMMAR Badr	Professeur	Université de Tlemcen	Examinateur
SOUIER Mehdi	Professeur	ESM de Tlemcen	Examinateur
DENNOUNI Nassim	MCA	Université de Chlef	Examinateur
BENAMAR Abdelkrim	Professeur	Université de Tlemcen	Directeur de thèse
TIBERMACINE Chouki	Professeur	Université de Montpellier	Co-Directeur de thèse
HADJILLA Fethallah	MCA	Université de Tlemcen	Invité

Laboratoire de Recherche en Informatique de Tlemcen (LRIT)
BP 119, 13000 Tlemcen - Algérie

Année universitaire 2021-2022

Remerciements

En témoignage de mes profonds sentiments de respect et de remerciements, je tiens à exprimer ma sincère gratitude à Messieurs BENAMAR Abdelkrim, TIBERMACINE Chouki et HADJILLA Fethallah, mes encadrants, pour leur disponibilité, leurs précieux et judicieux conseils, leurs encouragements et leurs sympathies qui m'ont permis de mener à bien cette thèse.

J'adresse mes sincères remerciements à Monsieur Chikh Azeddine, Professeur à l'université de Tlemcen, qui m'a fait l'honneur de présider le jury de cette thèse.

J'exprime ma profonde reconnaissance à Monsieur BENMAMMAR Badr, Professeur à l'université de Tlemcen, Monsieur SOUIER Mehdi, Professeur à l'École Supérieure de management de Tlemcen, et Monsieur DENNOUNI Nassim, Maitre de conférences à l'Université de Chlef, pour l'intérêt qu'ils ont bien voulu porter à ce travail en acceptant de l'examiner et d'en être rapporteurs.

Mes remerciements vont à toutes les personnes que j'ai rencontrées au cours de mon séjour au laboratoire LIRMM à Montpellier, pour leur convivialité et pour toutes les riches discussions qu'elles ont menées.

Je ne saurais oublier d'adresser un grand merci à tous mes amis et collègues enseignants du département d'informatique. Je leur exprime ma profonde sympathie et je leur souhaite une bonne continuation.

Je dédie cet humble travail

À mes chers parents
À ma chère femme et mes filles : Alae et Kawthar
À ma sœur, mes frères et leurs familles
À toute ma famille et amis

Résumé

Dans cette thèse, nous proposons une approche de prédiction de QoS fondée sur le modèle d'apprentissage profond combinée avec de la factorisation matricielle, où cette dernière est basée, à la fois, sur une architecture des auto-encodeurs et une technique de classification. Trois variants du modèle d'auto-encodeur ont été utilisé. Le premier est composé d'une seule couche cachée représentant le vecteur des facteurs latents des utilisateurs et/ou services. Une deuxième architecture considère plusieurs couches cachées. Un troisième modèle consiste en une combinaison entre l'architecture d'auto-encodeur profond avec celle d'un réseau antagoniste génératif. D'autres problèmes sous-jacents à l'estimation des valeurs manquantes de QoS ont été traités dans ce travail. Le premier est relatif à la vulnérabilité des systèmes de prédiction face au problème de la rareté de données. Notre proposition pour surmonter ce dernier consiste en une utilisation d'un algorithme de partitionnement basé sur les cartes auto-organisatrices de Kohonen, où l'initialisation est faite à la base des attributs de localisation. Le second problème traité est le démarrage à froid, survenu lors de l'ajout de nouveaux utilisateurs/services au système de prédiction. Ce dernier est globalement maîtrisé en exploitant également les caractéristiques spatiales. Les différentes évaluations empiriques que nous avons menées, montrent que nos propositions de solutions sont en mesure de donner de meilleures performances en matière de prédiction des valeurs de QoS manquantes, et fournissent ainsi de meilleures orientations aux utilisateurs dans leurs choix de services préférés que les méthodes existantes. Les paramètres, en termes de QoS, sur lesquels nous nous sommes appuyés pour réaliser ces différentes expériences sont le temps de réponse et le débit. Cependant, les algorithmes de prédiction de QoS, proposés dans cette thèse, peuvent être appliqués aisément à d'autres facteurs de QoS.

Mots-clés : Prédiction de QoS, Service Web, Filtrage collaboratif, Apprentissage profond, Auto-encodeur, Réseau antagoniste génératif, Carte auto-organisatrice.

Abstract

In this thesis, we propose a deep learning-based approach wich combines a matrix factorization model based on a deep auto-encoder (DAE) and a clustering technique. Three variants of the auto-encoder design have been used. The first one is composed of a single hidden layer that represents the vector of latent factors of users and/or services. A second architecture considers several hidden layers. A third model consists of a combination of a deep auto-encoder model and a generative adversarial network. Other problems underlying the estimation of missing QoS values were addressed in this work. The first one is related to the vulnerability of prediction systems to the data sparsity problem. To deal with this issue our proposal consists of in using a clustering algorithm based on Kohonen's self-organising maps, where the initialization is done using location attributes. The second one that we have dealt with is the cold start problem, which occurs when adding new users/services to the prediction system. The latter one is globally managed by exploiting a spatial features as well. The conducted experiments show that our proposals can provide better performances in terms of QoS prediction, and consequently provide more guidance for users in their choice of preferred services than existing methods do. The QoS parameters on which we relied on to carry out our various experiments are response time and throughput. However, the proposed QoS prediction algorithms can be applied to other QoS factors.

Keywords : QoS Prediction, Web service, Collaborative Filtering, Deep learning, Autoencoder, Generative Adversarial Network, Self-Organizing Map.

ملخص

في هذه الأطروحة, نقترح نهجًا قائمًا على التعلم العميق يجمع بين نموذج عامل المصفوفة استنادًا إلى وحدة التشفير التلقائي العميقة وتقنية التجميع. من أجل ذلك تم الإستناد على ثلاثة تصاميم مختلفة من شبكات التشفير التلقائي. يتكون النموذج الأول من طبقة مخفية واحدة من العصبونات التي ترمز لمجموعة العوامل الكامنة الممثلة للمستخدمين و / أو الخدمات. بينما يتكون النموذج الثاني من عدة طبقات مخفية. النموذج الثالث هو عبارة عن دمج بين نموذج التشفير التلقائي العميق من جهة وشبكة خصومية توليدية من جهة أخرى. في هذا السياق تمت معالجة عدة صُعُوبات أخرى متعلقة بمشكلة التنبؤ بالقيم المفقودة لجودة الخدمة. الصعوبة الأولى تتعلق بقابلية أنظمة التنبؤ للتأثر بمشكلة قلة البيانات. و للتعامل مع هذه المشكلة, يتألف اقتراحنا في استخدام خوارزمية التجميع إعتمادا على خريطة شبكات كوهونين ذاتية التنظيم, حيث تتم تهيئتها باستخدام مجموعة من الميزات المكانية. الصعوبة الثانية التي تعاملنا معها هي مشكلة الإنطلاقة الفاترة, والتي تحدث عند إضافة مستخدمين و / أو خدمات جديدة إلى نظام التنبؤ. تتم إدارة هذه المشكلة بصفة عامة من خلال استغلال الميزات المكانية كذلك. تُظهر التجارب التي تم إجراؤها أن مقترحاتنا بأسنطاعتها تقديم أداء أفضل من حيث التنبؤ بجودة الخدمة, وبالتالي توفير المزيد من الإرشادات للمستخدمين في اختيارهم للخدمات المفضلة مقارنة بالطرق الحالية. يجدر الإشارة على أن الخصائص المُعَرِّفة لجودة الخدمة التي اعتمدنها من إجل إجراء تجاربنا المختلفة هي زمن الاستجابة وسرعة النقل. ومع ذلك, يمكن تطبيق خوارزميات توقع جودة الخدمة المقترحة على العوامل الأخرى لجودة الخدمة.

الكلمات الدالة: التنبؤ بجودة الخدمة, خدمة ويب, التصفية التعاونية, التعلم العميق, التشفير التلقائي, شبكة خصومية توليدية, خريطة الخصائص ذاتية التنظيم.

Table des matières

Remerciements **i**

Résumé **ii**

Abstract **iii**

ملخص **iv**

Table des matières **vi**

Table des figures **ix**

Liste des tableaux **xi**

Liste des Algorithmes **xi**

Liste des abréviations et des sigles **xiii**

1 Introduction générale **1**
1.1 Contexte et motivations 1
1.2 Objectifs et problématiques 2
1.3 Contributions 4
1.4 Organisation du manuscrit 5

I Prérequis pour l'estimation de la QoS pour les services Web **7**

2 Services Web et attributs de QoS **8**
2.1 Introduction 8
2.2 Paradigme de l'approche orientée services 9
2.2.1 Définition et historique 9
2.2.2 Principes de SOA 9
2.3 Les bases des services Web 10
2.3.1 Notions de service 10
2.3.2 Notions de service Web 10
2.3.3 Modèle de base des service Web 11
2.3.4 Architecture en couches des services Web 12
2.4 La description des services Web 13
2.4.1 Description SOAP 13
2.4.2 Description REST 14
2.4.3 REST vs SOAP 17

2.5 Attributs de qualité pour les services Web . . . 18
2.5.1 Définition . . . 18
2.5.2 Paramètres de QoS . . . 19
2.5.3 Classification des attributs de QoS . . . 20
2.6 Conclusion . . . 22

3 Le filtrage collaboratif noyau de la prédiction 23
3.1 Introduction . . . 23
3.2 Les systèmes de recommandation . . . 24
3.2.1 Origines des systèmes de recommandation . . . 24
3.2.2 Définition de base . . . 25
3.2.3 Notations . . . 25
3.3 Filtrage collaboratif . . . 26
3.3.1 Classification des approches basées sur le FC . . . 26
3.3.2 Filtrage collaboratif basé sur la mémoire . . . 27
3.3.3 Filtrage collaboratif basé sur un modèle . . . 35
3.3.4 Les modèles à facteurs latents . . . 35
3.3.5 Les modèles probabilistes . . . 37
3.4 Prédiction versus Recommandation . . . 40
3.5 évaluation des techniques du FC . . . 41
3.6 Conclusion . . . 42

4 L'apprentissage profond 43
4.1 Introduction . . . 43
4.2 Apprentissage automatique . . . 44
4.2.1 Origine et inspiration . . . 45
4.2.2 Perceptron monocouche . . . 46
4.2.3 Réseaux de neurones artificiels . . . 48
4.3 Apprentissage profond . . . 54
4.3.1 Réseaux de neurones convolutifs . . . 55
4.3.2 Réseaux de neurones récurrents . . . 56
4.4 Les modèles génératifs d'apprentissage profond . . . 57
4.4.1 Auto-encodeurs . . . 57
4.4.2 Les réseaux antagonistes génératifs . . . 61
4.4.3 Principes et définition . . . 61
4.4.4 L'apprentissage dans les GANs . . . 62
4.5 Réseaux auto-organisateurs . . . 63
4.5.1 Principe de fonctionnement . . . 63
4.5.2 L'apprentissage dans les réseaux de Kohonen . . . 63
4.6 Conclusion . . . 64

II Estimation de la QoS pour les services Web 65

5 Prédiction de la QoS par FC : état de l'art (Partie 1) 66
5.1 Introduction . . . 66
5.2 Stratégie de recherche documentaire adoptée . . . 67
5.2.1 Processus de recherche bibliographique . . . 67
5.2.2 Sélection d'articles . . . 68
5.3 La recommandation des services Web . . . 69
5.3.1 La prédiction de la QoS . . . 69
5.3.2 Classification des différentes méthodes . . . 70
5.4 Prédiction en séries chronologiques . . . 72

5.4.1 Approches d'analyse statistiques . . . 73
5.4.2 Approches à base de réseaux de neurones récurrents . . . 75
5.4.3 Synthèse des approches sensibles au temps . . . 77
5.5 Prédiction basée sur la mémoire . . . 79
5.5.1 Voisinage utilisateur . . . 79
5.5.2 Voisinage service . . . 81
5.5.3 Méthodes hybrides . . . 82
5.5.4 Synthèse des approches basées voisinage . . . 83
5.6 Conclusion . . . 84

6 Prédiction de QoS par apprentissage profond 87
6.1 Introduction . . . 87
6.2 Prédiction de la QoS par FC : état de l'art (Partie 2) . . . 88
6.2.1 Prédiction basée sur les modèles . . . 88
6.2.2 Prédiction basée sur les modèles hybrides . . . 95
6.2.3 Avantages et limites des approches de FC . . . 100
6.3 Estimation de la QoS par apprentissage profond . . . 101
6.3.1 Les motivations sur nos choix de conception . . . 102
6.3.2 Architecture du modèle . . . 103
6.3.3 Partitionnement de données . . . 105
6.3.4 Prédiction de la QoS . . . 107
6.4 Problème du démarrage à froid . . . 110
6.5 Conclusion . . . 112

7 Expérimentations et évaluation des performances 114
7.1 Introduction . . . 114
7.2 Jeux de données et métriques d'évaluation . . . 115
7.2.1 Bases de test existantes . . . 115
7.2.2 Base de test utilisée . . . 116
7.2.3 Métriques d'évaluation utilisées . . . 117
7.3 Comparaison des performances . . . 118
7.3.1 Impact des paramètres d'apprentissage . . . 118
7.3.2 Impact des informations de localisation . . . 119
7.3.3 Choix de modèles . . . 122
7.3.4 Résultats des expérimentations & comparaison des performances . . . 125
7.4 Problème du démarrage à froid . . . 128
7.5 Menaces à la validité (Threats To Validity) . . . 129
7.5.1 Validité interne . . . 129
7.5.2 Validité externe . . . 129
7.6 Conclusion . . . 130

8 Conclusion générale 131
8.1 Synthèse . . . 131
8.2 Perspectives . . . 133

Liste des publications 134

Bibliographie 135

Table des figures

1.1 Classification globale des systèmes de recommandation 2

2.1 Les quatre concepts de base de SOA 9
2.2 Modèle d'interaction pour les services Web 11
2.3 La pile des technologies pour les services Web 12
2.4 Composition du message SOAP 14
2.5 Exemple de fichier WSDL 14
2.6 L'invocation de service dans REST 15
2.7 Évolution de l'intérêt des termes RESTful API and SOAP API 17
2.8 La pile des couches des Services Web vs QoS 20
2.9 La classification des attributs de QoS 21

3.1 Classification globale des systèmes de recommandation 24
3.2 Classification globale des techniques de filtrage collaboratif 26
3.3 Classification des techniques de filtrage collaboratif basées sur la mémoire 27
3.4 Filtrage collaboratif basé sur les utilisateurs 29
3.5 Filtrage collaboratif basé sur les items 32
3.6 Exemple illustratif 34
3.7 Classification globale des techniques de filtrage collaboratif basées sur les modèles . . . 36
3.8 Factorisation matricielle 37
3.9 Méthodes à facteurs latents 37
3.10 Prédiction vs Recommandation 40

4.1 Vue globale de l'intelligence artificielle 44
4.2 Neurone biologique 46
4.3 Neurone artificiel 46
4.4 Séparation binaire des instances 47
4.5 Exemples de fonctions d'activation 49
4.6 Exemple d'architecture pour un perceptron multicouches 51
4.7 Schéma conceptuelle du réseau de neurones convolutif 55
4.8 Structure en boucle dans le RNN 56
4.9 Architecture de base d'un autoencodeur 59
4.10 Auto-encodeur de débruitage 60
4.11 Structure générale contradictoire génératif 62

5.1 Processus de sélection d'articles bibliographiques 68
5.2 Classification globale des approches de prédiction de la QoS 71
5.3 Approches de prédiction de la QoS basées sur le FC 72
5.4 Modélisation de la relation utilisateur, service et intervalles de temps 72
5.6 Architecture du modèle QF-RNN 76

5.7 Architecture du modèle PLMF 76
5.8 Architecture du modèle BRAFC 80
5.10 Classification des travaux basés sur la mémoire 83

6.1 Framework pour la prédiction de la QoS du modèle JCM 91
6.2 Vue statistique sur les travaux de l'état-de-l'art examinés. 100
6.3 Aperçu de l'architecture du système de prédiction 104
6.4 Modèles de réseaux de neurones proposés. 104
6.5 Exemple illustratif pour la phase de partitionnement de données 105
6.6 Prédiction de la QoS par l'auto-encodeur profond 107
6.7 Prédiction de la QoS par l'adversarial auto-encodeur 109
6.8 Processus à appliquer pour résoudre le problème du démarrage à froid 111

7.1 Influence du choix de la taille des couches 118
7.2 Influence du choix de la fonction d'activation 119
7.3 Performances des auto-encodeurs profonds sur les différentes partitions (l'initialisation de la carte auto-organisatrice est basée sur les attributs de localisation) 120
7.4 Performances des auto-encodeurs profonds sur les différentes partitions (l'initialisation de la carte auto-organisatrice est basée sur un processus aléatoire) 121
7.5 Performance de la prédiction pour l'auto-encodeur profond 123
7.6 Performances lors de la phase d'apprentissage de l'auto-encodeur profond 123
7.7 Moyenne des performance de la prédiction pour les auto-encodeurs profonds 124
7.8 Moyenne des performances lors de la phase d'apprentissage des auto-encodeurs profonds partitionnés 124
7.9 Performance de la prédiction pour l'adversarial auto-encodeur profond 125

Liste des tableaux

1.1 Proportions de pages et références du manuscrit 5

2.1 Liste des méthodes HTTP, fréquemment utilisée, dans REST 17
2.2 Essai de classification des attributs de QoS 21

5.1 Formalisation des requêtes de recherche 69
5.2 Synthèse d'état de l'art sur les méthodes sensibles au temps 78
5.3 Synthèse d'état de l'art sur le FC basé voisinage. 85

6.1 Synthèse d'état de l'art sur le FC basé modèles. 93
6.2 Synthèse d'état de l'art sur le FC hybride 97

7.1 Jeux données de QoS existants 115
7.2 Détails d'information sur le jeu de données WS-Dream 116
7.3 Les différentes améliorations enregistrées sur le jeu de données 117
7.4 Relation entre la taille des partitions et le % de parcimonie de données. 121
7.5 Comparaisons de performances de prédiction (sur le paramètre RT) 126
7.6 Taux d'amélioration de nos méthodes par rapport à la méthode LDCF 127
7.7 Problème du démarrage à froid 129

Liste des Algorithmes

1 Algorithme d'apprentissage du Perceptron 48
2 Algorithme de rétropropagation de gradient 53

3 Partitionnement de données avec SOM 106
4 L'apprentissage de l'auto-encodeur profond partitionné 108
5 L'apprentissage de l'auto-encodeur profond 108
6 L'apprentissage de Adversarial auto-encodeur 110
7 Traitement du problème du démarrage à froid 112

Liste des abréviations et des sigles

AI	Artificial intelligence
ANN	Artificial neural network
CNN	Convolutional Neural Networks
CPS	Cyber-Physical Systems
DL	Deep Learning
DNN	Deep Neural Networks
EC	Edge Computing
FC	Filtrage Collaboratif
FM	Factorisation Matricielle
GAN	Generative Adversarial Networks
HMF	Hierarchical Matrix Factorization
IoT	Internet of Things
IPCC	Item-based CF method using PCC
MAE	Mean Absolute Error
MF	Matrix Factorization
ML	Machine Learning
MLP	MultiLayer Perceptron
MTS	Multivariate Time Series
NMF	Non Negative Matrix Factorization
PCA	Principal Component Analysis
PCC	Pearson correlation coefficient
PCC	Person Correlation Coefficient
PMF	Probabilist Matrix Factorization
QoS	Quality of Service
RMSE	Root Mean Squared Error
RNN	Recurrent neural networks
RS	Système de Recommandation
SOA	Service Oriented Architecture

SOM Self Organizing Map

SVD Singular Value Decomposition

UIPCC integrate UPCC and IPCC

UPCC User-based CF method using PCC

URI Universal Resource Identifier

Chapitre 1

Introduction générale

Sommaire

1.1 Contexte et motivations 1
1.2 Objectifs et problématiques 2
1.3 Contributions 4
1.4 Organisation du manuscrit 5

1.1 Contexte et motivations

Aux cours de ces dernières décennies, l'informatique a fait des avancées considérables, notamment avec ces différentes propositions de solutions automatisées pour divers domaines (Bouguettaya et al., 2017). Au début, l'information était représentée par un format facilement interprétable par une machine, constitué de bits et d'octets, appelés données. Au fil du temps, il y a eu un vif intérêt sur l'ajout du sens aux données, les transformant ainsi en informations. Avec les progrès de l'informatique, le concept d'information est équipé avec une couche de raisonnement, donnant ainsi naissance à la notion de connaissance. Les progrès ne s'arrêtent pas là ; le besoin d'un niveau d'abstraction plus élevé a récemment conduit à l'idée d'ajouter des actions aux connaissances, ce qui donne naissance à la notion de service.

De l'autre côté, la généralisation du Web 2.0 [1], a transformé le Web en une sorte de « cerveau global », en se focalisant essentiellement sur l'aspect de l'intelligence collective, facilitant ainsi la collaboration, la créativité et le partage entre les différents utilisateurs. Les nouvelles applications telles que les réseaux sociaux, les wikis, les forums, les moteurs de recherche, représentent des exemples sur les nouveaux usages mis en place par cette version du Web. L'une des idées centrales derrière le paradigme du Web 2.0 est l'accès aux différentes données sur le Web à partir de diverses applications (Shum et al., 2008). Cette tendance correspond dans une large mesure aux normes qui organisent l'architecture orientée service (SOA).

Cette notion de *service* combinée avec la popularité du *Web 2.0*, ainsi que les applications pratiques de *l'architecture orientée service*, ont donné naissance au paradigme du *service Web*. En effet, les Services Web offrent un moyen de communication entre des utilisateurs et un ensemble de services disponibles sur le Web. Ces services ne sont rien d'autre que des composants logiciels réutilisables avec une mise en œuvre architecturale distribuée, et qui sont facilement accessibles via Internet (Bell, 2008). Les services Web sont considérés parmi les concrétisations les plus abouties d'une architecture orientée services. Celle-ci représente un style de conception de logiciels dans lequel les applications utilisent des services disponibles sur le réseau Internet (Erl, 2005). Bien que les prémisses de cette architecture aient été établies avant l'apparition des services Web, ces derniers sont devenus beaucoup plus accessibles via un modèle simple de programmation et de déploiement basé sur SOA. En effet,

1. Concept utilisé en première fois lors d'une conférence en octobre 2004 (Fayon, 2010)

les services Web sont construits à la base de protocoles bien connus et indépendants des plateformes. Ils reposent principalement sur des technologies basées sur le langage XML, tels que UDDI, WSDL et SOAP (Hashimi, 2003).

Grâce au fait qu'ils reposent sur cette architecture, les services Web présentent plusieurs caractéristiques : (1) un niveau élevé d'interopérabilité entre les différents utilisateurs d'applications à travers le Web, (2) un niveau de couplage faible, où un service dans ce modèle doit être le plus autonome que possible, (3) un niveau d'indépendance élevé par rapport aux plateformes ; et enfin, (4) un très grand niveau de capacité d'intégration et de réutilisation.

Cependant, les services Web se développent très rapidement et continuent de croître de jour en jour (Pandharbale et al., 2021) ; ce qui les rend difficilement accessibles pour les utilisateurs qui ont besoin de les découvrir et de les filtrer afin d'en sélectionner les services les plus appropriés par rapport à leurs centres d'intérêts. C'est face à ce constat que les systèmes de recommandation et de prédiction de services Web ont vu le jour.

Dans la littérature, plusieurs méthodes ont été développées et mises en place pour les systèmes de recommandation (Resnick and Varian, 1997). Selon (Bhatnagar, 2017), et en fonction des ressources et des données employées pour faire de la recommandation, ces approches peuvent être classées en différentes catégories, comme l'illustre la figure 1.1.

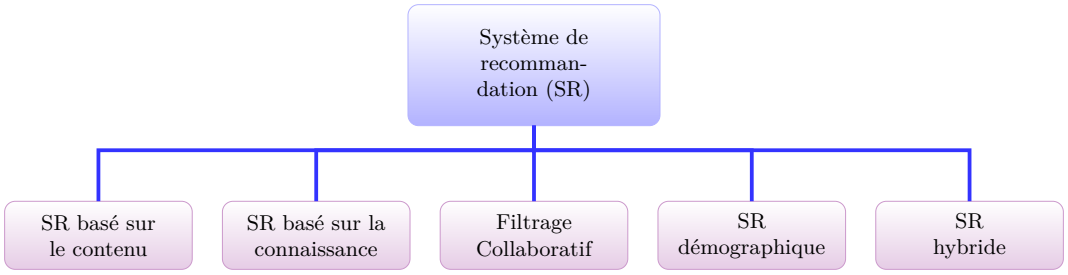


Figure 1.1 – Classification globale des systèmes de recommandation Bhatnagar (2017)

Le filtrage collaboratif (Goldberg et al., 1992) est considéré parmi les techniques, les plus émergentes, qui ont été utilisées dans les systèmes de recommandation. Le principe du concept s'est inspiré du monde réel. En effet, le filtrage collaboratif consiste en une collaboration entre plusieurs personnes afin qu'elles s'entraident pour effectuer des choix, tout en essayant de modéliser leurs réactions diverses.

La prolifération des services Web sur Internet a rendu la sélection d'un service parmi plusieurs services, fonctionnellement équivalents, une tâche fastidieuse et très compliquée. Plusieurs travaux dans la littérature (Wang et al., 2016 ; Yu et al., 2007 ; Zheng et al., 2014) ont été proposés pour discriminer des services équivalents, en s'appuyant sur des métadonnées comme la qualité de service (en anglais QoS, pour Quality of Service). En effet, la qualité de service désigne les mesures objectives effectuées lors de l'exécution du service pour donner suite à de nombreuses requêtes provenant d'applications clientes. Ces mesures concernent plusieurs attributs, comme la performance (temps d'exécution), la fiabilité (taux des requêtes traitées avec succès), etc.

Le contexte de cette thèse est celui de la prédiction des valeurs de ces attributs de QoS pour les services Web, basée sur une approche à filtrage collaboratif, lors du développement d'applications distribuées. L'estimation de ces paramètres s'effectue, contrairement aux travaux existants (Karim et al., 2015 ; Shao et al., 2007 ; Zhang et al., 2011a), via l'exploitation des avancées techniques de l'apprentissage automatique et/ou profond (Qu et al., 2018), tout en s'appuyant sur des ensembles de données (data sets) existants afin de valider nos différentes propositions.

1.2 Objectifs et problématiques

Le nombre de services Web est dans une croissance explosive ; ce qui les rendent difficilement accessibles pour les utilisateurs qui ont besoin de les chercher puis de les filtrer afin de les utiliser par la suite. A ce problème d'inaccessibilité, s'ajoute la problématique des fonctionnalités équivalentes des services Web. Actuellement, et avec la grande augmentation de la quantité de données disponibles sur

Internet, plusieurs fournisseurs de services offrent des services Web à fonctionnalités quasi similaires. Dès lors, le processus de sélection du service Web le plus approprié parmi ce grand nombre de services candidats équivalents, devient une tâche sensible et épineuse pour les systèmes de recommandation de services Web (Deng et al., 2016 ; Luo et al., 2019c).

La QoS joue un rôle essentiel dans les systèmes de recommandation de services Web. Elle est souvent utilisée pour décider du succès du service Web pour répondre aux exigences des utilisateurs (Bouguettaya et al., 2017). Les valeurs de QoS sont données soit par le fournisseur du service, soit par un système de monitoring externe ou des réseaux sociaux. Dans le premier cas, les mesures de qualité de service peuvent être obsolètes, et ne reflètent donc pas la réalité du service à l'instant présent. Dans le deuxième cas, ce sont des systèmes lourds, ayant souvent un impact négatif sur les performances des services supervisés, qu'il faudra mettre en place. Cependant, il est extrêmement très difficile pour les utilisateurs de sélectionner un service Web particulier, parmi tant d'alternatives possibles, d'où la nécessité d'une technique pour aider ces utilisateurs à trouver des services Web qui les sont appropriés.

Par ailleurs, les modèles d'apprentissage automatique et/ou profond ont récemment montré un grand potentiel pour l'apprentissage de représentations efficaces en offrant des performances considérables aux systèmes de recommandation en générale (Smirnova and Vasile, 2017 ; Strub et al., 2016), et à ceux de services Web en particulier (White et al., 2019 ; Wu et al., 2017c ; Yin et al., 2020). Pour ces approches, les caractéristiques des utilisateurs et des services ainsi que leurs relations latentes peuvent être apprises de manière supervisée ou non supervisée. Il devient donc intéressant d'intégrer l'apprentissage automatique avec les techniques de filtrage collaboratif pour la recommandation et la prédiction de services Web.

Nous nous intéressons dans notre travail à la problématique de la recommandation des services Web basée sur l'utilisation de leurs valeurs de QoS. L'objectif de notre thèse est d'améliorer les performances de la prédiction dans les systèmes de recommandation des services Web. Nous nous intéressons essentiellement à l'utilisation de l'approche du filtrage collaboratif combinée avec les approches d'apprentissage automatique et d'apprentissage profond pour pouvoir extraire les valeurs non disponibles de QoS d'un service Web ; tout en prenant en considération d'autres informations pour surmonter les problèmes de démarrage à froid (ou *cold start problem*) (Lika et al., 2014) et de la rareté de données (ou *data sparsity*) (Pan et al., 2010) relatifs aux systèmes de recommandation.

Ainsi, nous pouvons résumer la problématique abordée, dans notre thèse, en plusieurs grandes interrogations :

Q.1 Choix du modèle : Quels sont les modèles les plus appropriés, issus du domaine d'apprentissage automatique et/ou profond, que nous pouvons les utiliser afin de traiter la problématique de l'estimation des valeurs de QoS manquantes dans le contexte des services Web. Notons aussi que le modèle choisi doit être en mesure d'estimer la QoS en tenant compte des critères spatiales qui décrivent le contexte d'utilisation du composant logiciel.

Q.2 Rareté de données : Comment faire face au problème de rareté de données survenant en raison d'un taux d'interactions, en termes d'invocations de services, très faible de la part des utilisateurs. C'est-à-dire, souvent les utilisateurs n'évaluent pas la totalité des services disponibles, mais plutôt qu'un sous ensemble réduit de ces derniers.

Q.3 Démarrage à froid : Comment traiter ce problème qui survient quand un système de recommandation peine à fournir des prédictions dû à un manque d'informations, et qui est relatif à l'ajout de nouveaux utilisateurs et/ou de services.

Q.4 Paramétrage du modèle : Une fois les modèles architecturaux convenables choisis ; nous devons s'intéresser à la problématique du paramétrage de ces modèles. Nous partons du principe que de nombreux algorithmes d'apprentissage automatique dépendent étroitement des choix des hyperparamètres. Ces choix ont un impact énorme sur les performances de la prédiction de la QoS.

1.3 Contributions

Cette thèse décrit une étude approfondie sur la problématique d'estimation des valeurs des différentes propriétés de la QoS pour les services Web. Dans cette thèse, nous présentons une approche pour l'amélioration des performances dans la prédiction de la QoS en se concentrant sur l'utilisation des techniques et méthodes issues du domaine d'apprentissage automatique, le tout dans un contexte de filtrage collaboratif. A cette fin, nous proposons à travers la présente thèse un ensemble de contributions de nature théorique et expérimentale. Nous résumons ci-dessous nos principales contributions :

1) *L'élaboration de l'état de l'art* : Le premier objectif consiste en la réalisation d'un état de l'art autour de notre domaine d'intérêt. Celui-ci a évidemment pour idée principale d'explorer le paysage en termes des travaux de recherches sur la prédiction de la QoS pour les services Web à base de filtrage collaboratif. Nous avons scindé cette réalisation en deux parties principales. Dans la première partie, nous avons étalé l'existant des travaux sur la prédiction de la QoS dans des approches basées voisinage ainsi que celle basées sur les séries chronologiques. Tandis qu'une deuxième partie est consacrée à la prédiction de la QoS via l'utilisation d'un filtrage collaboratif à base de modèles ainsi que celui basée sur les approches hybrides. Une étude comparative des différents travaux existants, selon des critères d'évaluation que nous avons définis, nous a permis d'une part de dégager les avantages et les limites de chaque approche, et d'autre part de mieux positionner notre approche pour voir comment il serait concevable d'améliorer ce domaine de recherche.
2) *L'apprentissage automatique pour la prédiction de la QoS* : Proposition d'une première contribution pour traiter le problème de l'estimation des valeurs d'attributs de qualité des services Web (Smahi et al., 2018). Cette contribution est basée sur une approche d'apprentissage non supervisé via l'utilisation des auto-encodeurs (Hinton, 2006 ; Vincent et al., 2008). L'idée de base est de proposer un réseau de neurones artificiel symétrique permettant de copier les données disponibles, en termes de QoS, à partir de la couche d'entrée vers la couche de sortie en passant par une seule couche cachée du réseau artificiel. Cette proposition est basée sur un modèle de factorisation matricielle où l'idée de base consiste en l'apprentissage d'un modèle compact (facteurs latents) pour, respectivement, les utilisateurs et les services. La prédiction de la QoS est, par la suite, représentée par un produit scalaire entre les modèles latents de l'utilisateur et du service. Ce mécanisme nous a permis d'apprendre un modèle compact à la fois pour les utilisateurs et les services, tout en réduisant les problèmes liés à la parcimonie des données.
3) *L'apprentissage profond au service de la prédiction de la QoS* : Une deuxième contribution (Smahi et al., 2021) propose une approche basée sur des modèles d'apprentissage profond pour la prédiction de la qualité de service. L'idée est de combiner l'utilisation d'un réseau de neurones profond avec, à la fois, un modèle de factorisation matricielle et une technique de partitionnement. Le modèle de factorisation matricielle profond que nous avons choisi est basé sur un auto-encodeur avec plusieurs couches cachées. Tandis que l'algorithme de partitionnement a été mis en œuvre via l'utilisation des cartes auto-adaptatives de Kohonen (Kohonen, 2012). Ces réseaux de classification, connues aussi sous le nom de cartes auto-organisatrices (ou *Self Organizing Map - SOM*), représentent une catégorie de réseaux de neurones artificiels qui appartiennent aux méthodes d'apprentissage non-supervisées.
4) *L'Adversarial auto-encodeur* : Une troisième contribution consiste en une combinaison entre l'architecture des auto-encodeurs et celle des réseaux antagonistes génératifs (GAN) (Goodfellow et al., 2014). Pour des soucis d'amélioration des performances de prédiction, la phase d'apprentissage pour l'auto-encodeur est réalisée via l'utilisation d'un réseau GAN. L'objectif est de combiner la puissance de reconstruction relative aux auto-encodeurs avec celle d'échantillonnage des réseaux GAN.
5) *Mécanisme de gestion relatif au problème du démarrage à froid* : Le démarrage à froid par rapport à un nouvel utilisateur et/ou service est souvent un aspect handicapant pour les systèmes de prédiction et de recommandation. Afin de dépasser cette difficulté, nous proposons un mécanisme de solution via une utilisation explicite des différentes informations spatiales disponibles et qui

sont relatives à l'ensemble des utilisateurs et/ou services. L'algorithme proposé est basé sur les résultats de la mise en œuvre des cartes de Kohonen combiné avec le modèle auto-encodeur profond que nous avons déjà entraîné.

1.4 Organisation du manuscrit

Le contenu de la thèse est constitué de deux parties principales. La première partie, intitulée « Prérequis pour l'estimation de la QoS pour les services Web », introduit le contexte général de notre travail. Dans cette partie nous présentons un survol sur les différents axes de recherches autour de notre thématique de recherche. Tandis que dans la seconde partie, intitulée « Estimation de la QoS pour les services Web », nous regroupons toutes nos contributions pour faire face aux problèmes invoqués précédemment.

Tableau 1.1 – Proportions du nombre de pages ainsi que le nombre de références bibliographiques par chapitre

Partie	Chapitre	Pages	% sur 131 pages	% sur 346 références
Introduction générale		6	4.58%	9.75%
Prérequis pour l'estimation de la QoS pour les services Web	Services Web et attributs de QoS	15	11.45%	13.21%
	Le filtrage collaboratif noyau de la prédiction	20	15.27%	13.84%
	L'apprentissage profond	22	16.79%	17.3%
		57	**43.51%**	**44.34%**
Estimation de la QoS pour les services Web	Prédiction de la QoS par FC : état de l'art (Partie 1)	21	16.03%	34.91%
	Prédiction de QoS par apprentissage profond	27	20.61%	18.55%
	Expérimentations et évaluation des performances	17	12.98%	7.55%
		65	**49.62%**	**61.01%**
Conclusion générale		3	2.29%	0%

Le tableau 1.1 représente un listing des différents chapitres avec une vue statistique en termes de nombre de pages ainsi que le nombre de références utilisées pour chaque chapitre par rapport à l'ensemble du rapport.

Partie 1 : Prérequis pour l'estimation de la QoS pour les services Web Cette partie décrit les différents domaines relatifs au contexte globale de nos travaux. Elle est composée par trois chapitres :

Chapitre.2 Services Web et attributs de QoS
Dans ce premier chapitre, nous présentons un aperçu général sur les principes de la technologie des services Web. Nous évoquerons, en premier lieu, les concepts de base sur l'architecture orientée services, ainsi que son modèle à travers les différents standards sous-jacents basés sur la technologie XML. Nous concluons, ce chapitre, par une présentation détaillée de la notion de la qualité de service (QoS) au sein du modèle des services Web.

Chapitre.3 Le filtrage collaboratif noyau de la prédiction
Ce chapitre est consacré à la problématique de la prédiction basée sur le filtrage

collaboratif pour les systèmes de recommandation. Il présente les principes de base du filtrage collaboratif, suivi par les différentes classifications des méthodes utilisées dans le filtrage collaboratif. Nous nous intéressons aussi, dans ce chapitre, à présenter la différence qu'existe entre la notion de recommandation et prédiction dans le contexte des systèmes de recommandation ; ainsi que la notion des mesures d'évaluation utilisées pour valider la performance des prédictions et des recommandations générées.

Chapitre.4 L'apprentissage profond
Dans ce chapitre, nous évoquons les concepts et notions théoriques utilisés en apprentissage automatique ainsi qu'en apprentissage profond. Nous commençons par l'apprentissage automatique, où nous présentons les origines et l'historique des réseaux de neurones tout en fournissant quelques définitions sur les concepts de base. Ensuite, nous passons en revue les différentes architectures utilisées en apprentissage profond, telles que, les réseaux de neurones convolutionels et récurrents. Enfin, nous conclurons par la présentation de quelques algorithmes de classification non supervisée de données, et donnant une importance particulière à la classification à base des cartes auto-adaptatives de Kohonen.

Partie 2 : Estimation de la QoS pour les services Web Cette partie inclut trois chapitres :

Chapitre.5 Prédiction de la QoS par FC : état de l'art (Partie 1) Dans ce chapitre, nous identifions les différentes méthodes et techniques existantes, dans l'état-de-l'art, pour la prédiction de la QoS. Nous présentons en premier lieu notre stratégie de recherche bibliographique que nous avons adopté afin de collecter les articles les plus pertinents, et qui nous permettrons, par la suite, de dresser un état-de-l'art assez complet autour de note thématique de recherche. Dans ce chapitre, nous nous sommes intéresses, en premier lieu, aux travaux relatifs au filtrage collaboratif à base de séries temporelles. En second lieu, nous penchons vers les travaux utilisant des approches de filtrage collaboratif basées sur le voisinage.

Chapitre.6 Prédiction de QoS par apprentissage profond La première partie du chapitre se veut une continuité du chapitre précédent. En effet, et en raison du nombre assez important de travaux sur la prédiction de QoS basée sur les séries chronologiques ainsi que sur le voisinage ; la présentation de l'état de l'art sur les travaux de prédiction utilisant le filtrage collaboratif à base de modèles a été décalée dans cette partie. La deuxième partie du chapitre est consacré exclusivement à la description de notre approche pour résoudre le problème de prédiction de la QoS pour les services Web.

Chapitre.7 Expérimentations et évaluation des performances Dans le dernier chapitre du manuscrit, nous présentons les différentes comparaisons sur la précision des algorithmes de prédiction que nous avons proposées par rapport aux approches de pointe actuelles. La première partie décrit les jeux de données ainsi que les métriques que nous avons utilisés. Nous détaillerons, par la suite, nos arguments sur le choix de configuration des différents hyperparamètres. Enfin, nous présentons nos différentes évaluations expérimentales.

En dernier lieu, une conclusion générale fournie un aperçu des sur les principales idées de nos propositions. A l'occasion, nous reprenons certaines réflexions dans le but de mettre en avant les principales contributions et d'identifier grandes orientations des futures recherches.

Première partie

Prérequis pour l'estimation de la QoS pour les services Web

Chapitre 2

Services Web et attributs de QoS

Sommaire

2.1 Introduction 8
2.2 Paradigme de l'approche orientée services 9
2.2.1 Définition et historique 9
2.2.2 Principes de SOA 9
2.3 Les bases des services Web 10
2.3.1 Notions de service 10
2.3.2 Notions de service Web 10
2.3.3 Modèle de base des service Web 11
2.3.4 Architecture en couches des services Web 12
2.4 La description des services Web 13
2.4.1 Description SOAP 13
2.4.2 Description REST 14
2.4.3 REST vs SOAP 17
2.5 Attributs de qualité pour les services Web 18
2.5.1 Définition 18
2.5.2 Paramètres de QoS 19
2.5.3 Classification des attributs de QoS 20
2.6 Conclusion 22

2.1 Introduction

Les services Web sont un ensemble de technologies permettant de connecter les différents appareils informatiques les uns avec les autres via le réseau Internet ; cela pour un souci de partage et d'échange de données et d'information. Ils peuvent être considérés comme les briques de base pour la construction d'un système basé sur une architecture orientée service. La mise en place de tel modèle nécessite un ensemble de protocoles et de standards, dont la plupart sont basés sur la technologie XML, pour exécuter des fonctions ou des processus métier.

Dans ce chapitre, nous introduisons dans un premier temps le paradigme émergeant de l'architecture orienté service, en présentant toutes les notions liées à cette architecture. Nous exposons dans un deuxième temps les concepts de base qui sont utilisés pour la mise en place d'un modèle des service Web. La troisième partie sera consacrée exclusivement, au différents standards, basés sur la technologie XML, permettant la description des service Web. Ce chapitre sera conclu par la présentation de la notion de la qualité de service au sein du modèle des services Web.

2.2 Paradigme de l'approche orientée services

Actuellement, et afin de bâtir des applications, notamment des applications distribuées, l'utilisation de l'architecture SOA est largement répondue. En effet, le fait que la SOA considère les services comme des briques de base pour architecturer une application, a énormément réduit la nécessité de construire à nouveau des composants logiciels, où la reconstruction de ces derniers entraîne, par conséquence, des fonctionnalités redondantes.

2.2.1 Définition et historique

L'architecture orienté services (ou *Service Oriented Architecture*), abrégée en SOA, a été décrite dans (Kurbel, 2008) comme étant « *une architecture logicielle qui définit l'utilisation de services pour résoudre les tâches d'un système logiciel donné* ». Une deuxième définition, plus technique, est proposée dans (Ben Halima, 2009), et qui considère la SOA comme « *une approche architecturale permettant la création des systèmes basés sur une collection de services développés dans différents langages de programmation, hébergés sur différentes plates-formes avec divers modèles de sécurité et processus métier* ».

À noter que la SOA est apparue pour surmonter les limites rencontrées dans les architectures distribuées. Cette architecture n'est pas un simple effet de mode, elle s'inscrit plutôt dans la continuité logique des multiples tentatives d'homogénéisation dans les systèmes d'informations, d'intégration d'applications, de distribution de traitements, de répartition de données, etc. (Ben Halima, 2009).

Le concept SOA est nécessaire pour permettre l'informatique orientée services (SOC). Alors que les paradigmes précédents de "construction d'applications logicielles dépendent de composants ou d'objets, le moyen de construire des applications logicielles dans SOA sont des services.

2.2.2 Principes de SOA

L'objectif principal derrière l'utilisation des architectures orientées service s'inscrit dans la dynamique de transformation des systèmes d'information, en augmenter leur capacité à faire face à des situations imprévues. En d'autres termes, elle permet d'appuyer leur aptitude réactive à s'adapter rapidement aux différents changements de leur environnement extérieur. Cette réactivité se traduit en processus d'affaires flexibles, et elle est réalisable grâce au potentiel intégrateur spécifiques à l'architecture orientée service.

Cette intégration doit se conformée à divers principes de gestion des services influençant directement le style de conception d'une solution logicielle ainsi que son comportement intrinsèque. En considérant les ressources, manipulées par les systèmes d'information, comme étant des services ; plusieurs principes peuvent être appliquée, tels que, l'encapsulation, l'abstraction, réutilisation, la composition, la découverte, etc.

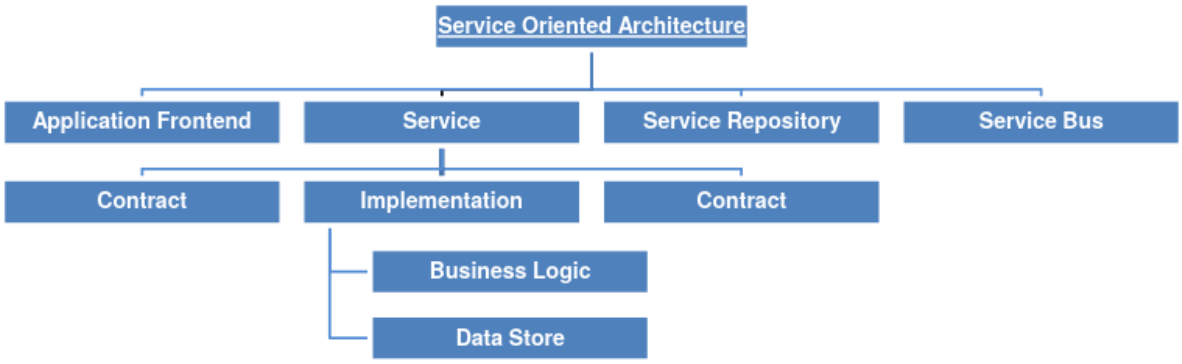


Figure 2.1 – Les quatre concepts de base de SOA (Rosen et al., 2012)

Comme le montre la Figure 2.1, une architecture orientée service considère principalement les quatre éléments suivants : (1) application de première ligne (ou *application front-end*), (2) un ensemble de services, (3) un référentiel de services contenant les spécifications de services, ce qui facilite leur la recherche par la suite, et (4) un bus de service qui fournit un mécanisme aux services pour leur interaction (Rosen et al., 2012).

Chaque service est caractérisé par un contrat définissant ses fonctionnalités. Un service, et pour être créer, doit implémente au moins une interface, qui lui permet d'être répertorier dans un référentiel, et par la suite, d'exposer son potentiel fonctionnel. Ces quatre éléments de base seront discutés en détails dans tout au long de ce chapitre.

2.3 Les bases des services Web

Le service Web est considéré comme étant une mise en œuvre ou une implémentation de l'architecture orientée services (Huhns and Singh, 2005).

2.3.1 Notions de service

En informatique, le terme *service* prend des significations différentes par rapport à de nombreux domaines. De ce fait, en littérature il n'existe pas une définition universelle commune, mais plutôt des définitions spécifiques aux différents domaines de recherches. Cependant, une définition, plutôt abstraite, de qu'est-ce qu'un service est donné par (Zhang et al., 2007). En effet, selon cette définition, le terme *service* fait référence à un type d'interaction basé sur l'existence d'une relation entre l'utilisateur du service et le fournisseur de celui-ci ; cela afin d'atteindre un certain objectif de solution. En d'autres termes, un service fait référence à une seule et unique interaction entre l'usager et le fournisseur de services.

Les services dans une SOA ont de nombreuses caractéristiques telles que (Erl, 1900) :

— Couplage faible : le demandeur du service ne doit pas se préoccuper ni sur la manière dont un service a été mis en œuvre ni sur l'endroit où se trouve le service ;

— Visibilité : cela signifie que le demandeur d'un service peut découvrir et trouver le service Web qui le convient en interrogeant un référentiel de services dédié ;

— Dynamiquement connectable : signifie que le demandeur d'un service peut se connecter au service Web en utilisant les informations du contrat au moment de l'exécution ;

— Interopérable : signifie qu'une application logicielle peut invoquer et accéder à un service même si ce dernier soit écrit dans un langage de programmation différent ou se trouve sur une plate-forme différente ;

— Accessibilité : le client peut invoquer un service en utilisant un réseau informatique (généralement Internet).

2.3.2 Notions de service Web

Dans le contexte des SOA, un service Web est considéré comme une brique de base pour la construction d'un composant logiciel possédant des caractéristiques qui le distinguent des composants logiciels traditionnels. Cela inclut la possibilité de sa réutilisabilité, son automaticité, en plus de sa caractéristique de haute portabilité. Dans la littérature des architectures orientées service, plusieurs définitions, pour le terme de service Web, ont été proposées :

Définition 1 Les services Web ont été définis par (Jin et al., 2002) comme suit : « *Web services are an Internet based applications that communicate with other applications to offer business data or functional services programmatically* » ; que nous la traduisons comme suit : « *les services Web sont des applications qui communiquent entre elles via le support Internet, afin d'offrir automatiquement des informations commerciales ou des services fonctionnels* ».

Définition 2 Une deuxième définition, proche à la première, est proposée par (Chavda, 2004), qui préconise que les services Web sont « *A programmable application component that can be accessed over the internet and used remotely* » ; c'est-à-dire que les services Web sont définis comme étant des composants d'applications programmables utilisés à distance et accessibles via Internet.

Définition 3 Une définition plus technique, s'inscrivant dans le cadre d'une implémentation possible des architectures orientées service, est donnée par le consortium W3C [2]. En effet, les services Web est considéré comme une application, ou composant logiciel qui doit être identifié par un URI [3], et accessible par Internet ou par l'intermédiaire d'un réseau privé. Il est conçu pour favoriser l'interopérabilité entre systèmes informatiques hétérogènes. Cette interopérabilité est garantie par une interface utilisant plusieurs normes d'échange de messages basée sur XML, à savoir les normes SOAP et WSDL.

En résumé, et par rapport à l'ensemble des définitions présentées précédemment, nous pouvons conclure que les services Web sont des logiciels conçus et destinés pour faciliter la collaboration entre machines. Les services Web permettent à des systèmes hétérogènes de s'intégrer en présentant leurs fonctionnalités qui peuvent être invoquées, dans le futur, via des protocoles dédiés et par l'intermédiaire d'un réseau informatique.

2.3.3 Modèle de base des service Web

Les services Web sont une technologie dans une architecture orientée services où le « service » est la base sur laquelle tous les fondements du modèle des services Web ont été conçus.

En première position, et via un processus appelé « *publish* », les services peuvent être annoncés par un fournisseur de services à un référentiel de services, tel que l'Universal Description Discovery and Integration, connu aussi sous l'acronyme UDDI. La mise en œuvre de ce processus est garantie par l'utilisation d'une norme d'échange de messages basée sur XML, appelée Web Services Description Language et abrégée par WSDL (le langage est présenté da la section 2.4.1).

En second position, un service peut, en outre, être découvert par un demandeur de service, aussi connu sous les noms de consommateur, d'utilisateur, ou usager de service, via un processus appelé « *find* ». Ce dernier utilise, essentiellement, l'annuaire de services UDDI afin de localiser les services appropriés, décrits, bien évidemment, dans la description WSDL du service.

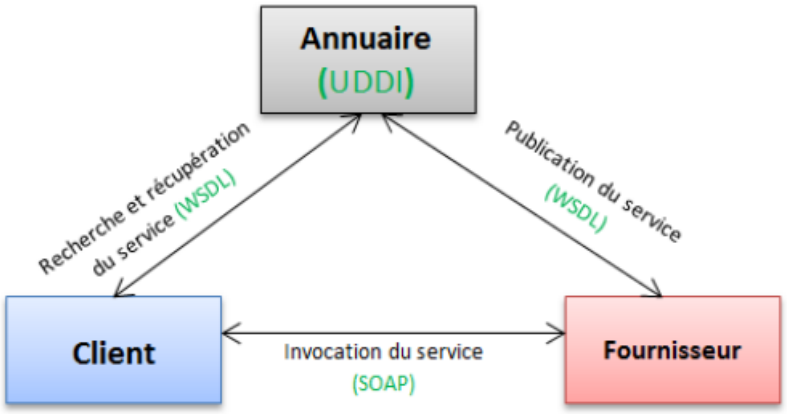


Figure 2.2 – Modèle d'interaction pour les services Web (version adaptée de Papazoglou (2008))

En dernier lieu, un service est invoqué par le biais d'un processus appelé « *bind* », en envoyant une requête au service distant et en recevant une réponse sur le réseau via un protocole d'échange d'information appelé SOAP, acronyme de Simple Object Access Protocol, basé sur échange de messages

2. W3C abréviation du World Wide Web Consortium. Il a été fondé en octobre 1994, par Tim berners Lee, pour promouvoir la compatibilité des technologies du Web. www.w3c.org

3. Uniform Resource Identifier

XML. L'ensemble des interactions, en termes opérations de publication, de recherche et d'invocation, est représenté dans la Figure 2.2.

De ce qui a été présenté ci-dessus, nous remarquons que le modèle des services Web fait intervenir trois catégories d'acteurs : le fournisseur de service, l'annuaire des services et le client.

— Le fournisseur de service : correspond au propriétaire du service. D'un point de vue technique, il est constitué par la plate-forme d'accueil du service ;

— Le client : correspond au demandeur de service. D'un point de vue technique, il est constitué par l'application qui va rechercher et invoquer un service. L'application cliente peut être elle-même un service web.

— L'annuaire des services : correspond à un registre de descriptions de services offrant des facilités de publication de services à l'intention des fournisseurs ainsi que des facilités de recherche de services à l'intention des clients.

2.3.4 Architecture en couches des services Web

Les services Web peuvent être considérés comme un ensemble de technologies ou de normes qu'on peut les empilées en un système de couches, comme le montre la Figure 2.3. Cette représentation permet de classifier les différentes données manipulées dans le contexte des services Web. Cette architecture est très basique, elle permet de décrire les interactions qu'existent entre les trois acteurs principaux que nous avons mentionnés précédemment (voir Figure 2.2). Cependant, Il existe d'autres architectures pour les services Web qui ajoutent plusieurs couches aux celles de l'architecture de base, telles que les couches de composition de découvertes, ou de présentation pour le modèle des services Web.

Les couches de la pile technologique des services Web, présentées dans la Figure 2.3, sont les suivantes :

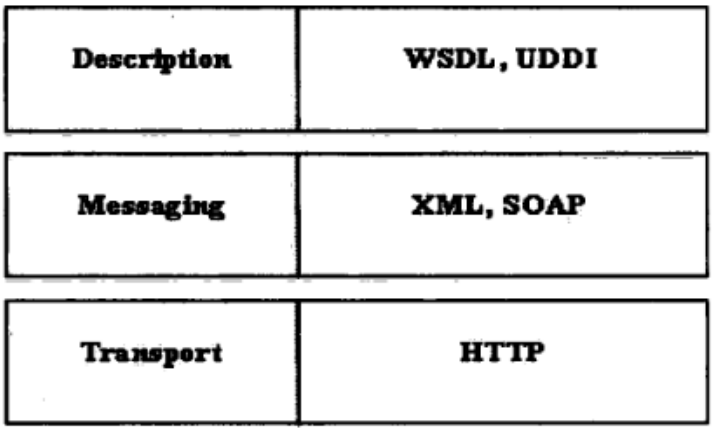


Figure 2.3 – La pile des technologies pour les services Web Hanna (2008)

Couche de transport des services La couche de base dans cette pile est celle de transport. Comme les Services Web sont essentiellement un mécanisme d'échange de messageries entre les applications dans une architecture distribuée ; l'utilisation des technologies de transport dédiées, et qui sont développés dans le domaine des réseaux, comme le protocole HTTP, sont le meilleur moyen pour assurer et garantir leur fonctionnement. Cette couche regroupe tous les les protocoles de transport de bas niveau, tels que HTTPS, SMTP, FTP, etc., nécessaires pour la prise en charge des requêtes et des réponses échangées entre services.

Couche de communication des services Au niveau de la couche de messagerie, nous trouverons les technologies fondamentales des services Web. Elle spécifie, les protocoles d'échanges, à base de documents XML, utilisées par le modèle des services Web, à savoir, SOAP ou REST. Le protocole SOAP

sera discuté dans la section 2.4.1, tandis que l'architecture REST est discutée dans la section 2.4.2 qui suit. Nous notons que lors de la phase de proposition d'approche et d'expérimentations de notre thèse, nous avons adopté seulement le modèle SOAP comme unique protocole d'échange pour les des service Web. L'autre modèle, à savoir REST, ne fait pas l'objet d'étude dans nos travaux.

Couche de description des services Cette couche est responsable de la description d'un service Web en détaillant le profil fonctionnel/non fonctionnel du service Web.

2.4 La description des services Web

Comme mentionné dans (Daigneau, 2012), il existe deux principales méthodes qui sont utilisées pour décrire les services Web, à savoir SAOP et REST. Dans la partie qui suit, nous présenterons, en détail, ces deux types de descriptions.

2.4.1 Description SOAP

Dans la définition des services Web proposée par le consortium W3C, le protocole SOAP est explicitement considéré comme une technologie des services Web, permettant à une interaction machine à machine d'effectuer des échanges de messages. En effet, SOAP est un protocole de messagerie qui exploite, à la fois les standards (HTTP/SMPT/POP) [4] comme protocole de communication, et XML [5] comme langage de représentation.

Le principe derrière l'utilisation de ces différentes normes est de pouvoir masquer l'hétérogénéité qui peut exister dans les plateformes distribuées et gérer, éventuellement, les différentes connexions entre les demandeurs de services et leurs fournisseurs. La mise œuvre de cette interaction est accompli par un échange de messages, sous forme de documents XML, entre les usages et les fournisseurs de services. Ce process est connu sous le nom d'enveloppes SOAP.

Comme indiqué dans la Figure 2.4, un message SOAP est composé de trois éléments distincts :

— l'**enveloppe**, qui peut être considérée comme un prologue permettant de spécifier, la version SOAP utilisée, ainsi que les règles d'encodage-décodage des messages ;

— l'**en-tête** du message permettant de spécifier certaines directives pour le traitement du message SOAP ;

— le **corps** qui répertorie les données (ou paramètres) utilisées dans un appel de procédure à distance effectué par le destinataire final ;

— Un quatrième élément peut être considéré, à savoir, les **pièces jointes** dans le cas où l'échange devrait intégrer des données qui ne peuvent pas être standardisées par le langage XML.

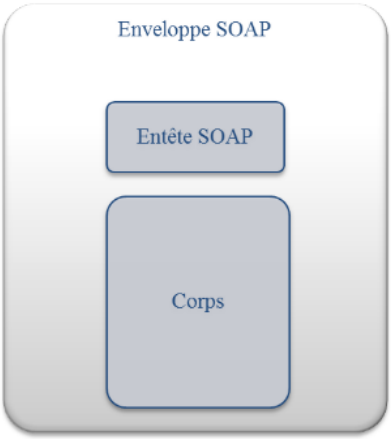
L'utilisation du protocole SOAP considère que le service doit être défini et répertorié préalablement. Ceci est réalisé, comme mentionné précédemment, en utilisant le référentiel de service UDDI. La Figure 2.2 illustre le processus d'appel des services Web dans le protocole SOAP.

Afin de favoriser l'automatisation des phases préliminaires à l'invocation des services Web, d'autres standards, en parallèle avec le protocole SOAP, peuvent être utilisés. Ainsi, le langage WSDL peut être exploiter, par le fournisseur de services, afin de donner les différentes descriptions techniques de ces services Web. Cette norme est utilisée, essentiellement, pour lister les informations relatives aux fournisseurs de services ainsi que leurs services qui sont disponibles. Ces descriptions peuvent être interprétées, sans la moindre intervention humaine, par une entité distribuée désireuse de chercher et ensuite d'invoquer un service particulier. À noter que, le WSDL est représenté, lui aussi, à l'aide du schéma XML (Papazoglou, 2008).

La Figure 2.5 présente un exemple de WSDL qui affiche les parties principales, à savoir, les fonctions assurées par le service et le type de données utilisé, ou en plus les différentes interfaces du service.

4. Hyper Text Transfer Protocol, Simple Mail Transfer Protocol et Post Office Protocol
5. Extensible Markup Language

Figure 2.4 – Composition du message SOAP Alonso et al. (2004)




```
<wsdl:definitions name="PurchaseOrderService"
    targetNamespace=http://supply.com/PurchaseService/wsdl
    ... ...
    xmlns:tns="http://supply.com/PurchaseService/wsdl">
    <wsdl:types>
<xsd:schema
            targetNamespace="http://supply.com/PurchaseService/wsdl">
            ... ...
          <xsd:complexType name="POType">
          ... ...
          </xsd:complexType>
          <xsd:complexType name="InvoiceType">
          ... ...
          </xsd:complexType>
          </xsd:schema?
    </wsdl:types>
<wsdl:message name="POMessage">
<wsdl:part name="PurchaseOrder" type="tns:POType"/>
</wsdl:message>
<wsdl:message name="InvMessage">
          <wsdl:part name="Invoice" type="tns:InvoiceType"/>
</wsdl:message>
<wsdl:portType name="PurchaseOrderPortType"
        <wsdl:operation name="SendPurchase">
            <wsdl:input message="tns:POMessage"/>
            <wsdl:output message =" tns:InvMessage"/>
        </wsdl:operation>
:
</wsdl:portType>
</wsdl:definitions>
```

Abstract data type definition

Data that is sent

Data that is returned

Port type with one operation

Figure 2.5 – Exemple de fichier WSDL Papazoglou (2008)

Le protocole SOAP décrit le processus de de mappage des informations des services dans WSDL à l'UDDI d'une manière claire et précise. En effet, WSDL est utilisé dans l'UDDI pour fournir des informations sur la description du service. Ce dernier est considéré comme des « pages jaunes » pour les clients afin de trouver une description WSDL des services Web qu'ils cherchent. Ces messages sont utilisés pour invoquer des services Web en encapsulant la requête SOAP dans le protocole de transport, ceci est réaliser en transmettant le message SOAP dans le corps d'une requête HTTP POST.

2.4.2 Description REST

Il est possible de distinguer différents modèles d'architecture lorsqu'on parle d'architectures orientées service. En effet, REST (ou *Representational State Transfer*) est une des méthodes qui implémente ce principe, en fournissant un style d'architecture indépendant de la plate-forme et du langage. Elle a été introduite pour la première par fois Roy Fiedling dans sa thèse de doctorat (Fielding, 2000). Initialement, les travaux de développement du protocole les a commencé à partir de l'année 1994, en parallèle avec le développant des spécifications de HTTP/1.0 et du HTTP/1.1 qui seront utilisés, par la suite, dans l'architecture du Web.

Théoriquement, une bonne compréhension des concepts REST peut aider à construire des systèmes distribués à haute performance (Vinoski, 2007). Techniquement, la mise en œuvre des principes de l'architecture REST est assurée par une API [6] dite « *RESTful* » qui propose des services à différents clients hétérogènes (Fielding, 2000).

Principes de base REST

REST n'est ni une spécification ni un protocole, mais plutôt il représente un style architectural (ou *design guideline*) dans le domaine de l'ingénierie des systèmes distribués. En définissant un ensemble de contraintes, REST permet de construire une application devant fonctionner sur des systèmes distribués, notamment Internet. Cet ensemble de contraintes est représenté par une liste exhaustive de contraintes architecturales qui permet de manipuler et d'accéder aux données. En effet, cette liste doit être respectée en intégralité, où aucune violation de contrainte n'est tolérable (c'est-à-dire, si un système n'applique pas une de ces contraintes, il ne peut pas être considéré comme utilisant l'architecture REST).

L'architecture REST est basée, à la fois, sur des ressources et ses représentations. En effet, la ressource est un des concepts importants de REST. Elle est définie comme tout ce qui est identifiable et accessible par un URI (ou *Uniform Resource Identifier*). Ce dernier est utilisé pour représenter chaque ressource de manière unique afin qu'elle ne soit pas en conflit avec d'autres ressources (Papazoglou, 2008). En revanche, la représentation est l'information, ou la donnée, qui décrit l'état d'une ressource. Le format de données de représentation est également connu sous le nom de type de média tel que les documents HTML (Fielding, 2000).

La Figure 2.6 montre le style de communication dans l'architecture REST. Il est à mentionner qu'il n'y a pas besoin d'une description du service WSDL, ou d'un registre de service UDDI, comme dans le protocole SOAP. En fait, il s'agit d'une architecture client-serveur où les messages reçus des ressources en REST incluent toutes les métadonnées nécessaires et qui sont liées au service reçu. Cela rend le REST plus facile à modifier par rapport au protocole SOAP, où toute modification côté serveur nécessite la mise à jour du WSDL pour assurer la récupération des services côté client (Bloomberg, 2013).

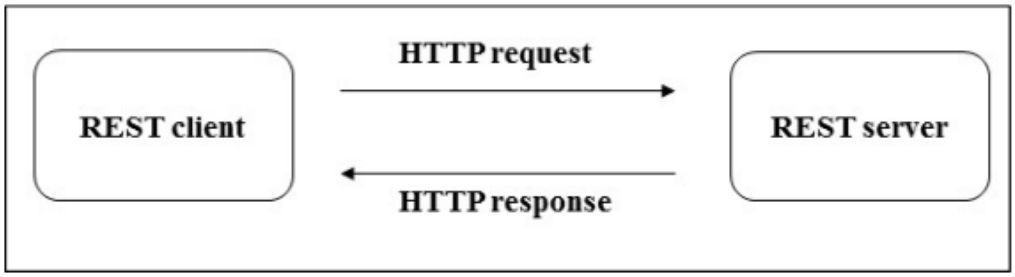


Figure 2.6 – L'invocation de service dans REST

Les services REST sont accessibles, depuis le serveur, à l'aide des requêtes HTTP, grâce aux quatre opérations de base CRUD (Create, Read, Update, et Delete) ou (Créer, Lire, Modifier et Supprimer) utilisées pour la persistance des données (Marinescu, 2017) (plus de détails sont présentées dans la section 6). Les messages HTTP peuvent être soit une requête d'un client vers un serveur, soit une réponse dans le sens inverse. Chaque message contient des champs d'en-tête HTTP et un corps de message HTTP (Fielding, 2000).

Caractéristiques de l'architecture REST

Le style architecturale REST pour les services Web, exige au total six contraintes architecturales lors de la construction d'une application distribuée.

6. Application Programming Interface

Client-serveur L'architecture client-serveur donne un avantage en termes d'évolutivité et de portabilité au service Web, en séparant les responsabilités entre le client et le serveur. Cette séparation permet aux composants d'évoluer d'une manière indépendante supportant ainsi de multiples domaines organisationnels nécessaires à l'échelle d'Internet. En effet, l'attribut d'évolutivité du système peut être amélioré par la simplification des composants serveurs. Cependant, l'attribut de portabilité de l'interface utilisateur sur plusieurs plateformes se retrouve aussi améliorée en découplant l'interface utilisateur du stockage des données à celle du service.

Sans état Le style architectural REST introduit la contrainte de services sans états (ou *Statelessness*). Ce principe stipule que l'état de session entre deux requêtes HTTP successives, en termes de communication client–serveur et gestion de session, n'est pas sauvegardé. En d'autres termes, cette contrainte impose que le serveur ne doit connaître aucun élément sur le client, et en même temps que la requête HTTP soit traitée de manière indépendante. La contrainte impose, aussi, que chaque requête envoyée par le client doit contenir toutes les informations nécessaires, relatives à son état pour qu'elle soit appréhendée par le serveur. Par conséquence, les attributs exprimant la tolérance aux échecs, la visibilité des interactions entre les composants et l'extensibilité, atteindrons leur niveau maximal car les requêtes sont complètes. Cependant une exception usuelle à ce mode sans état est la gestion de l'authentification du client, afin que celui-ci n'ait pas à renvoyer ces informations à chacune de ses requêtes.

Support avec mise en cache Connue en anglais par le terme« cacheable » ; cette caractéristique permet aux caches client et serveurs intermédiaires de réutiliser les données de réponse pour des requêtes précédentes similaires. Avec cette propriété les deux entités (clients et serveurs intermédiaires) peuvent mettre en cache les réponses du serveur principal. Ce dernier peut spécifier cette possibilité de mise en cache, d'une manière implicite ou explicite, afin de réduire le risque que les clients récupèrent des données obsolètes ou inappropriées en réponse à des requêtes ultérieures. L'objectif derrière cette contrainte est de réduire au minimum le nombre d'interactions entre les composants, et par conséquence, d'augmenter les performances du service Web.

Système en couches Un système en couches dans lequel les fonctionnalités sont divisées en couches hiérarchiques de telle sorte qu'un composant ne doit se soucier que des composants avec qui il en connexion directe. Cette caractéristique à un double avantages. D'un côté, elle permet de renforcer les politiques de sécurité, et de l'autre côté, elle améliore l'extensibilité du système à l'aide d'un mécanisme de répartition de charge et un cache partagé.

Code à la demande (optionnel) Cette contrainte permet d'améliorer la capacité du client en permettant au serveur de leur envoyer non seulement la description d'un état, mais également une logique de traitement (le code). Cette action est mise en œuvre via l'utilisation de scripts d'applets (Fielding, 2000), qui donne, temporairement, la main aux serveurs d'étendre ou de modifier les fonctionnalités du client en lui envoyant du code exécutable. L'avantage, pour cette contrainte, est qu'elle améliore l'extensibilité du système en réduisant le nombre de fonctionnalités que les clients mettre en œuvre. En revanche, cela a un impact sur la visibilité de l'organisation des ressources. C'est la raison pour laquelle, elle constitue une contrainte optionnelle dans une architecture REST.

Interface uniforme Cette caractéristique est primordiale pour la conception d'un système REST. Elle est définie par une interface entre le client et le serveur, permettant à chaque composant d'évoluer indépendamment en améliorant la simplicité et le découplage de l'architecture des services. Techniquement, les communications entre ces différents composants se font grâce au protocole HTTP. L'interface uniforme des ressources est constituée des quatre principaux méthodes CRUD, illustré par le tableau 2.1. Cela permet aux clients de se focaliser sur des ressources spécifiques qui sont identifiées par des URIs.

7. D'autres méthodes HTTP sont disponibles : HEAD, PATCH, CONNECT, OPTIONS et TRACE

Tableau 2.1 – Liste des méthodes HTTP, fréquemment utilisée, dans REST [7]

Méthode	Description
GET	Permet de demande le service du serveur ou de sélectionner une ressource du fournisseur à l'aide de son URI
POST	Permet l'envoie des données à traiter par le fournisseur, ce qui entraîne une création ou une mise à jour du service
PUT	Utiliser pour crée ou modifier une ressource
DELETE	Permet de supprime une ressource contenue dans le serveur

Il est important de mentionner que REST et SOA partagent la caractéristique de couplage lâche (ou *loose coupling*), qui permet le développement de systèmes distribués. Cette caractéristique est appliquée dans REST en utilisant l'interface uniforme par le biais des URI.

2.4.3 REST vs SOAP

Plusieurs études (Bora and Bezboruah, 2015 ; Markey and Clynch, 2013 ; Mohamed and Wijesekera, 2012 ; Mulligan and Gračanin, 2009) ont mis en évidence le contraste qu'existe entre le protocole SOAP et l'architecture REST. En effet, le point le plus mentionné, dans ces études, est la lourdeur occasionnée par l'utilisation du WSDL comme support de communication client-serveur pour le protocole SOAP. A l'opposé, cette caractéristique, non utilisée dans REST, est la justification de la composante légère de l'architecture REST, ce qui lui offre de meilleures performances par rapport à SOAP.

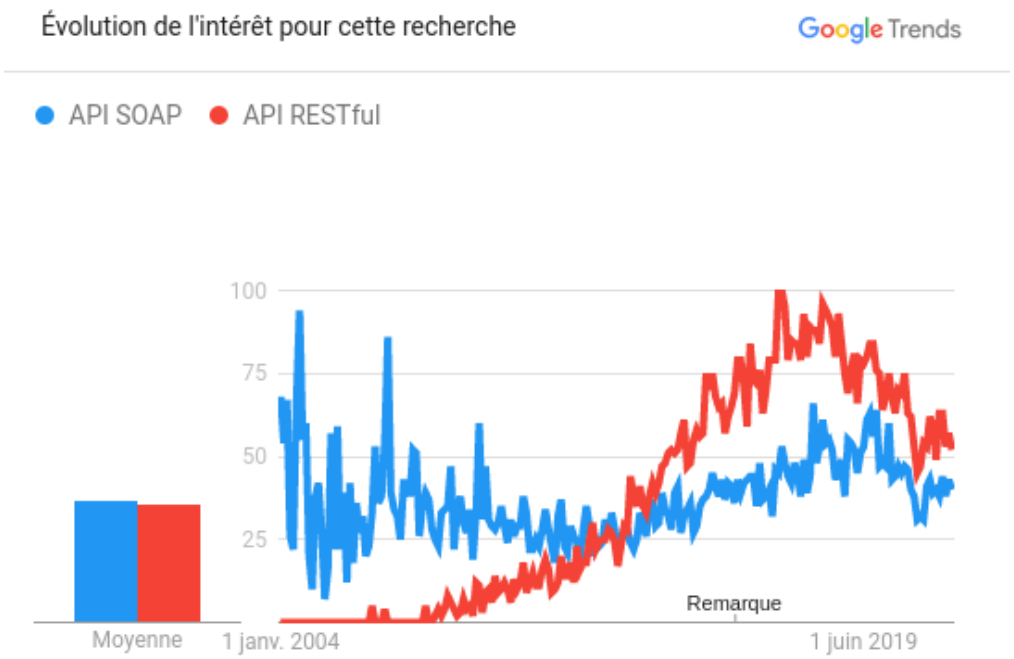


Figure 2.7 – Évolution de l'intérêt des termes RESTful API and SOAP API de 2004 à 2021 (Google Trends [8])

8. https://trends.google.com/

Les travaux dans (Mumbaikar et al., 2013) présentent une étude comparative entre l'utilisation de SOAP et REST dans les applications de conférence multimédia. Ils ont discuté de la nécessité de traiter les messages XML SAOP qui ne sont pas utilisés dans REST. L'étude a montré que les messages SOAP nécessitaient plus de bande passante et de consommation de ressources.

Pour conclure cette sous-section, et puisque les statistiques ne mentent pas ; la Figure 2.7 montre la tendance ou l'évolution de l'intérêt pour les termes « API SOAP » et « API RESTful » sur le moteur de recherche Google durant la période de 2004 à 2021. La figure démontre que les moyennes de recherches sont presque identiques bien que REST soit moins ancienne que le protocole SOAP. Mais surtout elle révèle un passage apparent des services SOAP aux services REST. Cette tendance est notamment due aux avantages offerts par REST tels que la simplicité, la facilité d'utilisation, un meilleur temps de réponse et une meilleure évolutivité du serveur (Velte et al., 2010) ; ce qui les rendent plus préférables dans les développements de l'industrie des API.

2.5 Attributs de qualité pour les services Web

À mesure que le nombre de services Web ayant des fonctionnalités similaires augmente, l'étape principale, pour un processus de recommandation des services Web basé sur les valeurs de QoS, consiste à manipuler et utiliser les attributs de QoS qui doivent être rassemblés et analysés tout au long de ce processus. Mais avant d'étudier davantage cette étape, il est nécessaire de connaître en détails les différents attributs de QoS utilisés dans le contexte des architectures orientées service.

2.5.1 Définition

La QoS est un mélange complexe et subjectif de plusieurs attributs ou facteurs et il n'existe pas de consensus dans la littérature sur une définition universelle ou une métrique unique pour quantifier la QoS. Bien que son importance omniprésente ait été, belle est bien, reconnue dans plusieurs domaines, le sens donné à la QoS varie selon les orientations des différentes communautés de recherches pour ces domaines.

Le concept de QoS a été introduit et utilisé pour la première fois dans la communauté des réseaux (Aurrecoechea et al., 1995). Dans ce contexte, la QoS vise explicitement à fournir certains niveaux de qualité pour les différentes caractéristiques d'une connexion réseau. Ces dernières peuvent être exprimées par le délai d'attente, la bande passante des liens du réseau, ou le nombre de paquets perdus lors des transmissions.

Le concept de QoS a ensuite été adopté dans la communauté des applications de gestion des ressources, notamment dans le domaine de la multimédia distribué (Hafid et al., 1996). Dans ce contexte, la QoS dans est définie, dans (Vogel et al., 1995), comme « *l'ensemble des caractéristiques quantitatives et qualitatives d'un système multimédia, nécessaires pour atteindre la fonctionnalité requise par l'application* ». En d'autres termes, la QoS vise à fournir aux usagers un niveau acceptable de qualité de présentation lors de l'accès à une ressource multimédia (la prise en charge de la QoS au niveau du réseau, fournir certaines garanties pour les ressources côté serveur, etc.).

Quant à son utilisation dans le contexte des services Web et les architectures orientées service, la QoS est considérée, par (Serrano et al., 2016) comme une garantie sur la capacité d'assurer un certain nombre d'exigences telles que la performance, la disponibilité, la fiabilité ou le coût. Selon (Viriyasitavat et al., 2012), la QoS peut être définie comme étant un accord ou un contrat qui est établi automatiquement entre le fournisseur et l'usager via un langage de spécification. Pour (Margolis, 2007), la QoS peut être élucidée par calcul précis de la performance globale d'un service Web en faisant référence à la fiabilité, à la sécurité, à la coordination des services et aux mises à jour d'exécution.

D'une manière générale, la QoS est considérée selon deux perspectives différentes. Du point de vue des fournisseurs de services, la QoS est considérée comme étant un excellent argument de vente pour leurs services et une opportunité d'attirer les usagers. Cependant, et du côté de ces derniers, la QoS est une indication, en termes d'exigences, du niveau de service auquel ils peuvent s'attendre et en même temps, elle constitue un critère primordial pour le choix du service le plus approprié.

2.5.2 Paramètres de QoS

Plusieurs paramètres de QoS applicables aux services Web peuvent être identifiés. Ces paramètres de QoS représentent l'aspect non-fonctionnel (Wiegers and Beatty, 2013) des services Web. En littérature, il existe un large éventail de propriétés de QoS ; souvent avec des interprétations et des définitions multiples et différentes. Nous listons ci-dessous, une catégorisation de ces attributs comme proposée dans (Zeng et al., 2007) :

— **La disponibilité** (ou *Availability*) (Hwang et al., 2007 ; Jaeger et al., 2004) est la probabilité que le système soit accessible pour une utilisation immédiate. Sa valeur est exprimée par un ratio entre la période où le service est actif et la durée totale durant laquelle l'utilisateur souhaite que le service soit actif. Selon (Ran, 2003), cette métrique est souvent en étroite relation avec celle de la fiabilité des services ;

— **Le prix** (ou *Price*) (Hwang et al., 2007 ; Liu et al., 2004) est le montant en argent qu'un usager doit payer au fournisseur de services pour chaque invocation d'un service Web ;

— **La fiabilité** (ou *Reliability*) (Costa and Demazeau, 1996 ; Ran, 2003) ou taux de réussite, correspond à la capacité d'un service Web à accomplir correctement ses fonctions requises. Il est exprimé par la probabilité que le service Web puisse être achevé avec succès (Hwang et al., 2007).

— **La capacité** (ou *Capacity*) (Costa and Demazeau, 1996 ; Ran, 2003) est le nombre maximum de requêtes traitées par le système simultanément tout en garantissant un certain niveau de performances ;

— **La latence** (ou *Latency*) (Mani, 2002) ce paramètre est utilisé pour désigner le temps écoulé entre l'invocation du service Web la réception d'une réponse. Elle peut être définie aussi comme le temps moyen garanti requis pour achever une demande de service. Dans certain cas (Hwang et al., 2007 ; Jaeger et al., 2004 ; Liu et al., 2004), cet attribut est connu sous le nom de temps de réponse du service Web ;

— **La performance** (ou *Performance*) (Costa and Demazeau, 1996 ; Ran, 2003) elle représente la vitesse à laquelle un service Web peut être exécuté. Souvent, un service Web est considéré performant dans le cas où le débit est élevé, et le temps de réponse est réduit.

— **La sécurité** (ou *Security*) (Jaeger et al., 2004) concerne la manière dont le service Web établit l'authentification pour les utilisateurs ou pour d'autres services Web. Elle concerne aussi la confidentialité dans le traitement des données. Et enfin, elle concerne la manière dont le service Web crypte les données.

— **La robustesse** (ou *Robustness*) (Costa and Demazeau, 1996) est la métrique mesurant le degré auquel un service Web peut fonctionner proprement même s'il reçoit des entrées invalides, incomplètes ou conflictuelles ;

— **La réputation** (ou *Reputation*) (Hwang et al., 2007 ; Liu et al., 2004) est définie comme la manière dont un service a été traité. Elle est considérée comme étant une propriété subjective car elle représente une mesure de la crédibilité du service. Cette dernière dépend principalement de l'expérience utilisateur, c'est-à-dire, à la façon d'utilisation du service Web par l'usager final ;

— **Le coût** (ou *Cost*) (Cardoso et al., 2004) ce paramètre englobe la quantité en termes de ressources, telles que temps d'exécution, espace mémoire réservé ou ressources humaines employées, que les services Web consomme ;

— **L'évolutivité** (ou *Scalability*) (Costa and Demazeau, 1996) est la propriété d'augmenter la capacité de calcul et de traitement du serveur des services Web afin de pouvoir traiter davantage les différentes invocations de clients.

— **Le débit** (ou *Throughput*) représente le nombre de requêtes que le service peut traiter pendant un intervalle de temps déterminé ;

À noter que d'autres facteurs de QoS sont définis dans la littérature dont la plupart sont des variantes ou combinaisons de la liste des attributs mentionnés ci-dessus.

2.5.3 Classification des attributs de QoS

Comme déjà mentionné précédemment, les services Web sont en croissance importante, et au fur et à mesure que le nombre de ces services augmente, le risque d'avoir des services Web ayant des fonctionnalités similaires augmente aussi. La première étape consiste à trouver les attributs QoS qui doivent être recueillis et analysés dans le processus de recommandation des services basés sur la QoS. La classification des paramètres QoS peut être utilisée comme méthode pour examiner les paramètres et faciliter le processus de recommandation. Alors que de nombreux articles parlent de moyens de spécifier la QoS pour les services Web et proposent des frameworks pour leur gestion, quelques-uns ont donné le temps d'essayer de classer les aspects QoS (Dobson et al., 2005 ; Menasce, 2002 ; Platenius et al., 2013 ; Sumra and Arulazi, 2003).

Dans (Sumra and Arulazi, 2003), l'auteur a créer, par analogie à la pile des couches définie pour les services Web, une classification en pile relative aux attributs de QoS. En effet la pile QoS, voir Figure 2.8, fournit une relation entre les technologies relatives aux services Web, divisées en couche de données, couche logique et couche de présentation et leurs concepts QoS appropriés.

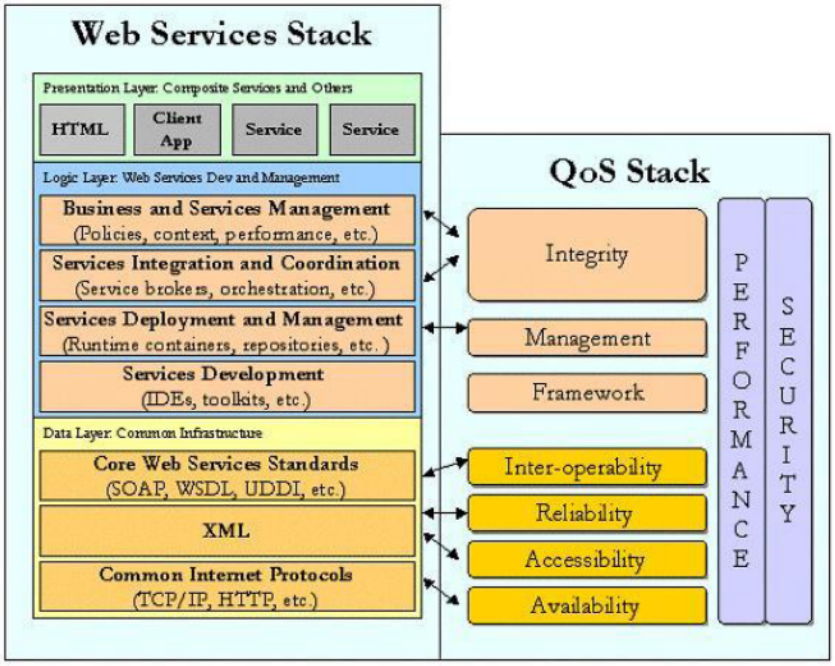


Figure 2.8 – La pile des couches des Services Web vs QoS Sumra and Arulazi (2003)

Dans (Dobson et al., 2005), nous trouverons une classification sémantique basée sur une ontologie, baptisée QoSOnt. Cette classification identifie deux classes d'attributs en termes de QoS : classe des QoS spécifiques, et classe des QoS génériques. Cette dernière catégorie est également divisée en paramètres mesurables et d'autres non mesurables. La classification est illustrée par la Figure 2.9.

Selon cette classification les attributs QoS sont considérés comme :

- **Attributs spécifiques**, Selon (Ben Halima, 2009) les paramètres de QoS spécifiques se rapportent à une application spécifique. La QoS spécifique peut être liée à des arguments relatifs à la logique métier des applications ;
- **Attributs génériques**, les paramètres QoS génériques peuvent être deviser en deux catégories différentes :
  - **Attributs mesurables** tels que la disponibilité, la fiabilité, le coût et le temps de réponse, qui représentent l'ensemble des paramètres de QoS qui peuvent être directement estimer ou l'estimation est basée sur une métrique composée par plusieurs autres métriques agrégées tel que la latence qui peut être calculée à partir du débit.
  - **Attributs non mesurables**, c'est tous les autres attributs qui n'appartiennent aux classes précédentes, tels que le prix, la réputation ou la sécurité.

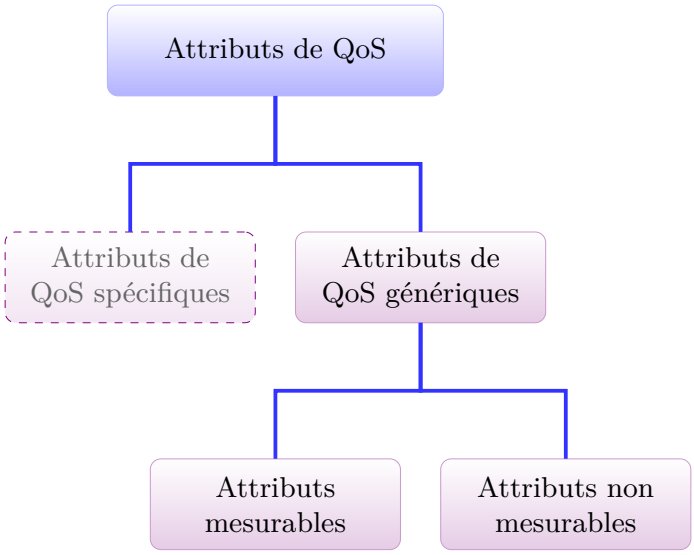


Figure 2.9 – La classification des attributs de QoS Dobson et al. (2005)

Une autre classification différente est donnée par (Araban and Sterling, 2004) où les attributs de QoS sont classés, selon leur mesure par rapport à trois perspectives possibles (fournisseur, usager, observation)

- **Perspective fournisseur**, elle représente tous les attributs QoS dont leurs valeurs sont mises à disposition des usagers par les fournisseurs de services ;
- **Perspective utilisateur**, ce sont les attributs qui peuvent être évalués par les différents utilisateurs pour les services Web ;
- **Perspective des observations** se sont les métriques où ni les fournisseurs de services Web ni les utilisateurs n'interviendront pas pour déterminées leurs valeurs. Généralement ces valeurs peuvent être obtenues après des phases de surveillance ou après des procédures de tests.

Le tableau 2.2 présente un essai de classification de l'ensemble des paramètres définies précédemment basé sur les différentes catégorisations vues en littérature.

Tableau 2.2 – Essai de classification des attributs de QoS

Attributs de QoS	Générique		Perspectives			Unité
	mesurable	non mesurable	fournisseur	usager	observé	
Coût	-	✓	✓	-	-	$
Débit	✓	-	✓	-	-	Ips [9]
Disponibilité	✓	-	-	-	✓	%
Évolutivité	✓	-	✓	-	-	Ms
Fiabilité	✓	-	-	-	✓	%
Latence	✓	-	-	✓	-	Ms [10]
Prix	-	✓	✓	-	-	$
Réputation	-	✓	-	✓	-	-
Robustesse	✓	-	-	-	✓	%
Sécurité	-	✓	✓	-	-	-

9. Invocation par seconde
10. Milliseconde

2.6 Conclusion

Ce chapitre sur la description de la QoS dans les services Web nous a permis de présenter une vue d'ensemble sur les principales spécifications qui supportent le modèle des services Web, tout en relatant leurs technologies et leur fonctionnement.

Au début du chapitre, nous avons présenté le paradigme de l'architecture orientée service. Nous avons discuté par la suite, la notion de service au sein de cette architecture. Le modèle des services Web a été aussi discutée afin d'expliquer comment les services Web s'intègrent dans une logique architecturale orientée service. Ensuite, les différents composants participant à une invocation de service Web ainsi que les différents normes et standards, basés sur les technologies XML, qui sont utilisés aussi pour la mise en œuvre de cette invocation, ont été abordés.

Avec la prolifération des services web, la notion de QoS émerge et prend de plus en plus une grande importance que ce soit pour les fournisseurs ou les utilisateurs de services. C'est pour ce contexte là que la fin du chapitre a été consacrée ; où nous avons introduit la notion de QoS, tout en détaillant ces différents attributs ainsi que ces différentes classifications présentées en littérature.

Les attributs de QoS représentent une indication importante, en termes d'exigences, du niveau du service auquel l'utilisateur peut s'attendre, et en même temps, elles prennent un rôle primordial pour son choix du service. C'est dans cette première perspective que s'intègre notre travail de recherche, où la prédiction des valeurs de la QoS va considérablement améliorée l'expérience utilisateur dans le choix de ses services Web.

Notre deuxième perspective pour cette thèse est, par conséquence, de discuter le moyen théorique utilisé afin de réaliser les prédictions des valeurs de la QoS. Ce moyen fera l'objet du chapitre suivant.

Chapitre 3

Le filtrage collaboratif noyau de la prédiction

Sommaire

3.1 Introduction 23
3.2 Les systèmes de recommandation 24
3.2.1 Origines des systèmes de recommandation 24
3.2.2 Définition de base 25
3.2.3 Notations 25
3.3 Filtrage collaboratif 26
3.3.1 Classification des approches basées sur le FC 26
3.3.2 Filtrage collaboratif basé sur la mémoire 27
3.3.3 Filtrage collaboratif basé sur un modèle 35
3.3.4 Les modèles à facteurs latents 35
3.3.5 Les modèles probabilistes 37
3.4 Prédiction versus Recommandation 40
3.5 évaluation des techniques du FC 41
3.6 Conclusion 42

3.1 Introduction

L'émergence d'Internet durant ces dernières années, grâce notamment à la prolifération des médias sociaux tels que les blogs, sites d'information, plateformes des réseaux sociaux (Facebook, Twiter et autres), commerce électronique, etc., a engendré une grande quantité de données, de produits mais aussi de services. Cette croissance exponentielle d'informations disponibles a causé, aux différents usagers du Web, une difficulté à trouver un contenu pertinent et adéquat selon leurs préférences personnelles et leurs intérêts divers. Le problème n'est donc plus de disposer de l'information mais plutôt de comment trouver une information qui soit la plus pertinente possible. C'est pour ces raisons que les systèmes de recommandation (Resnick and Varian, 1997) ont vu le jour.

Dans le monde numérique actuel, les systèmes de recommandation sont considérés parmi les outils d'information les plus puissants. Ils permettent de générer des recommandations, des suggestions et des prédictions aux différents utilisateurs. Ils sont utilisés dans plusieurs domaines où les internautes ont besoin d'avoir des suggestions afin de les aider à choisir parmi de nombreux objets différents (produits, services, amis, etc.).

Dans la littérature, plusieurs méthodes ont été développées et mises en place pour les systèmes de recommandation. Selon (Bhatnagar, 2017), et en fonction des ressources et des données employées pour faire de la recommandation, ces approches peuvent être classées en différentes catégories, à savoir : la « recommandation basée sur le contenu », le « filtrage collaboratif », la « recommandation basée sur la connaissance », la « recommandation démographique », et enfin les méthodes dites « hybrides ». Une classification globale des cinq principaux types de systèmes de recommandation est présentée dans la

Figure 3.1. Cependant, dans le cadre de cette thèse, nous nous sommes intéressés qu'aux techniques de filtrage collaboratif, les autres ne font pas l'objet d'étude dans nos travaux.

Dans cette partie de thèse, nous présentons une brève introduction sur les systèmes de recommandation. Nous enchaînons, par la suite, avec le filtrage collaboratif en abordons les différentes méthodes et techniques développées pour cette catégorie de systèmes de recommandation. La fin du chapitre est consacrée à la présentation de la notion des mesures d'évaluation des performances utilisés dans les systèmes de recommandation.

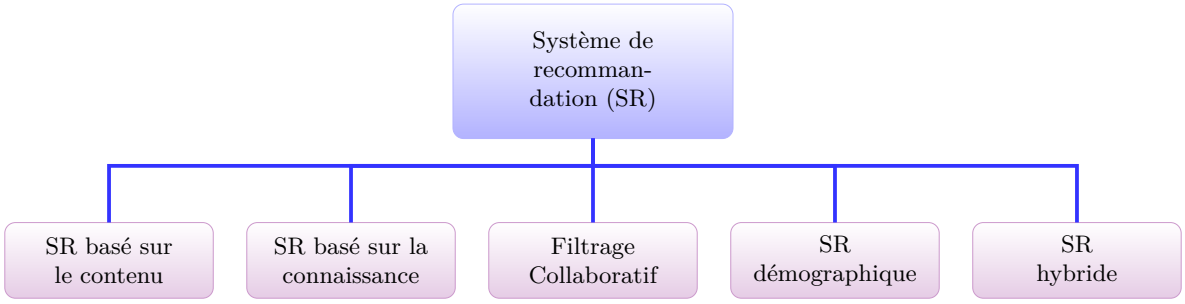


Figure 3.1 – Classification globale des systèmes de recommandation (Bhatnagar, 2017)

3.2 Les systèmes de recommandation

Dans le monde d'aujourd'hui, les systèmes de recommandation sont omniprésents sur Internet. Ils sont souvent utilisés par de grandes compagnies. Par exemple, pour chaque utilisateur de YouTube [11], Facebook [12], Amazon [13], etc., une liste de recommandation lui est automatiquement créée et proposée. La création de cette liste a été possible grâce à l'historique d'utilisation disponible sur les utilisateurs (recherches précédentes, commentaires, réactions, etc.) .

L'objectif d'un système de recommandation est de présenter à l'utilisateur des éléments d'information ou des objets (livres, films, services Web, etc.) considérés comme pertinents selon ses préférences. Il permet aux utilisateurs de surmonter une surcharge d'information en réduisant de manière considérable leur temps de recherche tout en augmentant leur efficacité à trouver les objets qui sont susceptibles de les intéresser.

3.2.1 Origines des systèmes de recommandation

Les origines des systèmes de recommandation découlent des travaux étendus dans de distincts domaines, tels que les sciences cognitives, la théorie d'approximation, la recherche d'information, la recherche documentaire et aussi des sciences de gestion et de marketing (Adomavicius et al., 2005). Ainsi, le système bibliothécaire GRUNDY de (Rich, 1979) est considéré comme étant la première tentative vers la construction de modèles informatiques représentants le choix de l'utilisateur. Le principe de ce système était assez primitif. Il préconisait l'utilisation de profils utilisateurs « stéréotypes ». Par la suite, il affectait chaque lecteur au profil stéréotypé adéquat à partir d'un ensemble d'information limité sur chacun d'eux (en se basant seulement sur une courte interview). Le système produisait ensuite des recommandations à partir de ces stéréotypes.

L'émergence, à partir des années 1990, des transactions électroniques et commerciales en ligne (via Amazon, Netflix [14], etc.), est considérée comme étant un véritable vecteur de développement des systèmes de recommandation. Depuis, les travaux sur les approches et techniques de recommandation ont connu un véritable essor et deviennent un sujet d'intérêt croissants dans le domaine informatique.

11. youtube.com
12. facebook.com
13. amazon.com
14. netflix.com

3.2.2 Définition de base

Dans la littérature, les systèmes de recommandation sont définis de selon diverses façons. Nous citons ici la définition que propose *Robin Burke* (Burke, 2002) : « *The term now has a broader connotation, describing any system that produces individualized recommendations as output or has the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options* » ; que nous la traduisons comme suit:« *Actuellement, le terme a une connotation plus large, décrivant tout système capable de fournir des recommandations individualisées ou permettant une orientation personnelle de l'utilisateur vers des objets intéressants ou utiles au sein d'un large espace de choix possibles* ».

Une deuxième définition a été proposée par (Ricci et al., 2011) : « *Recommender Systems are software tools and techniques providing suggestions for items to be of use to a user* ». Que nous pouvons la traduire comme suit : « *Les Systèmes de recommandation sont des outils logiciels et des techniques qui fournissent des suggestions d'items qui soient utiles pour un utilisateur* ».

De ces deux définitions, nous pouvons en déduire que la tâche principale pour un système de recommandation est de fournir à l'utilisateur ou un groupe d'utilisateurs un choix plus restreint d'objets pertinents par rapport à leurs préférences et centres d'intérêt.

A la base, un système de recommandation se compose exhaustivement de trois éléments principaux : d'une liste d'*utilisateurs*, une liste d'*items* et un ensemble de scores (notes, ou aussi évaluations) qui peut être compacté dans une structure matricielle appelée *matrice de scores*. Dans cette dernière, les lignes représentent les utilisateurs, tandis que les colonnes sont les items. À noter que les scores représentent le degré de préférence attribué par les utilisateurs aux différents items.

La mise en œuvre d'un tel système de recommandation peut se faire de deux façons :

1. A partir des valeurs déjà présentes dans une matrice d'utilité (matrice des scores), la première approche consiste à évaluer tous les scores manquants pour cette matrice. En d'autres termes, le système doit prédire le score probable qu'un utilisateur pouvait attribuer à chaque item.
2. La deuxième approche consiste à la création d'une liste ordonnée de k meilleures recommandations. Cette liste peut être créer soit à partir de mesures de similarité entre les différents utilisateurs et/ou items ; soit, et c'est souvent le cas, à partir de la première approche sans que la liste des Top-k recommandations ne soit pas nécessairement la liste des k items avec les plus hautes valeurs de prédiction.

3.2.3 Notations

Un système de recommandation s'adresse principalement à deux entités : les utilisateurs et les items. Formellement, un problème de recommandation peut être décrit comme suit (Adomavicius and Tuzhilin, 2005) :

$$\forall u \in \mathcal{U}, i_u^* = \underset{i \in \mathcal{I}}{\operatorname{argmax}}\ f(u, i) \tag{3.2.1}$$

, où $\mathcal{U}$ représente l'ensemble de tous les utilisateurs qui utilisent le système de recommandation ; $\mathcal{I}$ l'ensemble de tous les items qui peuvent être recommandées ; f Une fonction d'utilité qui mesure et estime le degré de préférence ou de pertinence d'un item $i \in \mathcal{I}$ pour un utilisateur $u \in \mathcal{U}$:

$$f : \mathcal{U} \times \mathcal{I} \longrightarrow \mathcal{R}$$

et $\mathcal{R}$ est la matrice d'utilité ou l'espace d'évaluation, qui est souvent un sous ensemble ordonné d'entiers ou de réels non négatifs ($\mathcal{R} \in \mathbb{R}^{u \times i}$).

Une fois que la fonction d'utilité f est définie, l'objectif de la recommandation consiste alors à prédire l'évaluation d'un item spécifique i pour un utilisateur u en maximisant la fonction f pour l'utilisateur u sur l'item i, cela pour une éventuelle recommandation.

Depuis, diverses méthodes et approches ont été développées pour trouver des solutions efficaces à l'équation 3.2.1. Parmi toutes ces méthodes, qui sont proposées en littérature, nous nous sommes intéressés à la catégorie de filtrage collaboratif, et qui sera abordé plus en détail dans les sections suivantes.

3.3 Filtrage collaboratif

Le filtrage collaboratif est une technique émergente qui a été utilisée dans les systèmes de recommandation. Le principe du concept s'est inspiré du monde réel. En effet, le comportement de certains utilisateurs est hautement considéré par d'autres utilisateurs. Souvent, les avis, choix et/ou décisions des internautes sont influencés par l'environnement qui les entourent, et qui peut être exprimé par des sondages, des dépêches d'information, de l'opinion publique, ou même par le biais du bouche-à-oreille. De cette constatation, le filtrage collaboratif consiste en une collaboration entre plusieurs personnes afin qu'elles s'entraident pour effectuer des choix, tout en essayant de modéliser leurs réactions diverses.

La littérature dans le domaine des systèmes de recommandation attribue à (Goldberg et al., 1992) la première utilisation du terme de « *Filtrage collaboratif* » (*Collaborative Filtering*). L'idée de base du filtrage collaboratif consiste à un partage d'opinions entre différents utilisateurs ayant des centres d'intérêts similaires. L'hypothèse fondamentale de cette méthode suppose que si un groupe d'utilisateurs a les mêmes préférences à l'égard d'un groupe d'items, rien n'empêche qu'ils auront les mêmes préférences par rapport à de nouveaux items.

Ainsi, une définition plus ou moins exhaustive du concept est donnée par (Bhatnagar, 2017), et qui stipule que : « *Collaborative filtering is defined as a technique that filters the information sought by the user and patterns by collaborating multiple data sets, such as viewpoints, multiple agents and pre-existing data about the users' behavior stored in matrices* ». Cette définition est traduite comme suit : « *Le filtrage collaboratif est défini comme une technique permettant de filtrer les informations recherchées par des utilisateur ou par profil d'utilisateur en faisant collaborer plusieurs données existantes, relatives au comportement des utilisateurs, qui sont mises dans des matrices* ».

Historiquement, les premières tentatives d'utilisation du filtrage collaboratif dans le domaine des systèmes de recommandation étaient avec les travaux de (Resnick et al., 1994) et (Hill et al., 1995). Le premier travail a utilisé, pour la première fois, la technique de FC afin de recommander les articles du réseau Usenet [15] susceptibles d'être intéressants pour un utilisateur donné en collectant les notes données par les utilisateurs lorsqu'ils lisent des articles. Le deuxième travail concerne le développement d'un système de recommandation multimédias.

3.3.1 Classification des approches basées sur le FC

Comme précisé dans (Deshpande and Karypis, 2004), le filtrage collaboratif permet de recommander un item à un utilisateur en se basant sur les opinions de personnes ayant des centres d'intérêts communs. Depuis, les différentes méthodes développées autour de ce terme, sont scindées en trois sous-familles principales (Su and Khoshgoftaar, 2009), et qui sont illustrées par la Figure 3.2 :

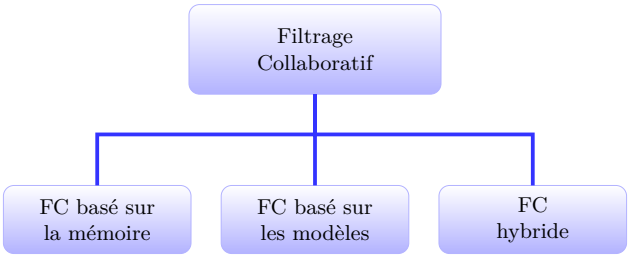


Figure 3.2 – Classification globale des techniques de filtrage collaboratif [16]

Famille de méthodes basées sur la mémoire (ou *Memory-Based*) (Herlocker et al., 1999). Ce type d'algorithmes est également connu sous le nom d'algorithmes *basés sur les voisins* (Ning et al., 2015) (ou *Neighborhood-Based*). Cette catégorie de méthodes exploite la totalité des données d'évaluation des utilisateurs, sur l'ensemble des items, contenues dans la matrice d'évaluations. L'objectif est

15. Wikipédia : Usenet est un système en réseau de forums, inventé en 1979, lancé publiquement en 1980 à l'Université de Caroline du Nord à Chapel Hill and Duke University. Page Consultée le 10/6/2021.
16. Classification adaptée à partir de Su and Khoshgoftaar (2009)

d'identifier la similarité qui peut exister entre les différents utilisateurs ainsi que la similarité qui peut exister aussi entres les différents items. Cette approche de filtrage collaboratif est parmi les approches les plus anciennes et les plus efficaces.
La mise en œuvre pour ce type d'approches, peut être effectuée par rapport aux deux dimensions de la matrice d'évaluation, à savoir l'utilisateur et l'item. En effet, les approches qui ont pour objectif d'identifier la similarité entre utilisateurs sont connues sous le nom d'approches basées sur les utilisateurs (ou *User-Based*) (Ricci et al., 2011). Dans le cas contraire, il y a les approches basées sur les items (ou *Item-Based*) (Deshpande and Karypis, 2004).

Famille de méthodes basées sur les modèles (ou *Model-Based*) (Canny, 2002). Ces méthodes ont pour objectif principale de fournir un modèle qui vaut une image réduite représentative de la matrice des évaluations initiale. Contrairement à la famille de filtrage collaboratif précédente, où les évaluations des utilisateurs sont directement impliquées dans le processus de prédiction ; les approches basées sur les modèles, utilisent plutôt ces évaluations pour apprendre des modèles prédictifs. Ces derniers sont souvent développés soit à l'aide d'algorithmes d'apprentissage automatique, soit via l'utilisation des techniques de fouille de données. Pour ce faire, plusieurs méthodes ont été proposées dans la littérature, comprenant divers modèles, tels que le modèle des facteurs latents, la factorisation matricielle, les réseaux bayésiens, les modèles probabiliste, etc. Ces différentes approches seront détaillées dans les sections ultérieures de ce chapitre.

Famille de méthodes hybrides (Bobadilla et al., 2020 ; Pennock et al., 2013). La technique de filtrage collaboratif hybride comprend tous les algorithmes qui combinent à la fois les approches de filtrage collaboratif basés sur la mémoire et ceux basées sur les modèles. L'objectif derrière cette concaténation est de trouver une solution afin de surmonter les problèmes courants rencontrés dans les deux techniques de filtrage collaboratif précédentes (la rareté des données, la perte d'information, etc.).

3.3.2 Filtrage collaboratif basé sur la mémoire

Le filtrage collaboratif basé sur la mémoire, connu aussi sous le nom de filtrage collaboratif basé sur les voisins, utilise généralement l'historique des interactions des différents utilisateurs pour faire de la recommandation. L'objectif est de prédire le taux d'utilité d'un item pour un utilisateur particulier en utilisant les évaluations des utilisateurs déjà mémorisés par le système. Cette famille de filtrage collaboratif peut être classée en deux catégories différentes connues sous le terme d'algorithme basé sur les utilisateurs ou algorithme basé sur les items, comme l'illustre la Figure 3.3.

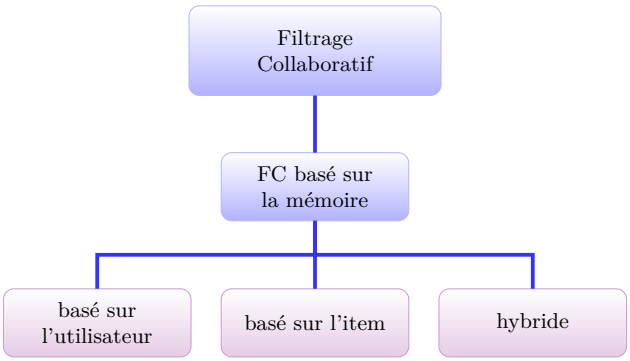


Figure 3.3 – Classification des techniques de filtrage collaboratif basées sur la mémoire

Dans cette catégorie de FC, la fonction d'utilité f de l'équation 3.2.1 est représentée par une fonction de calcul de similarité. Par conséquent, et en plus des notations présentées en section 3.2, nous distinguons deux notations afin d'exprimer ces deux métriques de mesure de similarité :

— $sim(u, v)$ est la fonction qui permet de mesurer la similarité entre deux utilisateurs u et v ;

— $sim(i, j)$ est la fonction qui calcule le taux de similarité entre deux items différents i et j.

Nous rappelons que ces deux métriques sont calculées à la base des différentes données stockées dans la matrice des évaluations $\mathcal{R} \in \mathbb{R}^{u \times i}$.

Cette catégorie d'approches de filtrage collaboratif présente plusieurs avantages, que nous pouvons les résumer comme suit (Ning et al., 2015) :

— **Simplicité** : ces méthodes sont intuitives et relativement simples à mettre en œuvre. Par exemple, dans leur forme la moins complexe, un seul paramètre (le nombre de voisins utilisés dans la prédiction) nécessite un ajustement ;

— **Justifiabilité** : de telles méthodes fournissent également une justification concise et intuitive des résultats obtenus. Cela peut aider l'utilisateur à mieux comprendre le fonctionnement de la recommandation et sa pertinence ;

— **Efficacité** : l'un des points forts de ses systèmes est leur efficacité. En effet, ce type d'approches ne nécessite pas de phases d'apprentissage coûteuses par rapport à d'autres catégories. Aussi, ce type de méthodes fournissent une recommandation quasi instantanée. En effet, et bien que la phase de calcul des plus proches voisins soit généralement plus coûteuse en termes de temps ; en revanche, elle peut se faire dans une étape en hors ligne. De plus, le stockage des résultats de traitement (les listes des plus proches voisins) nécessite très peu d'espace mémoire, ce qui les rend plus efficaces pour les applications ayant un nombre important d'utilisateurs et/ou d'items.

— **Stabilité** : ces modèles sont peu affectés par l'ajout constant d'utilisateurs, d'items et d'évaluations. Par exemple, une fois que les similitudes entre items ont été calculées, la recommandation aux nouveaux utilisateurs peut se faire sans avoir à recycler tout le système.

Avant de présenter en détails les différentes techniques de filtrage collaboratif basée sur le voisinage, nous introduisons dans la partie suivante, la notion de base qui est en étroite liaison avec ce type de filtrage collaboratif, à savoir, la notion de *mesure de similarité* en présentant quelques métriques parmi les plus utilisées dans ce contexte.

Mesures de similarité

Selon (McGill et al., 1979), il existe une soixantaine de mesures de similarité différentes. Ces mesures sont classées par rapport à leur objectif d'utilisation. Dans certain cas on s'intéresse à mesurer la similarité en fonction de la proportion dans laquelle deux objets possèdent les mêmes caractéristiques. Selon une autre approche, la similarité devrait être mesurée en fonction de la façon dont les objets partagent les mêmes objectifs. Pour le domaine des systèmes de recommandation en général, et le filtrage collaboratif en particulier, la mesure de similarité est une notion primordiale. Il s'agit d'une métrique utilisée pour comparer deux objets afin de déterminer leur degré de similitude. Dans le filtrage collaboratif, la similarité consiste à comparer qui peut exister entre soit les utilisateurs soit les items.

Dans ce contexte est parmi les différentes mesures de similarité développées en littérature, nous avons considéré deux mesures de similarité ; à savoir, la *corrélation de Pearson* (O'Mahony et al., 2005 ; Resnick et al., 1994) et la *similarité du cosinus* (Bao et al., 2013 ; Breese et al., 1998). Ces deux métriques sont considérées parmi les plus utilisées dans le contexte de filtrage collaboratif

Corrélation de Pearson Selon (Schafer et al., 2007), La métrique de corrélation des coefficients de Pearson (ou *Pearson correlation coefficient(PCC)*) est considérée comme la métrique la plus utilisée et la plus performante dans le domaine des systèmes de recommandation. C'est une méthode issue des statistiques permettant de mesurer la corrélation linéaire entre deux variables.

Cette métrique peut prendre diverses formes par rapport à plusieurs domaines dans lesquels est utilisée. Cette métrique a été exploitée pour calculer la similarité entre deux variables. Cependant, elle a été généralisée pour calculer la similarité entre deux vecteurs. Ainsi, la (PCC) est définie par l'équation 3.3.1 suivante (Benesty et al., 2009) :

$$\rho(X, Y) = \frac{COV(X, Y)}{\sigma_X \times \sigma_Y} \tag{3.3.1}$$

, où X et Y sont deux échantillons de variables ; σ_X et σ_Y désignent leurs écarts types représentant la mesure de la dispersion des valeurs des échantillons X et Y et $COV(X,Y)$ est la covariance qui qualifie à la fois le sens et l'intensité de liaison entre les deux variables X et Y. Elle correspond à l'espérance du produit des variables centrées :

$$COV(X,Y) = E\{[X - E(X)]\,[Y - E(Y)]\}$$

; où $E(X)$ et $E(Y)$ correspondent à la une moyenne pondérée des valeurs que peut prendre respectivement X et Y.

Similarité du cosinus Parmi les mesures de similarité les plus utilisées dans le domaine du filtrage collaboratif nous avons le calcul de similarité de cosinus (Huang, 2008). Elle est souvent connue sous le nom de similarité vectorielle.

Cette métrique consiste à déterminer la similarité du cosinus vectorielle entre deux vecteurs A et B, elle est exprimée comme suit (équation 3.3.2) :

$$cos(A,B) = \frac{A.B}{\|A\|.\|B\|} \tag{3.3.2}$$

À noter que pour leurs valeurs des deux mesures de similarité que nous avons présentées, varient sur l'intervalle $[-1,1]$. Ce qui est interprété comme suit : plus la similarité se rapproche de la valeur 1, plus les deux vecteurs sont similaires ; plus elle est proche de 0, moins les deux vecteurs sont similaires. Les vecteurs sont opposés dans le cas où la similarité se rapproche de -1.

Pour d'autres mesures de similarité utilisées dans le contexte de filtrage collaboratif, nous orientons le lecteur vers l'article (Khojamli and Razmara, 2021) qui dresse une synthèse assez récente et complète sur les différentes métriques de similarité définies en littérature,

Filtrage collaboratif basé sur les utilisateurs

Les approches basées sur le voisinage utilisateur partent du principe que si un item est apprécié par un utilisateur donné, ce même item a de fortes chances d'être apprécier et privilégier par d'autre utilisateurs voisins. L'ensemble d'utilisateurs voisins est composé des différents utilisateurs qui ont des préférences communes, et par conséquence les mêmes habitudes d'évaluation. Il est souvent considéré que la meilleure manière de déterminer et connaître, globalement, les préférences d'un utilisateur est de les déterminer à travers celles de ses utilisateurs voisins.
Un exemple de matrice d'évaluation est illustré dans la Figure 3.4. Les lignes, pour cette matrice, représentes les différents utilisateurs et les colonnes font références aux items (livres, articles, services, etc.) présents dans le système. Chaque cellule $r_{u,i}$ contient une évaluation de l'utilisateur u sur l'item i.

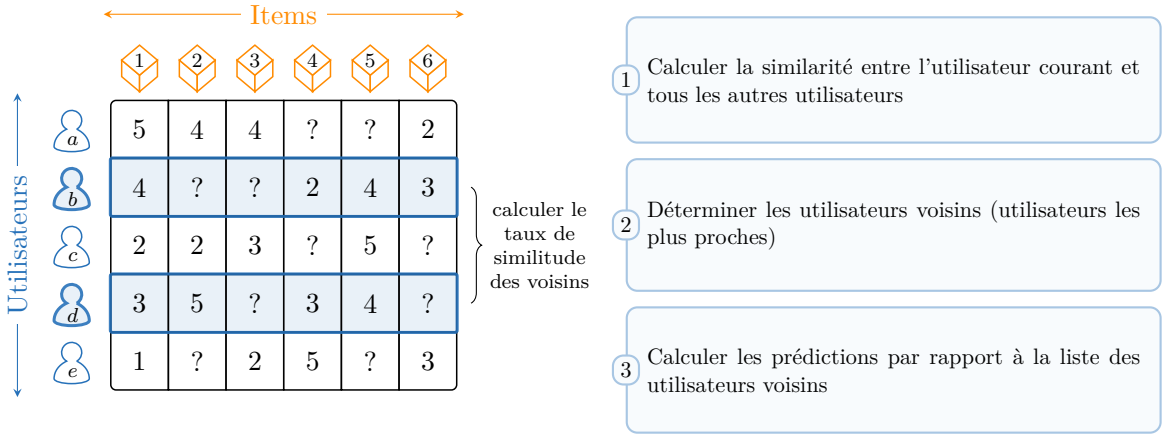


Figure 3.4 – Filtrage collaboratif basé sur les utilisateurs

Dans cette approche, le processus de recommandation d'un item i pour un utilisateur u s'effectue en trois phases consécutives, comme l'illustre la Figure 3.4:

1. La première phase consiste à déterminer le degré de similarité entre l'utilisateur courant u avec tous les autres utilisateurs présents dans le système, et qui ont déjà évalué l'item candidat i;
2. La deuxième phase consiste à déterminer les voisins de l'utilisateur u, en se basent sur les résultats de calculs de la phase précédente;
3. Et en fin, la dernière phase consiste à prédire l'évaluation que donnera l'utilisateur u pour l'item i. Cette étape s'effectue en combinant les scores des évaluations des utilisateurs voisins (c'est-à-dire calculer la moyenne pondérée des différents scores de l'item i).

.

Calcul de la similarité Les mesures de similarité sont principalement utilisées pour décrire une relation linéaire positive ou négative entre utilisateurs. Leurs valeurs sont comprises dans l'intervalle $[-1, 1]$. Plus cette valeur est proche de 1 (en valeur absolue), plus la relation est forte. la valeur nulle indique l'absence de corrélation.

1. ***Corrélation de Pearson***: Pour la mise en œuvre de cette métrique, l'équation 3.3.1 de corrélation de Pearson définie précédemment sera adaptée par rapport au contexte du filtrage collaboratif, afin de calculer le taux de similarité entre deux utilisateurs u et v. En littérature cette modification est connue sous le nom coefficient de corrélation empirique. Ainsi, la similarité est donnée par l'équation 3.3.3 suivante:

$$sim_{pcc}(u,v) = \frac{\sum\limits_{i \in \mathcal{I}_{uv}} (r_{u,i} - \bar{r}_u) \times (r_{v,i} - \bar{r}_v)}{\sqrt{\sum\limits_{i \in \mathcal{I}_{uv}} (r_{u,i} - \bar{r}_u)^2} \times \sqrt{\sum\limits_{i \in \mathcal{I}_{uv}} (r_{v,i} - \bar{r}_v)^2}} \tag{3.3.3}$$

   Avec $\mathcal{I}_{uv}$ est l'ensemble d'items co-évalué par les deux utilisateurs u et v, $r_{u,i}$ et $r_{v,i}$ représentent l'évaluation donné à l'item i par respectivement les utilisateurs u et v, $\bar{r}_u$ et $\bar{r}_v$ sont les moyennes des évaluations données par un utilisateur u respectivement v sur l'ensemble des items qu'ils ont évalués ($\mathcal{I}_u$), où:

$$\bar{r}_u = \frac{\sum\limits_{i \in \mathcal{I}_u} r_{u,i}}{|\mathcal{I}_u|}$$

   et

$$\bar{r}_v = \frac{\sum\limits_{v \in \mathcal{U}_i} r_{u,i}}{|\mathcal{I}_v|}$$

   Cependant, il est à noter que la corrélation de Pearson est insensible à la taille des items évalués; c'est-à-dire, plus le nombre d'items partagés est grand moins la corrélation soit bonne. En d'autres termes, si deux utilisateurs ont un nombre faible d'items en commun mais avec des évaluations identiques ou très semblables; leur corrélation de Pearson sera plus grande par rapport à un autre couple d'utilisateurs qui a un nombre important d'items en commun mais avec des scores un peu moins identiques. Pour pallier aux problèmes relatifs à la corrélation de Pearson, d'autres types de métriques peuvent être utilisées dans les systèmes de recommandation (Breese et al., 1998).

2. ***Similarité du cosinus***: Cette métrique considère les lignes de la matrice des scores comme un ensemble de vecteurs d'évaluation. En effet, dans cette méthode les utilisateurs (u et v) sont représentés par les vecteurs de leurs éléments ($\vec{u}$ et $\vec{v}$) dans un espace de $|\mathcal{I}_{uv}|$ dimensions (c'est-à-dire le nombre d'items évalués par les deux utilisateurs). La similarité est calculée à la base de tous les items évalués par les deux utilisateurs.

Pour se faire, l'équation 3.3.2 sera empiriquement modifiée par l'équation 3.3.4 suivante:

$$sim_{cos}(u,v) = \frac{\sum_{i \in \mathcal{I}_{uv}} r_{u,i} \times r_{v,i}}{\sqrt{\sum_{i \in \mathcal{I}_{uv}} r_{u,i}^2} \times \sqrt{\sum_{i \in \mathcal{I}_{uv}} r_{v,i}^2}} \tag{3.3.4}$$

Dans le cas où le système de notation ne considère que les valeurs positives, la valeur du cosinus varie dans l'intervalle $[0, 1]$. Dans le cas contraire, elle varie dans l'intervalle $[-1, 1]$; et comme déjà mentionnée, cela est interprété comme suit: -1 signifie préférences opposées, 0 est interprétée comme pas de préférences communes et 1 stipule que les deux utilisateurs ont les mêmes préférences.

Calcul de la prédiction Une fois que le calcul de similarité est effectué entre l'utilisateur actif u et les autres utilisateurs. La seconde étape consiste à la prédiction du taux de satisfaction de u à l'égard d'un item i non encore évalué par ce dernier.

Selon (Schafer et al., 2007), la prédiction de la valeur de satisfaction de l'utilisateur u pour l'item i est calculée à partir de la moyenne pondérée des taux de similarités entre les Top-k voisins de u qui ont déjà évalué l'item i. L'équation 3.3.5 illustre la formule correspondante pour le calcul de la prédiction.

$$pred(u,i) = \hat{r}_{ui} = \frac{\sum_{v \in \text{Top-}k(\mathcal{U}_i)} sim(u,v) \times r_{v,i}}{\sum_{v \in \text{Top-}k(\mathcal{U}_i)} sim(u,v)} \tag{3.3.5}$$

où $sim(u,v)$ est soit la similarité de Pearson (équation 3.3.3) ou du cosinus (équation3.3.4) entre les utilisateurs u et v, ou tout simple n'importe qu'elle autre mesure de similarité, $\bar{r}_v$ est la moyenne des évaluations données par l'utilisateur v, $\mathcal{U}_i$ est l'ensemble des utilisateurs qui ont notés l'item i, Top-$k(\mathcal{U}_i)$ est une liste ordonnée représentant le sous-ensemble des voisins les plus similaires à l'utilisateur u. Pour le calcul de la prédiction, il n'est pas obligatoire de considérer que les Top-k voisins. En effet, si le temps de calculs est optimal, il est préférable de considérer l'ensemble de tous les utilisateurs qui ont évalué l'item candidat i, c'est-à-dire Top-$k(\mathcal{U}_i) = \mathcal{U}_i$.

Dans le filtrage collaboratif, d'autres alternatives issues de la formule précédente peuvent être considérées pour calculer la prédiction. L'équation 3.3.6 est une amélioration de la formule précédente. Cette amélioration part du principe définit dans (Resnick et al., 1994), et qui considère que pour atténuer la variation de jugements qu'existe entre les différents utilisateurs; l'évaluation de chaque utilisateur v doit être ajustée par rapport à la moyenne de ses notes $\bar{r}_v$. De plus, et afin d'avoir une valeur de prédiction normalisée, il est nécessaire que la division se fait sur la somme des valeurs absolues des similarités entre u est ses différents voisins ($v \in \mathcal{U}_i$). La formule est modifiée comme suit (équation 3.3.6):

$$pred(u,i) = \hat{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in \mathcal{U}_i} sim(u,v) \times (r_{v,i} - \bar{r}_v)}{\sum_{v \in \mathcal{U}_i} |sim(u,v)|} \tag{3.3.6}$$

Cette formule peut être effectuée dans une approche de Top-k voisins; où le calcul ne prend pas en considération tous les utilisateurs qui ont évalué l'item i, mais plutôt, il se limite aux k premiers voisins, $v \in$ Top-$k(\mathcal{U}_i)$, considérés comme étant les plus proches de l'utilisateur u.

Filtrage basé sur les items

Alors que la prédiction dans le filtrage collaboratif à base de voisinage utilisateur est centrée sur l'avis des différents utilisateurs voisins, le filtrage collaboratif à base d'item prédit l'évaluation potentielle que donnera l'utilisateur u pour un item donné i à partir de ses propres évaluations pour

les items similaires à i. De ce fait, ce type de méthodes est vue, par rapport au type précédent, comme une transposition de la matrice d'utilité.

Le principe de fonctionnement du filtrage collaboratif à base d'items a été introduit par (Sarwar et al., 2001); il se compose de trois phases consécutives, comme l'illustre la Figure 3.5.

1. étant donné un utilisateur particulier u et un item candidat i, la première phase consiste à calculer la similarité entre l'item courant i par rapport à tous les autres items;
2. Cette étape consiste à la détermination des différents voisins proches de l'item i en calculant sa similarité par rapport aux autres items du systèmes. Ainsi, les items sont considérés similaires s'ils auront les mêmes scores (souvent des scores qui se rapprochent) par plusieurs utilisateurs.
3. La deuxième phase consiste à prédire le score de l'utilisateur u pour l'item i à partir de ses scores attribués aux différents voisins de i.

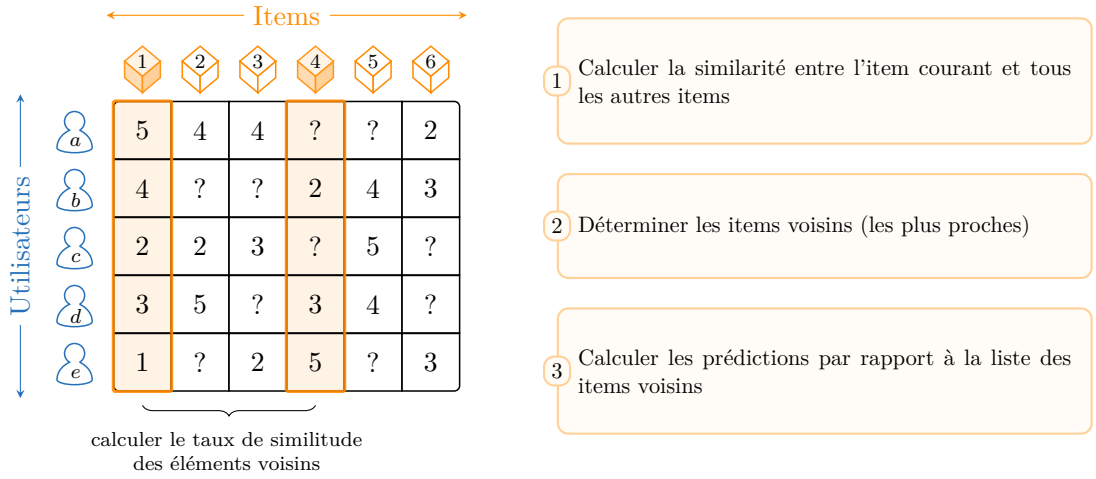


Figure 3.5 – Filtrage collaboratif basé sur les items

Calcul de la similarité L'objectif de la phase du calcul de la similarité est la détermination de l'ensemble des items les plus proches en fonction des différentes évaluations des utilisateurs. Ainsi, deux items sont considérés similaires si plusieurs utilisateurs les ont évalués d'une manière similaire ou proche. Pour le calcul de la similarité entre deux items i et j, nous utiliserons souvent les mêmes métriques utilisées pour le filtrage collaboratif centré utilisateur que nous avons présentées précédemment, à savoir, soit la corrélation de Pearson, soit la similarité vectorielle.

1. ***Corrélation de Pearson***: Comme nous l'avons présenté dans la section précédente, la corrélation de coefficients de Pearson sert à mesurer la variation d'une variable par rapport à une autre. Empiriquement, la similarité entre deux items i et j est égale à la covariance divisée par le produit de l'écart-type des évaluations données par tous les utilisateurs qui ont évalué les deux items en même temps. L'équation 3.3.7 présente la similarité de Pearson ($sim_{pcc}(i,j)$) entre deux items i et j.

$$sim_{pcc}(i,j) = \frac{\sum_{u \in \mathcal{U}_{ij}} (r_{u,i} - \bar{r}_i) \times (r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in \mathcal{U}_{ij}} (r_{u,i} - \bar{r}_i)^2} \times \sqrt{\sum_{u \in \mathcal{U}_{ij}} (r_{u,j} - \bar{r}_j)^2}} \tag{3.3.7}$$

Avec $\mathcal{U}_{ij}$ est l'ensemble des utilisateurs qui ont noté les deux items i et j, $r_{u,i}$ et $r_{u,j}$ représentent les évaluations données par l'utilisateur u à respectivement, les items i et j et enfin, $\bar{r}_i$ et $\bar{r}_j$ sont les moyennes des évaluations reçues par les items i et j, où:

$$\bar{r}_i = \frac{\sum_{u \in \mathcal{U}_i} r_{u,i}}{|\mathcal{U}_i|}$$

et

$$\bar{r}_j = \frac{\sum_{u \in \mathcal{U}_j} r_{u,j}}{|\mathcal{U}_j|}$$

2. ***Corrélation Cosinus Ajustée***: Selon (Schafer et al., 2007), les algorithmes de filtrage collaboratif utilisent une variation plus efficace et populaire de la formule précédente (équation 3.3.7). En effet, les travaux de (Lousame and Sánchez, 2009) ont démontré qu'il est plus intéressant de centrer les évaluations par rapport à la moyenne des notes des utilisateurs (c'est-à-dire $\bar{r}_u$) plutôt que par rapport à la moyenne des notes des items (c'est-à-dire $\bar{r}_i$) . Cette variation est connue sous le nom de la *similarité du Cosinus ajusté*. Empiriquement, le calcul de la similarité du cosinus ajustée entre deux items ($sim_{ajusté}(i,j)$) est donné par l'équation 3.3.8.

$$sim_{ajusté}(i,j) = \frac{\sum_{u \in \mathcal{U}_{ij}} (r_{u,i} - \bar{r}_u) \times (r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \in \mathcal{U}_{ij}} (r_{u,i} - \bar{r}_u)^2} \times \sqrt{\sum_{u \in \mathcal{U}_{ij}} (r_{u,j} - \bar{r}_v)^2}} \tag{3.3.8}$$

3. ***Similarité du cosinus***: Pour cette mesure, les items (i et j) sont représentés par les vecteurs de leurs éléments ($\vec{i}$ et $\vec{j}$) dans un espace de $|\mathcal{U}_{ij}|$ dimensions ($\mathcal{U}_{ij}$ est l'ensemble des utilisateurs qui ont évalué les deux items en même temps). La similarité du cosinus permet de calculer la similarité entre les deux vecteurs $\vec{i}$ et $\vec{j}$ en déterminant le cosinus de l'angle entre eux. Afin de calculer la similarité entre deux items i et j, il suffit de remplacer les utilisateurs dans l'équation 3.3.4 par les items. L'équation 3.3.9 s'est alors comme suit:

$$sim_c(i,j) = \frac{\sum_{u \in \mathcal{U}_{ij}} r_{u,i} \times r_{u,j}}{\sqrt{\sum_{u \in \mathcal{U}_{ij}} r_{u,i}^2} \times \sqrt{\sum_{u \in \mathcal{U}_{ij}} r_{u,j}^2}} \tag{3.3.9}$$

Calcul de la prédiction Une fois que la phase de calcul de la similarité entre items soit achevée; la phase suivante consiste à prédire pour l'utilisateur cible u une valeur pour l'item i candidat à la recommandation (non encore évalué). Selon (Sarwar et al., 2001) la valeur prévue pour la prédiction est la somme pondérée des évaluations pour l'item i divisée par la somme des similarités. Ainsi l'équation de prédiction est écrite comme suit (équation 3.3.10):

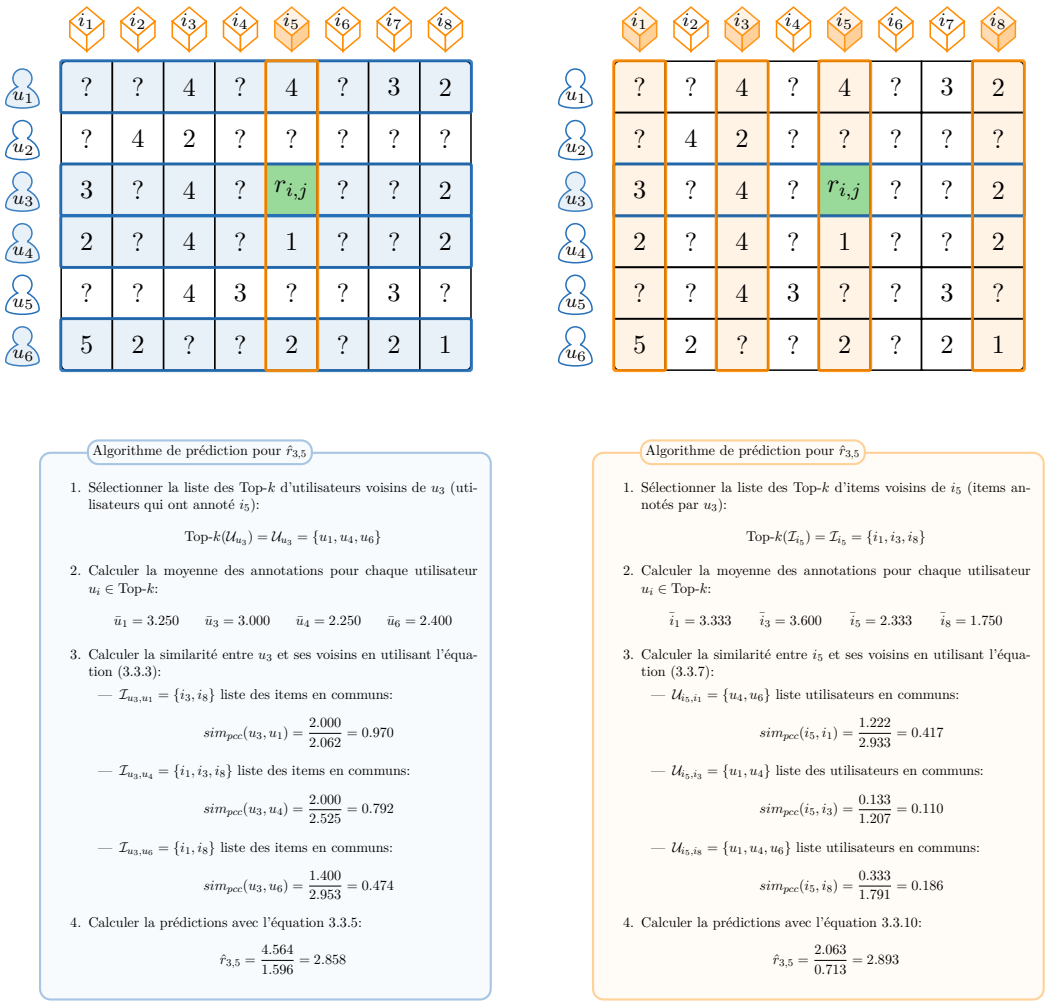
$$pred(u,i) = \hat{r}_{ui} = \frac{\sum_{j \in \mathcal{I}_u} sim(i,j) \times r_{u,j}}{\sum_{j \in \mathcal{I}_u} |sim(i,j)|} \tag{3.3.10}$$

Il est à noter que dans ce cas, l'ajustement effectué pour la prédiction lors d'un filtrage collaboratif à base d'utilisateur (voir équation 3.3.6) n'est pas nécessaire car il s'agit du même utilisateur.

Exemple illustratif détaillé

La Figure 3.6 présente un exemple détaillé sur le FC basé sur la mémoire. Dans cet exemple, la matrice d'évaluation est composée d'un ensemble d'utilisateurs $\mathcal{U} = \{u_1, \ldots, u_6\}$, et d'un ensemble d'items $\mathcal{I} = \{i_1, \ldots, i_8\}$. L'objectif est de prédire l'évaluation que donnera l'utilisateur u_3 concernant l'item i_5 en calculant la valeur $r_{3,5}$; cela par l'application des deux approches détaillées dans cette section.

En ce qui concerne la technique de filtrage collaboratif basé sur les utilisateurs, illustrée par la Figure 3.6a, nous avons procédé comme suit:



(a) FC basé sur les utilisateurs **(b)** FC basé sur les items

Figure 3.6 – Exemple illustratif

1. Sélectionner un sous ensemble d'utilisateurs voisins qui seront considérés comme recommandeurs. Il s'agit des utilisateurs qui ont déjà recommandé l'item i_5;
2. Calculer la similarité entre l'utilisateur courant i_3 et les utilisateurs voisins;
3. Calculer la prédiction en utilisant une combinaison pondérée des évaluations des voisins sélectionnés.

À noter que pour l'étape 2, nous avons appliqué la mesure de la corrélation des coefficients de Pearson (équation 3.3.3). Cette mesure est insensible à la taille de la liste des items évalués. En effet, plus le nombre d'items est grand moins la mesure de similarité soit bonne; l'inverse est vrai. Par exemple, bien que u_4 soit clairement plus semblable à u_3, leur coefficient de corrélation vaut 0.792 (sur trois items en commun), alors que celui entre u_3 et u_1 est de 0.970 (sur deux items seulement). En revanche, si nous avons opté pour la mesure du cosinus (équation 3.3.4) la similarité entre u_3 et u_1 vaut 0.554, alors qu'entre u_3 et u_4 vaut 0.966.

Par rapport au filtrage collaboratif basé sur les items, présentée dans Figure 3.6a, nous avons appliqué les 3trois étapes suivantes:

1. Sélectionner un sous ensemble d'item voisins. Il s'agit de l'ensemble des items déjà évalué par l'utilisateur u_3;
2. Calculer la similarité entre l'item courant et tous les items voisins;
3. Déterminer la valeur prédite en utilisant un algorithme basé sur l'évaluation par l'utilisateur u_3 des items appartenant au voisinage de l'item i_5.

Les résultats des différentes méthodes de filtrage collaboratif à base de voisinage, à savoir utilisateurs ou items, sont présentés par les figures 3.6a et 3.6b.

Filtrage hybride

Un filtrage collaboratif basé sur le voisinage est considéré comme hybride s'il combine les deux approches de filtrage collaboratif basé sur la mémoire, à savoir, l'approche basée sur l'utilisateur et celle basée l'item. Souvent cette hybridation est connue sous le nom d'*hybridation monolithique* (Abbasi et al., 2014) qui consiste à utiliser différentes sources et approches pour réaliser la recommandation. Cependant il existe d'autres types d'hybridation à savoir l'hybridation parallélisée et en cascade (Abbasi et al., 2014). Le premier type consiste en une combinaison des résultats de recommandation de plusieurs algorithmes afin de générer le résultat final de la recommandation. Cependant, le second type consiste en une utilisation en pipeline de plusieurs systèmes de recommandation où les résultats d'un système représentent l'entrée pour le système suivant.

3.3.3 Filtrage collaboratif basé sur un modèle

Le filtrage collaboratif basé sur le voisinage est considéré comme la technologie la plus probante et la plus utilisée dans la construction des systèmes de recommandation. Cependant, ces méthodes sont traditionnellement confrontées à plusieurs problèmes majeurs, à savoir, l'évolutivité (*scalability problem*) (Eirinaki et al., 2018), la rareté des données (*sparsity problem*) (Pan et al., 2010) et le problème du démarrage à froid (*cold-strat problem*) (Schein et al., 2002).

En effet, l'évolutivité du système de recommandation engendre une très grande complexité de calculs pour les différentes métriques de similarités, ainsi que des prédictions. Cette complexité croît linéairement avec la taille de la matrice des scores dû au nombre important d'utilisateurs et d'items. Ce qui occasionne, par la suite, des difficultés en termes d'espace mémoire et de temps de calcul. De l'autre côté, la rareté de données est due au fait que souvent les utilisateurs n'évaluent pas la totalité des items disponibles, mais plutôt qu'un sous ensemble réduit de ces items. Ce dernier problème est aggravé par le fait que les utilisateurs et/ou les items nouvellement ajoutés au système peuvent ne pas avoir aucune évaluation via ce type de filtrage collaboratif.

Pour faire face à ces différents problèmes, des techniques de recommandation basées sur des modèles ont été développées. En effet, et contrairement aux algorithmes de filtrage collaboratif basés sur la mémoire, les techniques basées sur les modèles utilisent les informations disponibles sur les différentes évaluations afin d'élaborer un modèle qui génère des prédictions. Le principe est d'analyser l'historique d'interaction des différents utilisateurs via une phase d'apprentissage préalable, afin d'apprendre un modèle descriptif de ces préférences décrivant les paramètres optimaux du modèle. Ce dernier sera utilisé, par la suite, pour faire une recommandation ou prédire toutes les évaluations manquantes.

Selon (Luo et al., 2012), les méthodes de filtrage collaboratif basées sur les modèles sont devenues populaires et très utilisées par rapport à celles basées sur la mémoire. Ceci est notamment grâce à leur grande capacité d'expression pour décrire les divers aspects relatifs à la matrice des évaluations. Aussi, l'avantage pour ces méthodes est que les modèles sont conçus en offline, pour être utilisés rapidement et facilement lors du processus de recommandation. En fin, ces techniques sont généralement accompagnées d'une plus grande précision dans la prédiction des évaluations manquantes.

Dans la littérature, de nombreuses méthodes ont été utilisées (voir Figure 3.7); souvent inspirées d'algorithmes d'apprentissage automatique. Ces méthodes permettent d'apprendre un modèle descriptif des préférences des utilisateurs, puis l'utilisent pour prédire les évaluations manquantes.

3.3.4 Les modèles à facteurs latents

Le modèle à facteurs latents, connu aussi sous le nom de factorisation matricielle (Koren et al., 2009), est l'approche la plus couramment utilisée dans le filtrage collaboratif. La factorisation matricielle permet de traiter efficacement les différents problèmes liés au passage à l'échelle (*scalability problem*) (Schafer et al., 2007), et aussi aux problèmes relatifs à la rareté de données (*sparsity problem*) (Ekstrand, 2011; Salakhutdinov and Mnih, 2009).

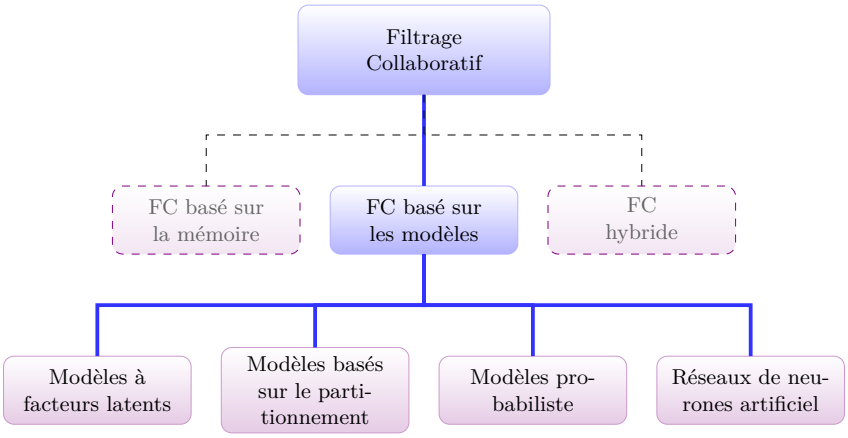


Figure 3.7 – Classification des techniques de filtrage collaboratif basées sur les modèles.

L'idée principale de la factorisation matricielle est de décomposer la matrice d'évaluation initiale en un produit de deux matrices à dimensions réduites afin de fournir, en offline, une représentation de faible rang pour un modèle initial de grande dimension. Cette représentation passe par la suite dans une phase d'apprentissage, avant d'être utilisée dans le processus de prédiction des différents scores manquants.

Partant du principe que les préférences et les intérêts des utilisateurs par rapport aux différents items sont influencés par un nombre de caractéristiques non observables; le modèle à facteurs latents fait correspondre ces utilisateurs/items à un espace factoriel latent commun de dimension réduite $\mathcal{K}$, de sorte que l'interaction entre eux est modélisée par un produit interne dans cet espace. À noter que la dimension $\mathcal{K}$ doit être très inférieure à la taille des données. Par conséquent, chaque utilisateur u est modélisé par un vecteur de caractéristique $p_u \in \mathbb{R}^{\mathcal{K}}$ représentant les variables latentes explicatives de cet utilisateur. De même, chaque item i est modélisé par un vecteur de caractéristique $q_i \in \mathbb{R}^{\mathcal{K}}$ représentant aussi, les variables latentes explicatives de cet item. La Figure 3.8 illustre en détail le processus de la factorisation matricielle.

Ainsi, le problème de factorisation matricielle est considéré comme un problème d'optimisation qui consiste à décomposer la matrice initiale de scores $\mathcal{R}$ en un produit de deux matrices (équation 3.3.11) de facteurs latents $P \in \mathbb{R}^{||\mathcal{U}||\times\mathcal{K}}$ et $Q \in \mathbb{R}^{||\mathcal{I}||\times\mathcal{K}}$ ayant des propriétés spécifiques, de sorte que la reconstruction (équation 3.3.12) estime la correspondance entre les caractéristiques de l'utilisateur u et de l'item i (c'est-à-dire le score estimé ou la prédiction $pred(u,i)$).

$$\mathcal{R} \approx \hat{\mathcal{R}} = P \times Q^T \tag{3.3.11}$$

$$pred(u,i) = \hat{r}_{ui} = p_u q_i^T \tag{3.3.12}$$

La factorisation matricielle a pour objectif de résoudre le problème de minimisation suivant:

$$\min_{P,Q} \sum_{(u,i)\in\mathcal{S}} (r_{u_i} - \hat{r}_{u_i})^2 + \lambda(||p_u||_F^2 + ||q_i||_F^2) \tag{3.3.13}$$

où:

— $\mathcal{S} = (u,i) \in \mathcal{U} \times \mathcal{I}$ tel que $y_{u,i}$ est observée;

— λ est un paramètre de régularisation;

— $||.||$ est la norme de Frobenius.

Les termes $||p_u||_F^2$ et $||q_i||_F^2$ permettent de régulariser les matrices P et Q, afin d'éviter le problème de sur-apprentissage.

À noter que l'erreur quadratique entre la prédiction et la valeur réelle de l'équation 3.3.13, ici notée par $(r_{u_i} - \hat{r}_{u_i})^2$ représente la confiance accordée à la prédiction. Cependant, dans le cadre

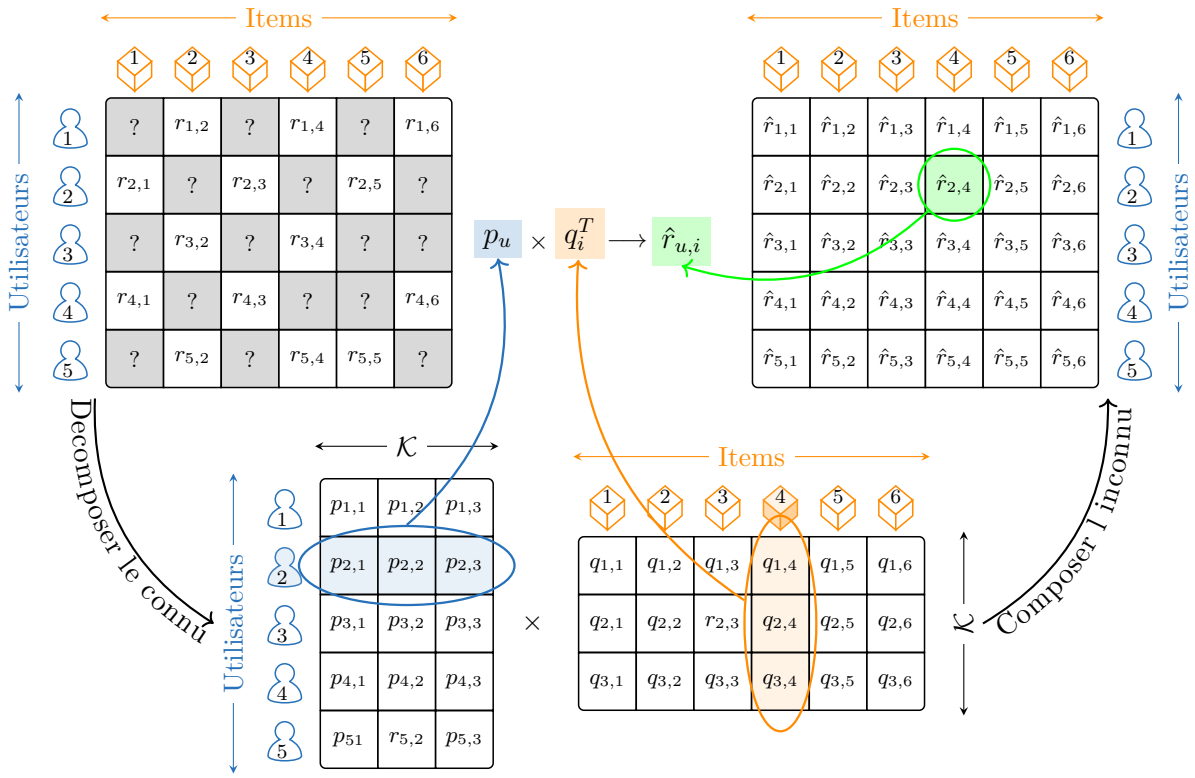


Figure 3.8 – Factorisation matricielle

d'une modélisation probabiliste un autre type d'erreur est utilisé connu sous le nom de maximum de vraisemblance (nous reviendrons à cette notion dans la partie 18).

Dans la partie qui suit, nous présenterons deux types de méthodes de réduction de la dimension, à savoir la Décomposition en Valeur Singulière (Koren et al., 2009) (ou *Singular Value Decomposition (SVD)*); et la factorisation en matrice non négative (ou *Non Negative Matrix Factorization (NMF)*) (Figure 3.9). Ces méthodes consistent à projeter les items (et/ou les utilisateurs) dans une dimension réduite définie par des variables latentes.

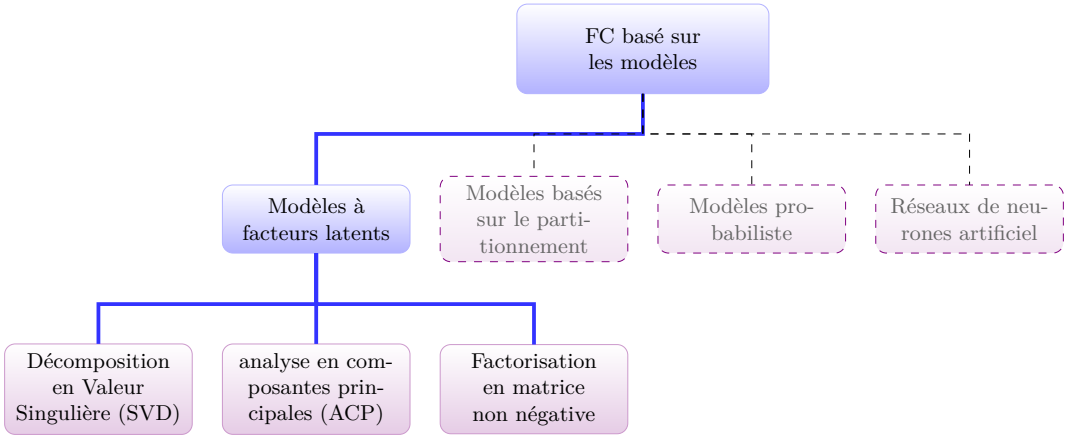


Figure 3.9 – Méthodes à facteurs latents

3.3.5 Les modèles probabilistes

Le filtrage collaboratif à base de modèles probabilistes vise à élaborer des modèles probabilistes du comportement des utilisateurs et à utiliser ces modèles pour prédire leurs comportements futurs. En effet, dans ce type d'algorithmes le calcul des prédictions est représenté sous forme de distribu-

tions de probabilité. L'idée sur laquelle reposent ces méthodes consiste à évaluer la probabilité qu'un utilisateur u attribue une note v à l'item i, notée $P(v|u,i)$. La prédiction $pred(u,i)$, telle que définie par la formule 3.3.14, doit correspondre soit à la note escomptée, soit à la note ayant la plus grande probabilité (Breese et al., 1998):

$$pred(u,i) = \sum_{v \in V} v.P(v|u,i) \tag{3.3.14}$$

où V est l'ensemble de valeurs que peut prendre une note.

Les approches probabilistes dans le domaine de filtrage collaboratif intègrent notamment les réseaux bayésiens. (Breese et al., 1998), dans un des travaux sur le filtrage collaboratif utilisant le modèle probabiliste, sont parmi les premiers à avoir proposé un modèle construit à partir des réseaux bayésiens et exploitant les arbres de décision pour calculer les probabilités. Ils ont démontré, en effet, que ce type de modèle en le combinant avec les arbres de décision améliore la précision pour un processus de recommandation.

Classificateur naïf de Bayes

Le classifieur naïf de Bayes est une approche probabiliste pour l'apprentissage inductif[17]. Il est considéré comme une forme particulière et très simple de classification probabiliste basée sur le théorème de Bayes avec une grande indépendance des hypothèses (Friedman et al., 1997). Ces types d'approches génèrent un modèle probabiliste basé sur l'historique des évaluations observées précédemment. Pour classer un ensemble d'observations, un classifieur naïf bayes est entraîné dans un contexte d'apprentissage supervisé.

Le principe du classificateur naïf de Bayes est de déterminer la probabilité d'appartenance d'une ressource r, ayant pour attributs $(\theta_1, \ldots, \theta_k)$, à une classe C, notée $P(C|r)$. Cette probabilité est calculée comme suit (équation 3.3.15):

$$P(C|r) = P(C|\theta_1, \ldots, \theta_k) = \frac{P(C)\prod_{i=1}^{k} P(\theta_i|C)}{P(r)} \tag{3.3.15}$$

Avec:

— $P(C)$ est la probabilité d'observer une ressource dans la classe C;

— $P(r|C) = \prod_{i=1}^{k} P(\theta_i|C)$ la probabilité d'observer la ressource r sachant la catégorie C;

— la probabilité $P(r)$ d'observer la ressource r.

Les valeurs de $P(C)$ et de $P(r|C)$ sont estimées à partir des données d'apprentissage.

Pour classer la ressource r, chaque valeur de la formule 3.3.15 est estimée pour chaque classe, et la classe avec la plus forte probabilité sera choisie:

$$P(C|r) = \underset{C_j}{\operatorname{argmax}} \frac{P(C)\prod_{i=1}^{k} P(\theta_i|C)}{P(r)} \tag{3.3.16}$$

Cette approche est inspirée de la formule de Bayes qui permet d'exprimer la probabilité qu'un évènement arrive à partir d'un autre évènement qui s'est déjà réalisé. Elle est basée sur l'estimateur du maximum a posteriori (MAP) (Kotsiantis, 2007) qui, au moyen de la probabilité antérieure de la catégorie, attribue la meilleure catégorie C à la ressource r.

« Cross-sell » (Kitts et al., 2000) est considéré parmi les premiers travaux qui ont utilisé les réseaux bayésiens. Il est basé sur la notion de probabilités conditionnelles[18] combinée avec un classifieur linéaire

17. Wikipédia: L'apprentissage inductif est un sous domaine de l'apprentissage automatique basée sur la programmation logique. (page consultée le 22/8/2021)

18. La notion de probabilité conditionnelle permet de tenir compte dans une prévision d'une information complémentaire.

bayésien naïf pour effectuer de la recommandation. Dès que le système collecte l'historique d'achat des utilisateurs, l'algorithme estime, pour chaque paire d'article (i, j), la probabilité $P(i|j)$ qu'un utilisateur achète l'article i étant donné qu'il a déjà acheté l'article j.

Les réseaux bayésiens pour le filtrage collaboratif ont également été utilisés dans plusieurs travaux tels que (Chien and George, 1999; Valdiviezo-Diaz et al., 2019; Zigoris and Zhang, 2006).

Factorisation en matrice probabiliste

Les algorithmes probabilistes de filtrage collaboratif utilisent explicitement les distributions de probabilité, via un modèle linéaire probabiliste avec un bruit d'observation gaussien, dans le calcul des scores prédits où lors d'établissement d'une liste d'items à recommander. Une des approches les plus utilisées est la factorisation probabiliste matricielle proposée par (Salakhutdinov and Mnih, 2009), en anglais « Probabilistic Mtarix Factorization » (PMF).

L'idée de base pour cet algorithme de filtrage collaboratif est de considérer que les préférences et les intérêts des utilisateurs suivent une loi de probabilité. Ainsi, la probabilité qu'un item i est choisi par l'utilisateur u par rapport à l'espace factoriel latent de dimension réduite $\mathcal{K}$ peut être exprimée comme suit:

$$P(i|u) = \sum_{k \in \mathcal{K}} P(i|k)P(k|u) \tag{3.3.17}$$

où

— $P(i|k)$ est la probabilité qu'un item i possède un facteur latent k;

— $P(k|u)$ représente la probabilité qu'un facteur latent k soit interagi par un utilisateur u.

Supposant que la distribution d'évaluations des différents utilisateurs effectuées sur l'ensemble d'items, souvent sous ensemble, suit une loi gaussienne, nommée $\mathcal{N}$. La fonction de densité de cette loi sera définie par:

$$\mathcal{N}(x, \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

où:

— σ représente la variance des données;

— μ est la moyenne des données.

En se basant sur l'équation précédente ainsi que les matrices de facteurs latents P et Q, la prédiction entre l'utilisateur u et l'item i peut donc se calculer ainsi:

$$\hat{r}_{u,i} \sim \mathcal{N}(R_{u,i}|P_u Q_i^T, \sigma^2) \tag{3.3.18}$$

En utilisant la distribution conditionnelle sur les évaluations observées, c'est-à-dire la vraisemblance associée aux données d'observation (matrice des scores $\mathcal{R}$) et les distributions a priori sur les matrices de facteurs latents P et Q. L'équation précédente peut être encore généralisée sur toute la matrice des évaluations $\mathcal{R}$ comme suit:

$$p(R|P, Q, \sigma^2) = \prod_{u \in \mathcal{U}} \prod_{i \in \mathcal{I}} [\mathcal{N}(R_{u,i}|P_u Q_i^T, \sigma^2)]^{O_{u,i}} \tag{3.3.19}$$

Avec $O_{u,i} = \begin{cases} 1 & \text{l'interaction (u,i) est observée} \\ 0 & \text{sinon} \end{cases}$

Les lois a priori sur les utilisateurs et sur les items seront définies comme suit:

$$p(P|\sigma_P^2) = \prod_{u \in \mathcal{U}} \mathcal{N}_{\mathcal{K}}(P_u|0, \sigma_P^2)$$

et

$$p(Q|\sigma_Q^2) = \prod_{i \in \mathcal{I}} \mathcal{N}_{\mathcal{K}}(Q_i|0, \sigma_Q^2)$$

Dans le cas de la factorisation en matrice probabiliste, le problème de complétion de matrice vu précédemment (équation 3.3.13) est interprété comme l'estimation au sens du maximum de vraisemblance d'un modèle à bruit additif gaussien:

$$\max_{P,Q} log\ p(R|P,Q,\sigma^2,\sigma_P^2,\sigma_Q^2)p(Q|\sigma_Q^2)p(Q|\sigma_P^2) \tag{3.3.20}$$

Ainsi, afin de trouver les estimateurs des coefficients des matrice P et Q, contenant les facteurs latents des utilisateurs et des items, l'approche de factorisation en matrice probabiliste utilise l'équation 3.3.16. Pour plus de détails sur l'apprentissage des facteurs latents pour la factorisation en matrice probabiliste, le lecteur intéressé peut se référer à (Gouvert, 2019; Salakhutdinov and Mnih, 2008).

Les approches probabilistes, vues précédemment, permettent de surmonter le problème de la rareté de données et d'améliorer ainsi, la qualité des recommandations (Breese et al., 1998). L'un des principaux avantages des algorithmes probabilistes est qu'ils sont en mesure de produire une répartition de probabilité parmi les valeurs d'évaluations possibles (Schafer et al., 2007). Cependant, leur principal inconvénient est que ces types de réseaux restent coûteux à construire et ne sont donc pas très adaptés aux grands volumes de données.

3.4 Prédiction versus Recommandation

Il est très important de faire la différence entre les concepts *prédiction* et *recommandation* pour les systèmes de recommandation (Sarwar et al., 2001). Ces deux termes imposent des contraintes distinctes pour un système de filtrage collaboratif.

En effet, la *recommandation* consiste à présenter une liste ou un sous ensemble d'items, généralement pas tous, à un utilisateur particulier. Ces items sont classés par ordre d'utilité pour l'utilisateur concerné. En d'autres termes, cela consiste à prédire le score que l'utilisateur attribuera à un item donné, sans qu'il soit nécessaire de disposer d'informations sur tous les autres items; puis à classer les items en fonction de ce score prédit. Ce type d'algorithmes permet d'économiser de l'espace mémoire ainsi que du temps de calcul (Linden et al., 2003; Sarwar et al., 2001). En littérature, ce processus est également connu sous le nom de Top-n recommandation.

Contrairement à l'estimation d'un ordre relatif correct des items qui caractérise un processus de recommandation; la *prédiction* consiste à calculer une valeur numérique exprimant l'évaluation probable qu'un utilisateur donnera à un item particulier. Dans ce cas, la prédiction doit considérer l'ensemble des items disponibles, même ceux qui sont rarement évalués. La Figure 3.10 illustre le schéma du processus traditionnel pour le filtrage collaboratif.

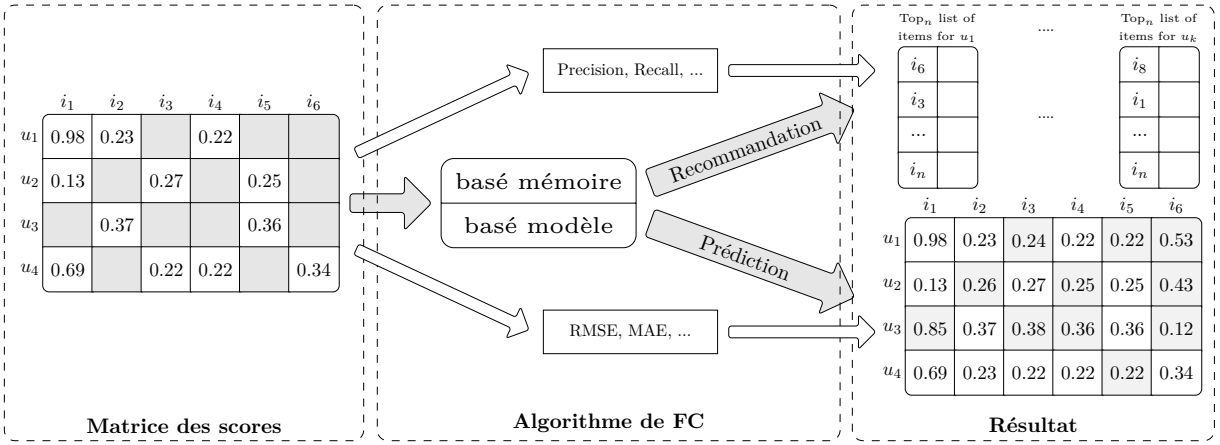


Figure 3.10 – Prédiction vs Recommandation

Cette distinction fondamentale entre ces deux concepts requière de différentes exigences pour un système de filtrage collaboratif. Celles-ci sont résumées comme suit:

— Pour faire la recommandation d'item, il n'est pas nécessaire de considérer tous les items du système. En revanche, et pour fournir des prédictions pour un item particulier, il est obligatoire de prendre en considération chaque item dans le processus de calcul.

— Les algorithmes de recommandation s'exécutent en un temps réduit et avec moins d'espace mémoire que les algorithmes utilisés pour les prédictions.

— L'évaluation de la prédiction et de la recommandation est complètement différente. Afin de connaître la qualité des techniques, des méthodes et des algorithmes de prédiction, plusieurs métriques d'évaluation peuvent être utilisées telle que l'erreur quadratique comme la RMSE (*Root of Mean Square Error*) ou la MSE (*Mean-Squared Error*). Nous pouvons utiliser aussi l'erreur absolue moyenne MAE (*Mean Absolute Error*) ou sa variante NMAE (*Normalized Mean Average Error*). Cependant, pour la recommandation d'autres mesures de qualité sont fréquemment utilisées telle que la précision, le rappel et la F-measure.

3.5 évaluation des techniques du FC

L'évaluation des résultats de la recommandation/prédiction est une tâche primordiale dans le contexte des systèmes de recommandation (Jalili et al., 2018) en général, et celui du filtrage collaboratif (Herlocker et al., 2004) en particulier. Grâce à cette évaluation la capacité des différents algorithmes et approches, en termes de précision des résultats de prédiction, peut être discutée. Ainsi, plusieurs mesures statistiques ont été proposées afin d'évaluer la performance de prédiction pour les systèmes de recommandation.

Nous nous intéressons, dans cette partie du chapitre, aux différentes mesures d'évaluation de la précision des résultats de prédictions. En effet, afin de juger et d'évaluer la performance des systèmes de recommandation basés sur filtrage collaboratif, plusieurs mesures d'évaluation ont été présentées. L'objectif est de mesurer la différence entre les valeurs des évaluations (scores) prédites par le système de recommandation, et celles des observations réelles déjà présentes dans le système (elle sont fournies par les utilisateurs ou le fournisseur de services). La mesure la plus fréquemment utilisée dans cette catégorie est donnée par l'équation 3.5.1. Elle représente l'erreur moyenne absolue (Mean Absolute Error (MAE)) dans un ensemble de prédictions, sans prendre en considération leur direction.

$$MAE = \frac{1}{|N|} \sum_{(u,i) \in N} |r_{u,i} - \hat{r}_{u,i}| \tag{3.5.1}$$

Avec: $r_{u,i}$ est l'évaluation réelle, $\hat{r}_{u,i}$ est le score prédit et N est le nombre total des prédictions.

La deuxième métrique s'agit de la racine carrée de la moyenne des différences au carré entre la prédiction et l'observation réelle, à savoir, la Root Mean Square Error (RMSE). Elle permet d'accentuer les différences larges entre le prédit et l'observé.

$$RMSE = \sqrt{\frac{1}{|N|} \sum_{(u,i) \in N} (r_{u,i} - \hat{r}_{u,i})^2} \tag{3.5.2}$$

Il est évident que l'intervalle de valeurs, pour ces deux mesures, varie entre 0 et $+\infty$ et sont insensibles à la direction des erreurs. Aussi, l'évaluation est orientée négativement, c'est-à-dire, plus l'erreur se rapproche de la valeur 0 plus la précision de la prédiction est meilleure. Cependant, la RMSE considère davantage les écarts élevés, par rapport à la MAE, puisque les erreurs sont élevées au carré avant d'être moyennées. Cela signifie que la RMSE devrait être plus utile lorsque les écarts importants sont particulièrement indésirables.

Il est important de mentionner que nous nous sommes intéressés qu'à un seul type de méthodes d'évaluation. Cette catégorie, connue sous le nom d'évaluation hors ligne (ou *offline*), est considérée comme la plus simple à réaliser et la moins risquée par rapport aux deux autres types; à savoir, études avec des utilisateurs (*user studies*) et tests réels (*real life testing*), selon la classification donnée dans (Shani and Gunawardana, 2011).

3.6 Conclusion

Dans cette partie de thèse, nous nous sommes intéressés, essentiellement, à la problématique de la prédiction basée sur le filtrage collaboratif. De ce fait, ce chapitre était consacré à la présentation des principes de base relatifs au filtrage collaboratif dans le contexte des systèmes de recommandation. Ces derniers ont été élaborés, comme nous l'avons présenté au début de ce chapitre, afin d'assister les utilisateurs à trouver les meilleures ressources par rapport à leurs préférences, parmi le nombre important des choix existants.

Nous avons suivi avec la classification des méthodes de filtrage collaboratif en deux approches principales: les méthodes de filtrage collaboratif basées sur le voisinage et celles basées sur les modèles. Dans les systèmes basés sur le voisinage, les performances de la prédiction sont en étroites relation avec le choix de la métrique de mesure de similarité. En revanche, pour un filtrage collaboratif basé sur les modèles, les performances sont relatives aux paramétrages du modèle prédictif qui a été choisi. Ensuite, nous avons présenté la différence qu'existe entre la notion de recommandation et prédiction dans le contexte des systèmes de recommandation en générale, et celui du filtrage collaboratif en particulier. Nous avons enfin consacré la dernière partie pour discuter la notion des mesures d'évaluation utilisées pour valider la performance des prédictions et des recommandations générées.

Actuellement et avec la disponibilité de grands jeux de données sur les préférences des utilisateurs; les techniques statistiques basiques s'avèrent de moins en moins envisageables pour entreprendre un processus de prédiction. Cependant, l'apprentissage automatique pour le filtrage collaboratif est considéré comme une meilleure alternative à ces techniques mathématiques classiques. Les différents méthodes et techniques développées dans le domaine d'apprentissage automatique ainsi que d'apprentissage profond feront l'objet du chapitre suivant.

Chapitre 4

L'apprentissage profond

Sommaire

4.1 Introduction 43
4.2 Apprentissage automatique 44
4.2.1 Origine et inspiration 45
4.2.2 Perceptron monocouche 46
4.2.3 Réseaux de neurones artificiels 48
4.3 Apprentissage profond 54
4.3.1 Réseaux de neurones convolutifs 55
4.3.2 Réseaux de neurones récurrents 56
4.4 Les modèles génératifs d'apprentissage profond 57
4.4.1 Auto-encodeurs 57
4.4.2 Les réseaux antagonistes génératifs 61
4.4.3 Principes et définition 61
4.4.4 L'apprentissage dans les GANs 62
4.5 Réseaux auto-organisateurs 63
4.5.1 Principe de fonctionnement 63
4.5.2 L'apprentissage dans les réseaux de Kohonen 63
4.6 Conclusion 64

4.1 Introduction

L'intelligence artificielle (IA) (ou *Artificial intelligence (AI)*) est un champ de recherches fondé sur un ensemble d'algorithmes et de méthodes visant à reproduire et à représenter au mieux, à l'aide de systèmes artificiels, les différentes capacités cognitives dont font preuve les êtres vivants. Ces techniques permettent de mettre en place des représentations qui structurent des modèles de compréhension, de perception et de décision. Selon la définition fournie par *Nils J.Nilsson*: « *Artificial intelligence is that activity devoted to making machines intelligent, and intelligence is that quality that enables an entity to function appropriately and with foresight in its environment* » (Nilsson, 2011). En d'autres termes, l'intelligence artificielle a le projet de construire des artefacts intelligents qui sont en mesure de percevoir leur environnement, traiter l'information et agir en conséquence dans le but de modéliser et simuler le raisonnement des êtres vivants à un degré ou à un autre.

Jusqu'aux années quatre-vingt-dix, les travaux sur l'intelligence artificielle se limitaient, à titre d'exemple, à la réalisation de programmes ou logiciels dotés de moteurs d'inférences pour des systèmes experts modélisant l'expertise d'un humain sur une tâche précise, ou au codage de programmes assez complexes sur des thématiques diverses tel que la traduction automatique de texte, la reconnaissance de caractères, la crypto-arithmétique, les jeux, etc. Cependant, cette approche « artisanale » dans la mise en œuvre de ces travaux a présenté certaines limites. En effet, il est très difficile de l'appliquer à

des tâches d'un niveau de complexité plus élevé, comme la reconnaissance automatique de la parole, le traitement d'images, la fouille de données, etc. Il est pratiquement impossible d'établir un programme robuste capable de gérer toutes les variables de tâches aussi complexes; c'est pour faire face à cette difficulté que l'apprentissage automatique (ou *Machine learning (ML)*) a vu le jour.

Historiquement, la première utilisation du terme remonte à 1959, par Arthur Samuel (Samuel, 1959). L'apprentissage automatique est considéré comme un sous domaine de l'intelligence artificielle, comme l'illustre la figure 4.1, représentant la branche connexionniste[19] de l'intelligence artificielle. L'apprentissage automatique vise alors à l'élaboration de modèles capables d'extraire une connaissance exploitable et pertinente à partir des grands volumes de données.

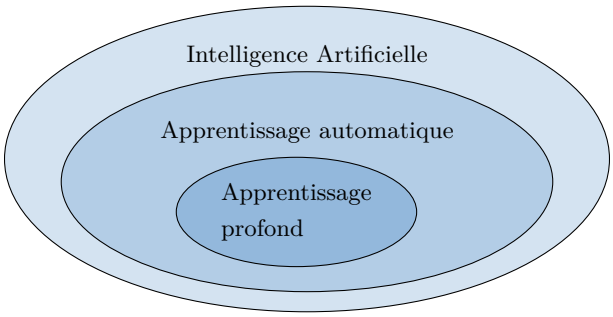


Figure 4.1 – Vue globale de l'intelligence artificielle

Cependant, et malgré leur efficacité à résoudre une grande variété de problèmes complexes, les méthodes d'apprentissage classique présentent certaines limites (nécessité de l'intervention d'un expert, difficulté à construire le vecteur de caractéristiques, etc.). L'apprentissage profond (Goodfellow et al., 2016) (*deep learning*), qui est une branche de l'apprentissage automatique basée sur les réseaux de neurones artificiels (Abiodun et al., 2018) (*artificial neural network* ANN) (figure 4.1), représente un moyen plus efficace afin de surmonter ce type de problèmes.

Nous représentons dans ce chapitre une vue d'ensemble sur les différentes notions liées à l'apprentissage automatique ainsi que l'apprentissage profond. En effet, nous détaillons les différentes notions liées aux algorithmes d'apprentissage automatique du plus petit module (neurone ou perceptron) jusqu'au fonctionnement total d'un réseau multicouche en passant par les différentes fonctions d'activation utilisées dans ce domaine. Ensuite, nous présentons plus particulièrement les modèles générationnels ainsi que les différentes méthodes de partitionnement de données (*data clustering*).

4.2 Apprentissage automatique

Inutile de dire que l'apprentissage automatique est devenu l'une des technologies les plus passionnantes de notre époque. En effet, de grandes entreprises telles que Google, Facebook, Apple, Amazon, IBM et bien d'autres investissent à juste titre des capitaux importants dans le développement de méthodes et d'applications logicielles dans le domaine de l'apprentissage automatique. Cet axe de recherche consiste à l'application d'une variante d'algorithmes qui permettent d'extraire des informations à partir de données sources puis les présenter sous forme d'un modèle particulier. Ce dernier est utilisé pour déduire d'autres informations qu'éventuellement n'existaient pas dans le modèle initial.

Le but principal de l'apprentissage automatique est d'automatiser une tâche de classification, régression, association, ou regroupement. Jusqu'à récemment, les systèmes classiques d'apprentissage

19. Les techniques d'apprentissage utilisant des architectures basées sur les réseaux de neurones sont connues sous le nom d'approches connexionnistes. Par opposition, il existe les approches symboliques qui ont pour objectif de doter les machines, à travers un ensemble de programmes ou logiciels, par des règles permettant de manipuler des représentations de connaissances de haut niveau.

automatique étaient composés de deux modules principaux: un extracteur de caractéristiques (*feature extractor*) et un classificateur entraînable (*trainable classifier*). Le premier module, conçu « manuellement » par un expert dans le domaine d'application, permet d'extraire et de préparer le vecteur des caractéristiques discriminantes à partir des données originales brutes. Le second module est un classificateur utilisé afin d'apprendre, à partir du vecteur de caractéristiques initial, une fonction qui corresponde au mieux à l'objectif à atteindre (classement des données, prédiction, etc.).

Dans cette section, nous présentons brièvement le fonctionnement des réseaux de neurones artificiels en introduisant leurs principaux concepts, modèles et algorithmes les plus pertinents.

4.2.1 Origine et inspiration

Le cerveau humain est la structure biologique la plus complexe du monde. En s'inspirant des acquis de la neurobiologie, les travaux sur l'apprentissage automatique ont permis de créer et d'explorer des systèmes artificiels qui permettent de reproduire le comportement et les processus de pensée des êtres biologiques.

Bref aperçu historique

Les recherches sur les réseaux de neurones artificiels ont connu trois périodes de revitalisation. La première est dans les années 1940, grâce notamment aux travaux de (McCulloch and Pitts, 1943), qui ont développé le premier neurone artificiel pour la résolution des fonctions booléennes. La seconde période était dans les années 1960 avec l'introduction de la notion de perceptron pour une classification binaire développé par *Rosenblatt* (Rosenblatt, 1958). Contrairement à l'approche booléenne proposée précédemment, l'approche de *Rosenblatt* était basée sur un processus d'apprentissage. En 1958, les travaux de *Frank Rosenblatt* ont abouti à l'élaboration de la version « hardware » (un dispositif électronique) du perceptron. En 1960, « Mark-1 » était la première machine développée simulant le travail conjoint de l'œil et du cerveau humain [20].

Dans leur travail (Minsky and Papert, 2017), les auteurs ont souligné la capacité limitée du perceptron quant à la résolution d'une simple fonction booléenne XOR à cause de son adaptation au séparations linéaires (le perceptron fonctionne que sur des données linéairement séparables). Les résultats de ces travaux ont éteint, pour longtemps, l'enthousiasme de la plupart des chercheurs dans le domaine des ANN. L'accalmie qui s'est produite dans les recherches sur les réseaux de neurones a duré près de 15 ans.

Depuis le début des années 1980, les réseaux de neurones ont de nouveau suscité l'intérêt de la communauté de chercheurs grâce aux travaux de Hopfield (Hopfield, 1982, 1984), et à l'introduction de l'algorithme de rétropropagation (proposé pour la première fois par *Verbos* (Werbos, 1990) pour l'apprentissage des perceptrons multicouches (*multilayer perceptron* MLP).

Neurone biologique

La manière classique pour résoudre n'importe quel problème via une machine est de composer un algorithme qui est implémenté par la suite sous la forme d'un programme. Par exemple, pour concevoir un robot joueur de basket, un ensemble d'équations différentielles représentant la trajectoire de la balle à partir de différentes positions doit être composé. Pour cela, plusieurs paramètres seront pris en considération tels que la résistance de l'air, la parallaxe des capteurs, la position du joueur, etc. Cependant qu'en réalité, personne n'exige qu'un joueur ordinaire de basket doit connaître si bien les mathématiques.

Le cerveau humain, évidemment, résout ce type de problème, ainsi que tout autres problèmes, d'une manière fondamentalement différente. Le système nerveux et le cerveau humain sont composés de neurones, massivement interconnectés [21] par des fibres nerveuses. Ces fibres nerveuses ont pour objectif de transmettre des messages via des impulsions électriques entre les différents neurones.

20. Mark-1 pouvait reconnaître certaines des lettres écrites sur des cartes portées devant une caméra (les "yeux" de la machine). Elle pouvait distinguer les lettres de l'alphabet; cependant elle était sensible à leur orthographe.

21. En général, le cerveau humain contient environ 10^{14} à 10^{15} interconnexions.

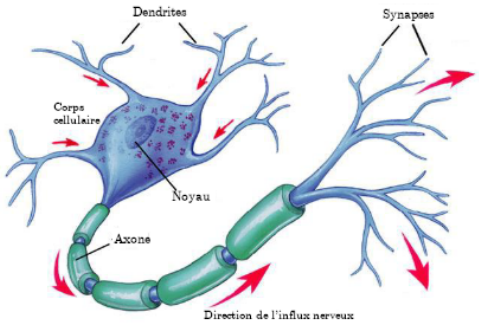


Figure 4.2 – Neurone biologique

Considérons la structure d'un neurone biologique (figure 4.2) qui est composée de plusieurs parties: le noyau, le corps cellulaire, les fibres nerveuses et les synapses. Chaque neurone possède deux types de fibres nerveuses: les dendrites, le long desquelles les impulsions sont reçues, et le seul axone auquel le neurone peut transmettre des impulsions. L'axone, à l'extrémité, se ramifie en fibres et entre en contact avec les dendrites d'autres neurones par l'intermédiaire de structures spéciales, à savoir les synapses, qui influent sur la force de l'impulsion électrique. En effet, la force des impulsions est modifiée en fonction de la nature des synapses. Les synapses excitatrices augmentent cette impulsion, tandis que les inhibitrices réduisent sa valeur. Ensuite, ces impulsions sont regroupées et traitées par le corps cellulaire du neurone.

4.2.2 Perceptron monocouche

Avant d'aborder en détail les notions de base sur les réseaux de neurones artificiels tels que nous les connaissons aujourd'hui, un bref aperçu sur l'élément de base qui compose un réseau de neurones est plus que nécessaire. En effet, le perceptron ou neurone formel, ou plus spécifiquement, le perceptron monocouche est le premier neurone artificiel utilisé dans l'apprentissage automatique, il appartient à la catégorie des classificateurs linéaires. Les travaux de (Rosenblatt, 1958), (Widrow and Hoff, 1960) ont largement contribué à leur développement.

Neurone artificiel

Le modèle mathématique généralisé d'un neurone artificiel est le suivant (Luger, 2005) (Figure 4.3):

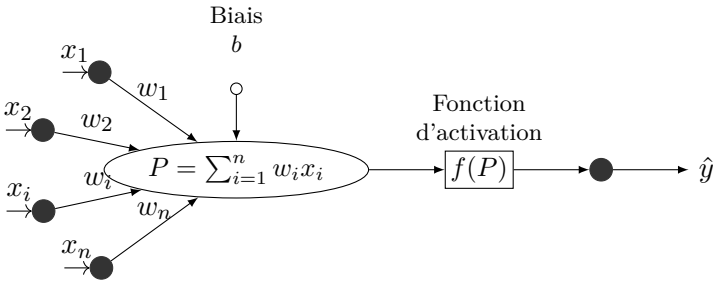


Figure 4.3 – Neurone artificiel

— Signaux d'entrée x_i: ce sont les données provenant de l'environnement extérieur ou d'autres neurones actifs. Ils sont exprimés par les valeurs $x_i, i \in \mathbb{N}$; ces valeurs peuvent être discrètes à partir des ensembles $[0, 1]$ ou $[-1, 1]$ ou prendre toutes autres valeurs;

— Coefficients de poids w_i: déterminent la force de la connexion entre les neurones. Ils représentent la notion des dendrites pour un neurone biologique;

— Fonction de combinaison, ou le niveau d'activation, entre les valeurs des signaux en entrée x_i avec leurs poids w_i du neurone en question exprimée par la somme pondérée suivante $P = \sum_{i=1}^{n} w_i x_i$;

— Fonction d'activation $f(P)$: utilisée pour calculer la valeur de sortie du signal transmis aux autres neurones.

Un neurone artificiel imite un neurone biologique dans plusieurs fonctions. En effet, les signaux d'entrée x_i sont pondérés en les multipliant par des coefficients w_i appelés poids synaptiques. Par la suite, la somme pondérée résultante $P = \sum_{i=1}^{n} w_i x_i$ est traitée par une fonction $f(P)$ appelée fonction d'activation. Il est à noter que même le signal de sortie Y peut également faire l'objet d'une pondération via une mise à l'échelle.

Perceptron

La composante principale dans un réseau de neurones est automatiquement, le neurone artificiel. La figure 4.3 présente la structure de base du perceptron, qui se compose d'un additionneur et d'un bloc de transformation non linéaire (la fonction d'activation). Cette structure permet d'effectuer l'opération de transformation non linéaire de la somme des produits des valeurs d'entrée avec les coefficients de poids. Ainsi un neurone artificiel est une application $f_w : X \to Y$ définie par:

$$\hat{y} = f_w(x) = \phi(\sum_{i=1}^{n} w_i x_i) = \phi(WX) \tag{4.2.1}$$

où:

— $X = (x_1, x_2, \ldots, x_n)^T$ représente l'espace d'entrée généralement $\mathbb{R}^n$ avec $n \in \mathbb{N}$;

— $\hat{y}$ un espace de sortie où les valeurs peuvent être discrètes à partir d'un ensemble dénombrable fin ($[0, 1]$ ou $[-1, 1]$), ou prendre toutes autres valeurs dans $\mathbb{R}^n$;

— $W = (w_1, w_2, \ldots, w_n) \in \mathbb{R}^n$ est un vecteur des poids synaptiques;

— $\phi : \mathbb{R} \to \mathbb{R}$ un opérateur de transformation non linéaire. Cet opérateur est appelé fonction d'activation des éléments neuronaux, et qui doit être dans l'idéal monotone, dérivable et saturante [22].

Les réseaux de neurones à couche unique forment une surface de division linéaire, c'est-à-dire, ils permettent de résoudre les problèmes linéairement séparables à deux classes seulement. Un perceptron monocouche s'interprète géométriquement comme une droite qui permet de séparer les différentes instances en deux classes distinctes comme l'illustre la Figure 4.4.

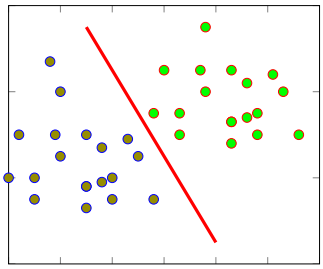


Figure 4.4 – Séparation binaire des instances

Le perceptron est caractérisé par n entrées et une seule sortie prédite y. Chaque $i^{\text{ème}}$ entrée d'un neurone correspond à un coefficient de poids w_i (synapse), qui caractérise la force de la connexion synaptique (par analogie avec un neurone biologique). L'objectif principal du neurone artificiel est l'apprentissage du vecteur des poids W afin de satisfaire le modèle d'entrée X avec son modèle de sortie Y qui le correspond (c'est-à-dire $f_w : X \to Y$).

Afin d'atteindre cet objectif (c'est-à-dire déduction des valeurs optimales des poids synaptiques), un processus itératif est appliqué sur une base d'apprentissage initiale $\mathcal{D} = \{x^{(n)}, y^{(n)}\}_{n=1}^{N}$. Ce processus est illustré par l'Algorithme 1; considéré comme l'un des premiers algorithmes d'apprentissage automatique. En effet, après une initialisation aléatoire des poids synaptiques $w = (w_1, w_2, \ldots, w_m)$

22. Plus de détails sur la définition mathématique de ce terme sont présentés dans (Cybenko, 1989).

(ligne 1) ainsi que la valeur du biais b (ligne 2); le perceptron procède à la réalisation d'un ensemble d'étapes sur toute la base de d'apprentissage:

Algorithme 1 : Algorithme d'apprentissage du Perceptron

Input : $\mathcal{D} = \{x^{(n)}, y^{(n)}\}_{n=1}^{N}$

1 Initialiser aléatoirement les poids $w = (w_1, w_2, \ldots, w_m)$
2 Initialiser aléatoirement le biais b
3 $t \leftarrow 0$
4 **while** $t \leq iterations$ **do**
5 **foreach** $(x^{(i)}, y^{(i)}) \in (\mathcal{D}, \{0, 1\})$ **do**
6 $\hat{y}^{(i)} = \phi(\sum_{j=1}^{m} x_j^{(i)} \times w_j + b)$
7 $\Delta w = \alpha(y^{(i)} - \hat{y}^{(i)})x^{(i)}$
8 $w = w - \Delta w$
9 **end foreach**
10 **end while**

1. Sélection d'un exemple $x^{(i)} = (x_1, x_2, \ldots, x_m) \in \mathcal{D}$ ainsi que son modèle de sortie $y^{(i)} \in \{0, 1\}$ correspondant;
2. Détermination de la prédiction du perceptron, par rapport à l'entrée $x^{(i)}$, via l'équation suivante (ligne 6):
$$\hat{y}^{(i)} = \phi(\sum_{j=1}^{m} x_j^{(i)} \times w_j + b) \tag{4.2.2}$$
3. Modification des valeurs du vecteur des poids w en fonction de la différence entre le modèle de sortie réel et le modèle prédit (ligne 7 et 8); avec Δw la différence de changement de la valeur du poids, et α est le taux d'apprentissage (une valeur supérieure à 0 contrôlant le taux de changement à effectuer sur les poids).

Lors de sa création, le perceptron a été considéré comme une avancée majeure dans le domaine de l'apprentissage automatique. Malheureusement, plusieurs limites techniques ont été rapidement détectées. Les auteurs (Minsky and Papert, 2017) ont démontré que le problème du « *ou exclusif* » ne peut pas être résolu à l'aide d'un perceptron car celui-ci ne fonctionne que sur des données linéairement séparables [23] (voir Figure 4.4); ce qui limite l'éventail des tâches qu'il peut résoudre. Et afin de contourner ces limitations, un modèle plus complexe composé de plusieurs perceptrons a été introduit connu sous le nom de réseau de neurones artificiels.

4.2.3 Réseaux de neurones artificiels

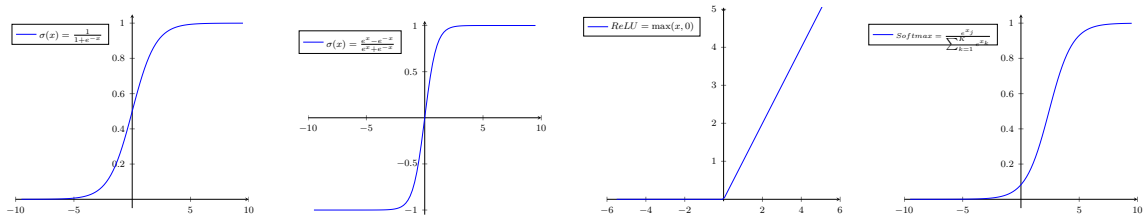
Les perceptrons multicouches sont des réseaux neuronaux multicouches, à action directe, capables d'effectuer n'importe quel mappage de vecteurs d'entrée en vecteurs de sortie. Le perceptron multicouche est composé de plusieurs types de couches: d'entrée, cachée et de sortie. Jusqu'en 2006, selon (Golovko, 2017), la communauté scientifique avait un paradigme préférentiel, à savoir qu'un perceptron avec une ou aux plus deux couches cachées est suffisamment adéquat pour résoudre divers problèmes, et l'utilisation d'un perceptron avec plus de deux couches cachées n'avait pas beaucoup d'intérêt. Actuellement, le concept dominant est qu'un perceptron avec plus de deux couches cachées est plus efficace.

23. Bien que, un perceptron monocouche puisse représenter la fonction « *XOR* » via l'utilisation d'une fonction d'activation. Ces fonctions seront étudiées en détails dans la section 4.2.3

Fonctions d'activation

Dans les réseaux de neurones artificiels, la notion de fonction d'activation est très importante car elle aide à apprendre et à donner une interprétation aux mappages non linéaires et compliqués entre les entrées et les sorties correspondantes. Le but principal des fonctions d'activation est de convertir un signal en entrée d'un nœud à un signal en sortie. Cette conversion consiste à introduire une non-linéarité au perceptron afin qu'il puisse résoudre les problèmes non linéaires et traiter les données plus complexes. Les fonctions d'activation sont spécialement utilisées dans les réseaux neuronaux artificiels pour transformer un signal d'entrée en un signal de sortie qui, à son tour, sert d'entrée à la couche suivante de la pile des couches. Dans un réseau de neurones artificiels, la somme des produits des entrées et de leurs poids correspondants est calculée, puis une fonction d'activation est appliquée afin d'obtenir la sortie de cette couche particulière et la fournir comme entrée à la couche suivante.

Parmi les fonctions d'activation les plus connues, dans la littérature, nous avons: la sigmoïde logistique, la tangente hyperbolique, l'unité linéaire rectifiée, et la Softmax, comme illustré par la figure 4.5.



(a) Fonction sigmoïde **(b)** Fonction tangente **(c)** Fonction ReLU **(d)** Fonction softmax

Figure 4.5 – Exemples de fonctions d'activation

La sigmoïde logistique La fonction sigmoïde logistique (Neal, 1992) est considérée comme la fonction d'activation la plus utilisée car il s'agit d'une fonction non linéaire. La fonction sigmoïde transforme les différentes valeurs dans la plage de 0 à 1. Elle peut être définie via l'équation 4.2.3 suivante:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.2.3}$$

Cette fonction est souvent utilisée pour les tâches de classification binaires car elle permet d'introduire plus de non linéarité aux couches cachées et/ou de sorties et de prédire les probabilités des classes en sortie.

La tangente hyperbolique La fonction tangente hyperbolique (Kalman and Kwasny, 1992), définie par l'équation 4.2.4 est similaire à la fonction sigmoïde mais elle est symétrique autour de l'origine.

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{4.2.4}$$

Cela se traduit par des signes différents de sorties des couches précédentes qui seront transmises en entrée à la couche suivante. Cela facilite la modélisation des entrées avec des valeurs négatives, neutres et positives. Selon (Karlik and Olgac, 2011), cette fonction est caractérisée par sa bonne précision en reconnaissance par rapport à la fonction sigmoïde logistique.

L'unité linéaire rectifié ReLU (Sun et al., 2015) signifie unité linéaire rectifiée, elle est une fonction d'activation non linéaire largement utilisée par les couches cachées dans réseaux de neurones. ReLU est plus efficace que d'autres fonctions comme la sigmoïde et la tangente hyperbolique, grâce notamment à sa simplicité. En effet, cette fonction, exprimée par l'équation 4.2.5, n'active pas tous les neurones en même temps, mais plutôt un certain nombre de neurones qui sont activés à la fois. Aussi, dans certains cas, les valeurs du gradient négatives sont neutralisées à zéro, ce qui permet d'accélérer le

temps d'apprentissage des réseaux de neurones. Tous ces critères ont fait de cette fonction est la plus utilisée en apprentissage profond (Ramachandran et al., 2017).

$$\text{relu}(x) = \max(x, 0) \tag{4.2.5}$$

Softmax La fonction Softmax est une fonction exponentielle normalisée (Bishop, 2006). Elle est une fonction non linéaire utilisée par la couche en sortie. Elle permet de transformer un vecteur d'origine de dimension D avec des valeurs réelles arbitraires en un vecteur de probabilité de dimension D avec des valeurs réelles comprises dans la l'intervalle $[0, 1]$. Cette fonction, représentée par l'équation 4.2.6, est considérée comme une généralisation de la fonction sigmoïde et qui est souvent adaptée aux problèmes de classification multi-classes de la régression logistique. Avec $j \in \{1, \ldots, K\}$, $K > 2$, $\text{softmax}(x) \in [0, 1]$ et $x \in \mathbb{R}$, la fonction d'activation Softmax est définie comme suit:

$$\text{softmax}(x) = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}} \tag{4.2.6}$$

Types d'apprentissage

Les algorithmes d'apprentissage automatique peuvent être classés en deux catégories distinctes, à savoir, algorithmes d'apprentissage supervisés ou catégorie d'approches non-supervisées. Bien que certains auteurs classent également d'autres algorithmes dans la catégorie par renforcement, car ces techniques apprennent des données et identifient des modèles dans le but de réagir à un environnement (Alloghani et al., 2020). Nous présentons dans ce qui suit ces deux types d'apprentissage.

Apprentissage supervisé L'apprentissage supervisé est un mode d'apprentissage automatique permettant d'étudier une tâche qui transforme des données d'entrée en données de sortie sur la base d'un échantillon de paires d'entrée-sortie (Van Gerven and Bohte, 2017). Il est considéré comme la méthodologie la plus importante dans les domaines d'apprentissage automatique et profond. Supposons $\mathcal{D}^{(n)} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\} \in (\mathcal{X} \times \mathcal{Y})^{(n)}$ un ensemble de données où n est le nombre d'échantillons disponibles et la paire $(x^{(i)}, y^{(1)})$ est un exemple regroupant la donnée d'entrée avec sa donnée de sortie correspondante. La tâche principale de l'apprentissage supervisé est d'apprendre une fonction $f : \mathcal{X} \to \mathcal{Y}$, à partir des données d'apprentissage étiquetées, qui se rapproche le plus de la relation qu'existe entre une entrée $x^{(i)}$ et son étiquette $y^{(i)}$ correspondante. En d'autres termes, l'apprentissage supervisé consiste à apprendre une correspondance entre un ensemble de variables d'entrée et une variable de sortie correspondante et d'appliquer cette correspondance pour prédire les sorties pour des données inconnues.

Apprentissage non supervisé L'apprentissage non supervisé dans les réseaux de neurones artificiels a le potentiel de redéfinir l'écosystème artificiel. Contrairement à l'apprentissage supervisé, pour l'apprentissage non supervisé, aucune donnée annotée n'est requise et donc l'ensemble de données est limité à $\mathcal{D}^{(n)} = \{(x^{(1)}), \ldots, (x^{(n)})\} \in (\mathcal{X})^{(n)}$. Ce type d'approche d'apprentissage s'efforce de trouver des structures, des dépendances entre les échantillons donnés en entrée; en d'autres termes, la tâche d'apprentissage non supervisé consiste à trouver la représentation la plus appropriée de ces données d'entrée.

À noter que dans notre thèse, la plupart des approches que nous avons développées appartiennent à la catégorie d'apprentissage non supervisé.

Topologie des perceptrons multicouches

L'architecture générale du perceptron multicouche est composée de plusieurs couches à plusieurs nœuds (Figure 4.6). La couche d'entrée assure la distribution des différentes instances à partir de la base d'apprentissage. Cependant, la couche de sortie sert à traiter les informations reçues des couches précédentes et à prédire les classes des instances en entrée. Comme la couche de sortie, les couches

cachées sont des couches de traitement, où la sortie de chaque nœud de la couche précédente du réseau de neurones est reliée, via des connexions synaptiques, avec tous les autres nœuds qui composent la couche suivante. Ainsi, la topologie du réseau de neurones multicouches est uniforme et régulière. Mathématiquement le nombre total des connexions est calculé via la formule suivante:

$$V = \sum_{l=1}^{L-1} N_l \times N_{l+1} + \sum_{l=1}^{L-1} N_{l+1}. \tag{4.2.7}$$

, où L est le nombre de couches du réseau, N_l est le nombre de nœuds dans la couche $l \in \{1, \ldots, L\}$.

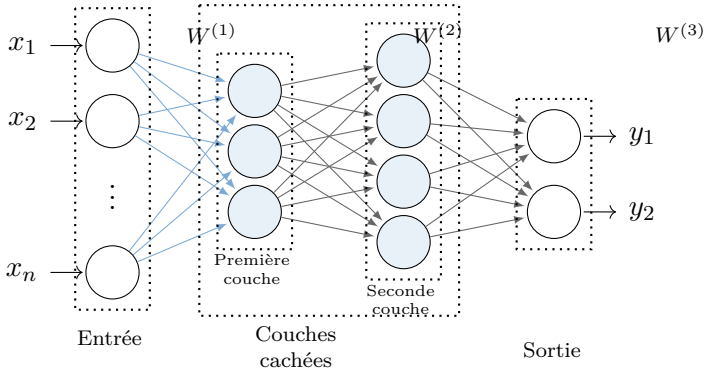


Figure 4.6 – Exemple d'architecture pour un perceptron multicouches

Si nous considérons la matrice des coefficients des poids synaptiques $W^{[l]}$; les valeurs du vecteur de sortie $\hat{Y}$ pour un perceptron multicouche à deux couches cachées (cas de l'exemple précédent) peuvent être déterminées via l'équation suivante:

$$\hat{Y} = \phi^{[3]}(\phi^{[2]}(\phi^{[1]}(XW^{[1]} + b^{[1]})W^{[2]} + b^{[2]})W^{[3]} + b^{[3]}) \tag{4.2.8}$$

, où $X = (x_1, x_2, \ldots, x_n)$ est le vecteur d'entrée, et ϕ est la fonction d'activation. Afin de généraliser l'équation précédente (équation 4.2.8), on appellera $F_W = F^{[1]} \circ \cdots \circ F^{[L]}$ la fonction de transfert associée au réseau global. A ce titre, la prédiction $\hat{y}$ du réseau, par rapport à l'entrée $x \in X$ pourra être notée comme suit:

$$\hat{y} = F_{(W,b)}(x) \tag{4.2.9}$$

En résumé, un perceptron multicouche se constitue par un empilement de trois types de couches:

— *Couche d'entrée*, les neurones de la couche d'entrée reçoivent les données de la base d'apprentissage. La taille de cette couche dépendra du nombre d'attributs supposées décrire le problème à analyser;

— *Couche cachées*, ce sont les couches comprises entre les couches d'entrée et sortie du réseau de neurones. Leur principal objectif est d'effectuer des calculs en transmettant les données de la couche d'entrée vers la couche de sortie. Le choix du nombre de couches appropriées ainsi que le nombre de nœuds pour chaque couche est crucial. En effet, il peut être choisi aléatoirement ou selon un processus d'optimisation, mais qui peut (doit) être ajusté par la suite afin de maximiser les performances pour le problème traité. Chaque neurone $n_j^{[l]}, j \in \{1, \ldots, N_l\}$, possède des connexions, via des poids synaptiques $w_{j,k}^{[l],[l+1]}$, avec chaque neurone $n_k^{[l+1]}, k \in \{1, \ldots, N_{l+1}\}$, de la couche adjacente. Le calcul pour chaque neurone $n_k^{[l+1]}$ est effectué selon l'équation suivante (équation 4.2.10):

$$n_k^{[l+1]} = \phi^{[l]}(\sum_{j=1}^{N_l} n_j^{[l]} \times w_{j,k}^{[l],[l+1]} + b_k^{[l+1]}) \tag{4.2.10}$$

— *Couche de sortie*, elle correspond aux neurones artificiels qui sont associés aux classes de sorties du modèle, où chaque neurone se réfère à une classe cible de celui-ci.

L'apprentissage dans un ANN

L'apprentissage est la phase la plus importante dans les réseaux de neurones. Il s'agit d'une phase de traitement et de calcul durant laquelle le réseau de neurones subi des modifications jusqu'à obtention d'un certain niveau de convergence vers un comportement attendu afin de résoudre la tâche donnée. Il convient à noter que dans cette section, nous nous concentrerons que sur le type d'apprentissage supervisé.

Principe d'apprentissage L'apprentissage est un problème d'optimisation qui consiste en un ajustement d'un ensemble de paramètres $\theta = (W, b)$, où W est la matrice des poids des connexions et b est le vecteur du biais. Le principe consiste à classifier correctement les exemples d'entrée de la base d'apprentissage $\mathcal{D} = \{(x^{(1)}, y^{(1)}), \dots (x^{(m)}, y^{(m)})\}$ avec leurs classes correspondantes en minimisant le risque empirique ou la perte qu'existe entre les classes réelles et les classes prédites des différents exemples de la base d'apprentissage:

$$\text{Perte} = \frac{1}{m}\sum_{i=1}^{m}\varepsilon(y^{(i)}, \hat{y}^{(i)}) \tag{4.2.11}$$

; où ε est la fonction de perte qui mesure la différence entre $y \in \mathcal{D}$ qui représente la vraie valeur et $\hat{y}$ la valeur prédite par le réseau. En littérature, il existe plusieurs types de fonctions de perte, où chaque type présente certaines caractéristiques. Cependant, dans le but de résoudre un problème d'optimisation tout en garantissant une convergence plus rapide, les fonctions de coût qui soient à la fois sous-différentiables et convexes sont les plus utilisées. Parmi les exemples de fonctions de perte les plus connues et les plus utilisées en apprentissage supervisé, nous pouvons citer, d'une manière non exhaustive, l'entropie croisée (CE) en classification (Brescia et al., 2009) (équation 4.2.12); et l'erreur moyenne quadratique (MSE) en régression (Kline and Berardi, 2005) (équation 4.2.13).

$$\varepsilon_{CE} = \begin{cases} -\sum_{i=1}^{m} y^{(i)}\log(\hat{y}^{(i)}) + (1-y^{(i)})(1-\log(1-\hat{y}^{(i)})) & \text{Classification binaire} \\ -\sum_{i=1}^{m} y^{(i)}\log(\hat{y}^{(i)}) & \text{Classification multi-classes} \end{cases} \tag{4.2.12}$$

$$\varepsilon_{MSE} = \frac{1}{m}\sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2 \tag{4.2.13}$$

Donc l'objectif principal de la phase d'apprentissage est de déterminer l'ensemble de paramètres θ de sorte que:

$$(W^*, b^*) \in \underset{\binom{W=\{W^{[1]},\dots,W^{[L]}\}}{b=\{b^{[1]},\dots,b^{[L]}\}}}{\operatorname{argmin}} \frac{1}{m}\sum_{i=1}^{m}\varepsilon(y^{(i)}, F_{(W,b)}(x^{(i)})) \tag{4.2.14}$$

En apprentissage statistique et afin d'optimiser les paramètres de l'équation 4.2.14, l'un des algorithmes les plus efficaces est la méthode de descente de gradient (GD) par rétropropagation présentée dans (Rumelhart et al., 1986). Il est important à noter que cette méthode a été proposé pour la première fois dans (Werbos, 1974). La descente de gradient est une méthode itérative permettant de minimiser l'écart entre les valeurs réelles et à celles prédites; tout en assurant une convergence vers un optimum local (pas forcement global).
Le procédé d'apprentissage consiste en deux étapes:

1. Une étape de propagation (*forward propagation*) de l'erreur vers l'avant dans laquelle le réseau de neurones utilisera les paramètres $\theta(t)$ actuels afin de prédire la valeur de sortie $\hat{y}^{(i)}$ pour l'entrée $x^{(i)}$. L'erreur moyenne commise, à chaque itération, par le réseau de neurone sur toute la base d'apprentissage est calculée en utilisant une des formules de coût présentées précédemment, à savoir ε_{CE} définie par l'équation 4.2.12, la fonction ε_{MSE} via l'équation 4.2.13 ou tout autre formule définie en littérature.

2. La seconde étape consiste à la rétropropagation (*back-propagation*) de l'erreur. L'objectif est de refaire à « contre sens » le processus de propagation de l'erreur. L'idée est de mettre à jour les paramètres $\theta(t+1)$ de l'avant dernière couche jusqu'à l'arrivé à la première couche cachée du réseau de neurones. Cette mise à jour est effectuée par l'estimation de l'erreur de calcul pour toutes les couches cachées en utilisant le taux de sensibilité calculé pour la couche de sortie à la fin de la phase de propagation. Cette descente de gradient est assurée par un ensemble de dérivés partielles, où pour chaque itération, nous avons:

$$\begin{aligned} \frac{\delta\varepsilon}{\varepsilon w(t)} = \Delta w, w(t+1) = w(t) - \alpha w; \\ \frac{\delta\varepsilon}{\varepsilon b(t)} = \Delta b, w(t+1) = w(t) - \alpha b \end{aligned} \tag{4.2.15}$$

, avec α représente le taux d'apprentissage.
Ce qui se réécrit:

$$w(t+1) = w(t) - \alpha \frac{1}{m} \sum_{i=1}^{m} \frac{\delta\varepsilon(y^{(i)}, F_{(W,b)}(x^{(i)}))}{\delta w(t)} \tag{4.2.16}$$

$$b(t+1) = b(t) - \alpha \frac{1}{m} \sum_{i=1}^{m} \frac{\delta\varepsilon(y^{(i)}, F_{(W,b)}(x^{(i)}))}{\delta b(t)} \tag{4.2.17}$$

Algorithme de descente de gradient Avant que le processus itératif d'entraînement, représenté par l'algorithme 2, ne soit déclenché, tous les paramètres du réseau doivent être initialisés soit aléatoirement (de la ligne 1 à la ligne 4), soit à l'aide de certaines fonctions de distributions. En effet, l'utilisation de telles distributions rend l'ajustement du réseau vers les valeurs optimales plus rapide. Il faut savoir que la loi normale, la loi uniforme, et la distribution définie dans (Glorot and Bengio, 2010) sont parmi les types les plus utilisés dans les réseaux de neurones.

Algorithme 2 : Algorithme de descente de gradient

Input : $\mathcal{D} = \{x^{(m)}, y^{(m)}\}_{m=1}^{M}$, *iter_max*, *err_max*

`/* Initialisation aléatoire des paramètres du réseau */`

1 **foreach** $l \in \{1, 2, \ldots, L-1\}$ **do**
2 **foreach** $j \in \{1, 2, \ldots, N_l\}$ **do**
3 **foreach** $k \in \{1, 2, \ldots, N_{l+1}\}$ **do**
4 $(w_{j,k}^{[l][l+1]}, b_{j,k}^{[l]}) \leftarrow A \sim \mathcal{U}(-1, 1)$;
5 $t \leftarrow 0$;
6 **while** $\varepsilon \geq err\_max$ *and* $t \leq iter\_max$ **do**
 `/* Propagation de l'erreur */`
7 **foreach** $(x^{(i)}, y^{(i)})_{i=1}^{m} \in \mathcal{D}$ **do**
8 $\hat{y}^{(i)} \leftarrow F_{(W,b)}(x^{(i)})$ ▷ Propagation de l'exemple $x^{(i)}$
9 $\epsilon^{(i)} \leftarrow \varepsilon(\hat{y}^{(i)} - y^{(i)})$ ▷ Calcul de l'erreur de l'exemple $(x^{(i)}, y^{(i)})$
 `/* Rétropropagation de l'erreur */`
10 $\epsilon \leftarrow \frac{1}{m} \sum_{i=1}^{m} \epsilon^{(i)}$ ▷ Calcul de l'erreur de la base d'apprentissage
11 $(\Delta w, \Delta b) \leftarrow (\frac{\delta\varepsilon}{\delta w(t)}, \frac{\delta\varepsilon}{\delta b(t)})$
12 $\theta(t+1) \leftarrow \theta(t) - \alpha\Delta\theta$ ▷ Mettre à jour les paramètres

La première partie de la deuxième phase, pour cet algorithme, consiste en une propagation de l'erreur pour l'ensemble des exemples de la base de test. Cette propagation avant est effectuée via l'application de la fonction $F_{(W,b)}$ à l'exemple $x^{(i)}$ (ligne 8). Une fois que tous les exemples ont été

prédits, l'erreur moyenne de la base d'apprentissage sera calculée (ligne 10) Une fois terminer, la rétropropagation consiste en une mise à jour basée sur des informations de gradient local avec un point initial aléatoire α (comme illustré par la ligne 12). Ce processus est répété jusqu'à un nombre d'itération définit soit atteint où l'algorithme de descente de gradient se converge vers un optimale local.

D'autres variants d'algorithme L'algorithme de rétropropagation (Algorithme 2) présenté précédemment est connu sous le nom de la descente de gradient *Batch*. Il est la méthode d'apprentissage la plus utilisée dans les réseaux de neurones. Empiriquement, il a été prouvé que pour ce type d'algorithme, la convergence vers l'optimum est plutôt directe. Néanmoins, cette méthode n'est pas optimale dans le cas d'une base d'apprentissage volumineuse. En effet, pour chaque itération et pour chaque paramètre plusieurs dérivées seront calculées. Pour faire face, d'autres variantes de cet algorithme qui s'adaptent mieux à la volumétrie de la base d'apprentissage ainsi qu'à la parallélisation des calculs, peuvent être utilisés, à savoir:

— *La descente de gradient stochastique*: Dans cet algorithme, et au contraire de l'algorithme précédent, la descente du gradient se fait pour chaque exemple $x^{(i)}$ de la base d'apprentissage. En effet, au lieu de propager l'erreur pour tous les exemples de la base d'apprentissage, la modification des paramètres du modèle s'effectue pour un seul exemple $x^{(i)}$ jusqu'à l'obtention d'une prédiction $\hat{y}^{(i)}$ qui soit la plus proche possible de la vraie valeur $y^{(i)}$. Par la suite, ce même processus est itéré jusqu'à avoir parcouru l'ensemble des exemples de la base d'apprentissage. De ce fait, les équations précédentes (équation 4.2.16 et équation 4.2.17) seront réécrites comme suit:

$$w(t+1) = w(t) - \alpha \frac{\delta \varepsilon(y^{(i)}, F_{(W,b)}(x^{(i)}))}{\delta w(t)} \tag{4.2.18}$$

$$b(t+1) = b(t) - \alpha \frac{\delta \varepsilon(y^{(i)}, F_{(W,b)}(x^{(i)}))}{\delta b(t)} \tag{4.2.19}$$

— *La descente de gradient mini-batch*: cette troisième alternative est considérée comme un croisement entre les deux premiers algorithmes. Au lieu de prendre les deux extimités, à savoir tous les exemples de la base d'apprentissage pour une descente de gradient *batch*, ou un seul exemple pour une descente de gradient *stochastique*, à chaque itération, un certain nombre $batch \in \mathbb{N}$ d'exemples sera considéré seulement. En plus de la modification des boucles ainsi que leurs critères d'arrêts de l'algorithme 2, les équations précédentes (équation 4.2.16 et équation 4.2.17) seront modifiées comme suit:

$$w(t+1) = w(t) - \alpha \frac{1}{batch} \sum_{z=1}^{batch} \frac{\delta \varepsilon(y^{(z)}, F_{(W,b)}(x^{(z)}))}{\delta w(t)} \tag{4.2.20}$$

$$b(t+1) = b(t) - \alpha \frac{1}{batch} \sum_{z=1}^{batch} \frac{\delta \varepsilon(y^{(z)}, F_{(W,b)}(x^{(z)}))}{\delta b(t)} \tag{4.2.21}$$

À noter que la taille du mini-batch doit être suffisamment significatif pour représenter les statistiques de la distribution de l'ensemble d'entraînement.

4.3 Apprentissage profond

L'apprentissage profond est considéré comme une étape révolutionnaire dans le domaine de l'intelligence artificielle. En effet, les réseaux de neurones profonds ont montré une très grande efficacité à résoudre plusieurs tâches complexes par rapport aux réseaux de neurones artificiels traditionnels. Le principe général des réseaux de neurones profonds est d'effectuer des transformations hiérarchiques profondes sur les données de la base d'apprentissage. Cela est notamment grâce aux nombreuses

couches en éléments neuronaux qui composent ce type de réseaux; mais aussi, grâce aux progrès technologiques importants enregistrés en matière de puissance de calcul (notamment avec l'apparition des GPUs *Graphics Processing Unit*). Historiquement, le premier réseau de neurones était le perceptron profond, qui représente généralement un perceptron multicouche avec plus de deux couches cachées (Golovko, 2017).

Ces dernières années, plusieurs architectures ont émergé dans les réseaux de neurones. Elles sont répertoriées en quatre catégories, à savoir: réseaux de neurones convolutifs (ou *Convolutional Neural Networks (CNN)*), les réseaux de neurones récurrents (ou *Recurrent Neural Networks (RNN)*), réseaux de croyances profonds (ou *Deep Belief Neural Networks (DBN)*) et les auto-encodeurs (ou *Autoencoders*).

4.3.1 Réseaux de neurones convolutifs

L'architecture des réseaux de neurones convolutifs (ou *Convolutional Neural Networks (CNN)*) est une forme particulière des réseaux ANN, inspirée de l'organisation du cortex visuel des animaux; c'est-à-dire la partie de cerveau responsable du traitement des informations provenant de l'œil. Dans les années 1960, les auteurs dans (Hubel and Wiesel, 1962) ont proposé un concept appelé champs réceptifs. Ils ont étudié les neurones chargés de l'orientation sélective sensible locale dans le système visuel du chat lorsqu'il observe quelques formes de base. Ils ont découvert que les arrangements complexes des cellules étaient contenus dans le cortex visuel animal chargé de la détection de la lumière dans les sous-régions du champ visuel qui se chevauchent et ont ensuite proposé le réseau neuronal à convolution.

Le réseau CNN est un algorithme de reconnaissance efficace qui est largement utilisé dans la reconnaissance de formes et le traitement d'images. Le premier CNN utilisé pour la détection d'objets remonte à 1989 (LeCun et al., 1989), qui fut plus tard appelé LeNet-5 (Lecun et al., 1998). Les auteurs ont exploité la méthode de rétropropagation afin d'accomplir une tâche de classification supervisée. Depuis, les architectures des CNN actuels sont assez similaires à celles du LeNet-5 mais plus complexes. Cette complexité est due au développement rapide des techniques de calcul accéléré par GPU, qui ont été exploitées pour entraîner plus efficacement les CNN; ce qui a encouragé la communauté de l'intelligence artificielle à proposer d'autres variantes optimisées de l'architecture CNN.

Comme le montre la figure 4.7, le CNN est un réseau de neurones multicouche qui se compose de trois types de couches différentes, à savoir des couches de convolution, des couches de sous-échantillonnage et une couche entièrement connectée (Goodfellow et al., 2016; Krizhevsky et al., 2017). Les couches de convolution et de sous-échantillonnage sont connectées alternativement et forment la partie centrale du réseau.

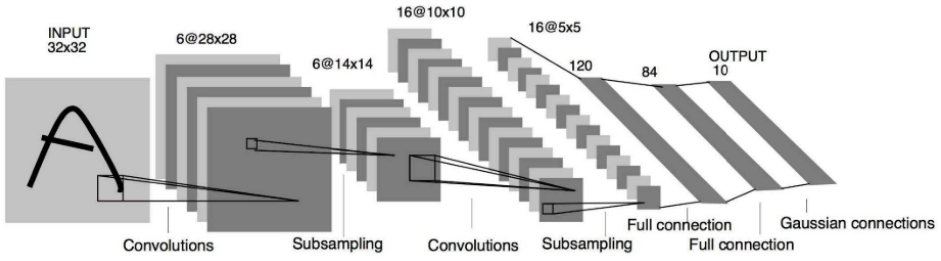


Figure 4.7 – Schéma conceptuelle du réseau de neurones convolutif (Lecun et al., 1998)

Couche de convolution

La couche de convolution est le composant clé des réseaux de neurones convolutifs et l'élément principal du bloc convolutif. L'objectif de cette couche est d'identifier implicitement la présence d'un ensemble de caractéristiques pertinentes dans l'image d'entrée à l'aide d'un opérateur de convolution durant la phase d'apprentissage.

Couche de sous-échantillonnage

Une fois que l'ensemble des caractéristiques pertinentes sont extraites, la relation de position entre elles peut être déterminée. Une couche de sous-échantillonnage est essentielle pour la cartographie de ces caractéristiques. La couche de sous-échantillonnage, également appelée couche de pooling, consiste à réduire la dimensionnalité des couches de convolution ou de ces cartes d'activation résultantes. Elle permet, d'un côté, de réduire le nombre de paramètres et les exigences de calcul. De l'autre côté, elle permet de réduire les cartes d'activation de sorte que le réseau devienne plus efficace, ce qui minimisera la probabilité d'un sur-apprentissage (Jarrett et al., 2009).

Couche entièrement connectée

Les neurones de cette couche sont entièrement connectés aux neurones de la couche précédente (généralement deux à trois couches). L'ensemble de ces couches ont la même structure qu'un perceptron multicouches qui prend le vecteur de caractéristiques en entrée et le transforme en classes de sortie finales ou en scores de classe afin d'apprendre les combinaisons non linéaires entre les caractéristiques extraites par les couches de convolution. Autrement dit, cette partie entièrement connectée est utilisée pour associer les différentes caractéristiques pertinentes de l'image d'entrée afin d'en déduire la classe correcte. En classification supervisée, la dernière couche est utilisée pour la prédiction en se basant sur la fonction d'activation Softmax.

Pour plus de détails sur les réseaux de neurones convolutifs, nous orientons le lecteur vers (Goodfellow et al., 2016).

4.3.2 Réseaux de neurones récurrents

Les réseaux de neurones récurrents (ou *Recurrent neural networks (RNN)*) (Rumelhart et al., 1986) sont une catégorie de réseaux de neurones destinés à traiter des données séquentielles. Par analogie aux réseaux convolutifs spécialisés dans le traitement d'une grille de valeurs X telle qu'une image, les réseaux de neurones récurrents sont des réseaux spécialisés dans le traitement d'une séquence de valeurs $x^{(1)}, \ldots, x^{(r)}$. Un réseau de neurones récurrent est un modèle de séquence neuronale qui atteint des performances de pointe dans des tâches importantes, notamment la modélisation du langage (Mikolov and Zweig, 2012), la reconnaissance vocale (Graves et al., 2013) et la traduction automatique (Kalchbrenner and Blunsom, 2013).

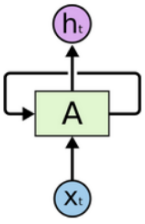


Figure 4.8 – Structure en boucle dans le RNN [24]

Un RNN est un type de réseau doté de boucles qui leur permettent de conserver les informations du passé dans le modèle de réseau. Sur la figure 4.8, le carré du milieu désigne un réseau de neurones, qui prend à l'instant t l'entrée x_t et donne sa valeur correspondante en sortie h_t au même instant t. La boucle représentée dans cette structure lui permet d'utiliser les informations relatives aux instants de temps précédents pour produire une sortie pour l'instant t actuel. De cette façon, nous pouvons dire que la décision prise à l'instant $t-1$ affecte la décision à prendre à l'instant suivante t. Ainsi, la réaction du réseau aux nouvelles données dépend de l'entrée actuelle ainsi que de la sortie des données précédentes. Le calcul de la sortie RNN est basé sur le calcul itératif de la sortie des deux équations

24. Understanding LSTM networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/

suivantes:

$$h_t = H(W_{xh}x_t + W_{hh}x_{t-1} + b_h) \tag{4.3.1}$$

et

$$y_t = (W_{hy}h_t + b_y) \tag{4.3.2}$$

où x_t est la séquence d'entrée à l'instant actuel t, y_t est la séquence de sortie à l'instant t et h représente la séquence de vecteurs cachés de tous les instants passés (du premier instant du temps jusqu'à l'instant t). W et b représentent, respectivement, les matrices de poids et les biais. Enfin, H est la fonction d'activation utilisée pour la couche cachée.

Les réseaux de neurones récurrents, sont un type de réseau largement utilisé dans le domaine de l'apprentissage profond (Schmidhuber, 2015). En revanche, et malgré leurs avantages en traitement des séquences temporelles, ils peuvent causer un problème étroitement lié à celui de la dégradation du gradient. Ce problème se produit lors de la multiplication récursive des matrices de poids dont plusieurs entrées sont supérieures à un dans la phase d'apprentissage. Au fil des itérations, le modèle n'est plus en mesure de reconnaître les associations entre les séquences distantes, et les gradients de ces séquences ont tendance de converger vers la valeur 0 (Goodfellow et al., 2016). Afin de résoudre ces limitations, d'autres variantes ont été développées à savoir les réseaux de mémoire à long-court terme LSTM (Hochreiter and Schmidhuber, 1997) (qui sera discuté en section 5.4.2) et réseau de neurones récurrents à portes GRU (Cho et al., 2014) (qui sera discuté en section 5.4.2).

4.4 Les modèles génératifs d'apprentissage profond

L'objectif des modèles génératifs est d'apprendre un modèle de densité de probabilité aussi similaire que possible à la distribution génératrice des données réelles. Ce type de modèle peut avoir des applications directes telles que la compression de données ou la génération de données. Il est également raisonnable de supposer que pour être performant et bien fonctionner, un tel modèle doit construire des représentations riches et abstraites des données. Dans cette section, nous nous intéressons à deux modèles génératifs que nous avons utilisé pour la prédiction de la QoS pour les services Web; à savoir les auto-encodeurs et les réseaux antagonistes génératifs.

4.4.1 Auto-encodeurs

Les auto-encodeurs (Hinton, 2006; Vincent et al., 2008) sont des réseaux de neurones artificiels symétriques, qui apprennent la fonction identité sous contrainte en utilisant une approche d'apprentissage non supervisé. L'idée de base d'un auto-encodeur est de copier les données de la couche d'entrée vers la couche de sortie en passant par une ou plusieurs couches cachées. En d'autres termes, un auto-encodeur est un réseau de neurones qui apprend un modèle afin de découvrir la structure cachée dans la distribution d'entrée en reproduisant, en sortie, la même information d'entrée avec un minimum de déformation possible.

Techniquement, une première partie de l'auto-encodeur, via une ou plusieurs couches cachées, codera le vecteur d'entrée en un autre vecteur de dimension réduite. Il s'agit d'une approximation de la structure cachée, communément appelée représentation latente. Par la suite, la deuxième partie de l'auto-encodeur décodera le vecteur latent afin de récupérer une approximation de l'entrée originale. La dissimilarité entre le vecteur d'entrée et celui de la sortie peut être mesurée par une fonction de perte.

Actuellement, les auto-encodeurs sont largement utilisés dans plusieurs domaines (détection d'anomalies, débruitage d'images, etc.). Dans cette partie, nous présentons l'architecture de base de ce type de réseaux de neurones ainsi que les différentes architectures existantes.

Principes de base des auto-encodeurs

Un auto-encodeur est composée, essentiellement, par deux opérateurs qui représentent deux sous réseaux de neurones distincts, à savoir:

1. **Encodeur**, il permet de transformer le vecteur d'entrée x en un vecteur latent de faible dimension $z = f(x)$. Comme ce dernier est de faible dimension, l'encodeur ne va apprendre, à travers une ou plusieurs couches cachées, que les caractéristiques les plus significatives sur la distribution des données d'entrée.
2. **Décodeur**, contrairement à l'encodeur, le décodeur cherche à restituer le même vecteur d'entrée à partir du vecteur latent, c'est-à-dire $g(z) = \hat{x}$. Bien que le vecteur latent ait une faible dimension, le décodeur doit être en mesure, via une ou plusieurs couches cachées, de récupérer les données d'entrée.

En règle générale, l'encodeur et le décodeur doivent être des fonctions non linéaires. L'objectif la fonction de codage est de construire les facteurs latents du vecteur d'entrée x. Cependant, l'objectif de la deuxième fonction est de réduire au maximum la déformation du vecteur de sortie $\hat{x}$ par rapport au vecteur original. Dans un auto-encodeur, les facteurs latents sont représentés par une couche unique compacte et significative, dont la taille doit être inférieure à la couche d'entrée. Cette dimension représente le nombre de caractéristiques latentes qu'elle peut représenter. Il est important de mentionner que la dimension de z doit être généralement beaucoup plus petite que la dimension du vecteur d'entrée pour des raisons d'efficacité et afin de contraindre le code latent à n'apprendre que les facteurs latents les plus significatifs de la distribution des données d'entrée (Goodfellow et al., 2016). Dans le cas où la dimension de la couche des facteurs latents z est significativement plus grande que l'entrée x, l'auto-encodeur a tendance à mémoriser les données d'entrée et se comportant ainsi essentiellement comme une fonction identité [25].

Architecture de l'auto-encodeur

Comme les autres types de réseaux de neurones, l'auto-encodeur cherche à minimiser la différence entre les données d'entrée et les données de sortie durant la phase d'apprentissage tel que, $h_{W,b}(x) = g_{(W_g,b_g)}(f_{(W_f,b_f)}(x)) \approx x$. La figure 4.9 illustre en détail l'architecture de base d'un auto-encodeur profond. L'encodeur est la fonction qui compacte l'entrée x, en un vecteur latent de faible dimension z. Ce dernier représente les caractéristiques importantes de la distribution des données en entrée. Le décodeur, quant à lui, essaie de récupérer l'entrée originale à partir du vecteur latent sous la forme $\hat{x}$.

En effet, l'architecture de base d'un auto-encodeur consiste en deux « sous réseaux »:

— L'encodeur est représenté par un réseau neuronal artificiel dont les couches sont fortement connectées. Il représente la fonction d'encodage f_e qui prend en entrée, le vecteur $x \in \mathbb{R}^d$ est le transforme en un vecteur latent de faible dimension $z \in \mathbb{R}^k$ $(z = f(x))$;

— Le décodeur représente l'opération inverse de l'opération de l'encodeur. Il s'agit d'un réseau de neurones artificiels dont les couches sont fortement connectées. Il est considéré comme une fonction de décodage g qui prend en entrée la représentation latente obtenue précédemment z, et la transforme à nouveau en un vecteur reconstruit $\hat{x} \in \mathbb{R}^d$, c'est-à-dire $\hat{x} = g(z)$.

— Finalement, la partie facteurs latents représentée par couche unique de neurones artificiels dont la taille doit être inférieure à celle de la couche d'entrée. Elle représente à la fois la couche de sortie du sous réseau de neurones de l'encodeur et la couche d'entrée du sous réseau de neurones du décodeur.

Ainsi, la sortie d'un auto-encoder pourra être notée comme suit:

$$\hat{x} = g_{(W_g,b_g)}(f_{(W_f,b_f)}(x)) \tag{4.4.1}$$

25. Dans ce cas d'autres contraintes peuvent être appliquées afin d'éviter ce problème.

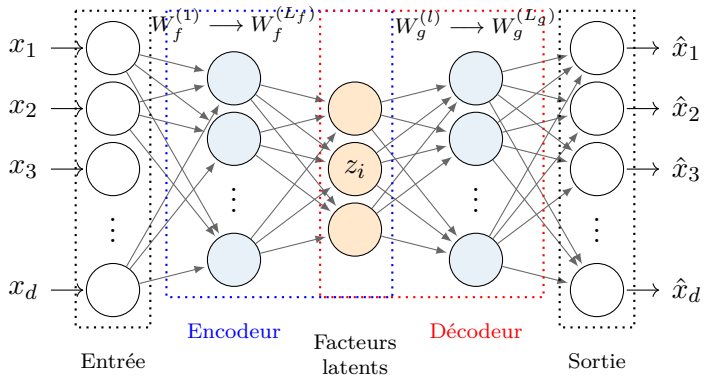


Figure 4.9 – Architecture de base d'un autoencodeur

En termes d'apprentissage, l'auto-encodeur est entraîné afin de minimiser la dissimilarité entre l'entrée x, et la sortie qui est l'entrée récupérée $\hat{x}$

$$\underset{\substack{W_f, W_g \\ b_f, b_g}}{\operatorname{argmin}} \sum_{i=1}^{m} \varepsilon(x^{(i)}, g_{(W_g, b_g)}(f_{(W_f, b_f)}(x^{(i)}))) \tag{4.4.2}$$

, où m est la taille de la base d'apprentissage. ($W_f = \{W_f^{[1]}, \ldots, W_f^{[L_f]}\}$, $W_g = \{W_g^{[1]}, \ldots, W_g^{[L_g]}\}$) et ($b_f = \{b_f^{[1]}, \ldots, b_f^{[L_f]}\}$, $b_g = \{b_g^{[1]}, \ldots, b_g^{[L_g]}\}$) sont les matrices des poids et vecteurs des biais pour respectivement, l'encodeur et le décodeur. ε est la mesure de dissemblance entre l'entrée x et la sortie $\hat{x}$. Comme le montre l'équation 4.4.3 suivante, l'erreur quadratique moyenne (MSE) est un exemple d'une telle fonction de perte:

$$\varepsilon_{\text{reconstruction}} = \varepsilon_{MSE}(x, \hat{x}) = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)} - \hat{x}^{(i)})^2 \tag{4.4.3}$$

Il est à noter que l'erreur quadratique moyenne est souvent utilisée si l'entrée est évaluée en valeur réelle. Dans le cas contraire, d'autres fonctions peuvent être suggérées telle que l'entropie croisée.

Autres types d'auto-encodeurs

Le principe de base des auto-encodeurs est de trouver le moyen le plus efficace de comment compresser l'entrée afin de pouvoir la recréer plus tard. L'auto-encodeur compresse l'entrée en un petit nombre de variables cachées (variables latentes), qui ont été transmises au décodeur pour restaurer l'entrée. Cependant, il existe plusieurs variantes d'auto-encodeurs qui ont été élaborés afin d'améliorer l'architecture de base initiale tels que: les auto-encodeurs débruiteurs, les auto-encodeurs épars, les auto-encodeurs contractifs, les auto-encodeurs convolutionnels.

Auto-encodeur débruiteurs Un auto-encodeur de débruitage (*denoising autoencoder*) (Vincent et al., 2010) est une version partiellement variante de l'auto-encodeur de base présenté précédemment. Il a pratiquement la même structure que l'auto-codeur, mais ce type de réseaux inflige aux données d'entrée originales un bruit qui soit aléatoire. Par la suite, le réseau sera entraîné, via les données bruitées, à reconstruire l'entrée originale non déformée même si elle a été partiellement corrompue au début. De ce fait, la sortie de l'auto-encodeur de débruitage est déterminée comme suit:

$$\hat{x} = g(f(x + bruit)) \tag{4.4.4}$$

Cependant, l'équation 4.4.2 ne sera pas changée, car l'erreur à rétro-propager est calculée entre la sortie prédite $\hat{x}$ par le modèle et l'entrée originale x et non pas la version bruitée, comme illustré par la figure 4.10.

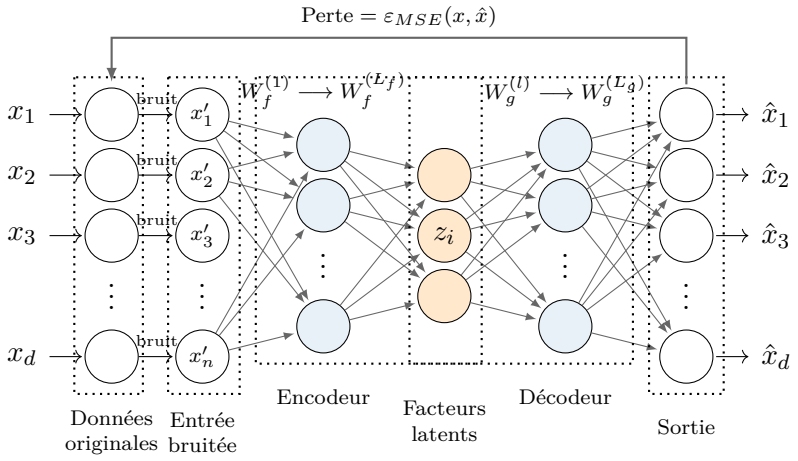


Figure 4.10 – Auto-encodeur de débruitage

Dans ce cas, il a été prouvé dans (Alain and Bengio, 2014) que l'apprentissage de l'auto-encodeur force les fonctions d'encodage f et de décodage g à apprendre implicitement la structure des données d'entrée originales.

Auto-encodeur épars (ou *sparse autoencoder*) (Ng, 2011) représente une importante variante des auto-encodeurs standards. Cette version s'est inspirée du fonctionnement du cerveau humain qui arrive à contrôler l'ensemble des 10^{10} neurones avec, à peu près, leurs 10^{14} synapses. L'une des principales caractéristiques qui permet au cerveau d'agir ainsi est la rareté de l'activation neuronale : à tout moment, seule une très petite fraction des neurones du cerveau est active. En effet, le cerveau humain est, heureusement d'ailleurs, dans l'incapacité d'être un modèle dense où tous les neurones sont actifs en même temps. Il s'avère que cette rareté des activations est une propriété souhaitable non seulement pour le cerveau, pour lequel elle est vitale; mais aussi pour les modèles artificiels. En effet plus le réseau est dense plus l'interprétation des résultats devienne très difficile.

En d'autres termes, le fait que le réseau de neurones contient plusieurs éléments inactifs, cela laisse espérer que des neurones distincts et uniques sont responsables des différentes propriétés « utiles » extraites des données d'entrée. Contrairement à la situation où un grand groupe de neurones est immédiatement responsable d'une propriété, il devient très difficile d'interpréter leurs valeurs, et le modèle s'avère trop fragile.

En littérature, plusieurs travaux (Lee et al., 2007; Ranzato et al., 2007) ont démontré que la parcimonie effectuée sur un sous ensemble de neurones des couches cachées présente certains avantages en termes de robustesse et d'amélioration de la classification dans les espaces à haute dimension.

Formellement, considérons $\hat{\rho}_j$ la moyenne des activations pour tous les neurones qui composent la couche cachée j sur m exemples d'apprentissage:

$$\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^{m} (h^{[j]}(x^{(i)}))$$

, où h désigne la fonction d'activation de la couche j.

Concrètement, pour construire un auto-encoder épars, il faut inciter un nombre déterminé de neurones de la couche j à être inactifs (c'est-à-dire leur valeur vaut 0). Pour se faire, le paramètre $\hat{\rho}_j$ doit se rapprocher de la valeur nulle. Ce qui en résulte que cette méthode implique la contrainte suivante:

$$\hat{\rho}_j = \rho$$

, où ρ est une constante définie par l'utilisateur, représentant le taux de parcimonie à appliquer aux couches cachées de l'auto-encodeur. Plus ce paramètre se rapproche de la valeur zéro, plus les activations des neurones cachés se rapprochent aussi de zéro, ce qui implique que le nombre de neurones inactifs serait plus élevé.

Pour y parvenir, un terme de pénalité supplémentaire doit être ajouter à la fonction objectif définie précédemment pour un auto-encodeur de base (voir équation 4.4.3). Ce qui revient à écrire (Bengio et al., 2013):

$$\varepsilon_{sparse}(x, \hat{x}) = \varepsilon_{\text{reconstruction}}(x, \hat{x}) + \beta \sum_{l=1}^{L} \sum_{j=1}^{N_l} KL(\rho \| \hat{\rho}_j) \tag{4.4.5}$$

, où N_l est le nombre de neurones dans la couche implicite l, β un paramètre qui contrôle le poids, ou le degré d'importance, de la pénalité et KL une pénalité de parcimonie qui représente une mesure de dissimilarité entre de distribution de probabilités, dans notre cas ρ et $\hat{\rho}_j$ connue sous le nom de Kullback–Leibler (KL) divergence (Youssef et al., 2016) ou encore, entropie relative. L'expression mathématique de la divergence KL est la suivante:

$$KL(\rho \| \hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j} \tag{4.4.6}$$

Lorsque la somme $\hat{\rho}_j$ est égale avec le paramètre ρ, la valeur de la divergence KL est égale à 0 ; sinon, la valeur de la divergence KL augmente progressivement avec l'écart.

Auto-encodeur contractuels L'auto-encodeur contractif (Rifai et al., 2011) est un type d'auto-encodeur où une régularisation explicite est ajoutée à la fonction objectif. Cet ajustement permet de pénaliser la sensibilité du modèle aux légères variations des valeurs d'entrées $x^{(i)}$. En effet, pour que le modèle ait un comportement robuste sur la représentation $g(f(x))$ obtenue pour un vecteur d'entrée x, il est proposé d'ajouter un terme de régularisation qui correspond à la norme de Frobenius de la matrice Jacobienne. Ce qui modifie l'équation 4.4.3 comme suit:

$$\varepsilon_{sparse}(x, \hat{x}) = \varepsilon_{\text{reconstruction}}(x, \hat{x}) + \lambda \| \nabla_x h \|^2 \tag{4.4.7}$$

, ou ce terme n'est que la norme au carré du gradient de la représentation cachée par rapport à l'entrée x. Par conséquent, la perte globale minimisera la variation de la couche cachée compte tenu de la variation de l'entrée.

4.4.2 Les réseaux antagonistes génératifs

Dans notre vie quotidienne, affronter un adversaire, que ce soit dans des jeux ou des compétitions, nous permet d'être plus performants. Cette pensée antagoniste existe aussi dans le domaine de l'IA. Dans le domaine de l'apprentissage profond, le concept de confrontation est traité de deux manières différentes. La première manière consiste à traiter les vulnérabilités de l'apprentissage profond pour rendre l'apprentissage plus robuste. Tandis que la deuxième tendance considère que le rôle de l'adversaire est d'aider à former un modèle (réseau neuronal) tout en étant en compétition avec lui, ce qui crée un environnement auto-stimulant pour le processus d'apprentissage. Ce domaine d'intérêt est connu sous le nom de réseau antagonistes.

4.4.3 Principes et définition

Les GAN pour Generative Adversarial Networks (Goodfellow et al., 2014) sont des modèles génératifs basés sur les réseaux de neurones. Une modélisation générative est considérée comme une tâche d'apprentissage non supervisé en apprentissage automatique. L'idée principale du GAN est inspirée de la notion de l'équilibre de Nash en théorie des jeux (Creswell et al., 2018). Lorsque deux joueurs ou plus s'affrontent, un équilibre peut être obtenu si chaque joueur est capable d'effectuer le meilleur mouvement possible dans le jeu, en tenant compte des mouvements de l'autre joueur.

Dans ce contexte, les auteurs dans (Goodfellow et al., 2014) ont proposé un jeu à deux joueurs où chaque joueur est représenté par un réseau de neurones artificiels. Le premier joueur est appelé générateur (noté $\mathcal{G}$) vise à générer des images à partir d'un bruit aléatoire tout en respectant la distribution d'une base de données donnée d'images réelles. Le second est un arbitre appelé discriminateur (noté $\mathcal{D}$). Ce dernier a pour objectif de déterminer correctement si une image donnée est une image réelle

(issue de l'ensemble d'apprentissage) ou une image fausse générée par le générateur. Une illustration de l'architecture d'un réseau GAN est donnée par la figure 4.11.

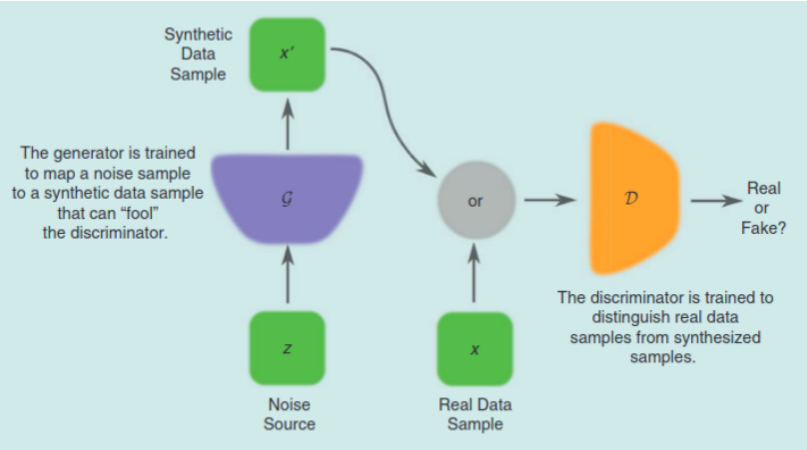


Figure 4.11 – Structure générale contradictoire génératif Creswell et al. (2018)

Pour gagner la partie, les deux joueurs $\mathcal{D}$ et $\mathcal{G}$ doivent constamment optimiser leurs paramètres pour améliorer la capacité de génération (respectivement de discrimination) du générateur (respectivement du discriminateur). Dans le cas optimal, le critère d'arrêt du processus d'optimisation est d'atteindre un équilibre de Nash entre les deux participants.

4.4.4 L'apprentissage dans les GANs

Comme tout réseau de neurones artificiels classique, la rétropropagation est utilisée lors de la phase d'apprentissage des paramètres du réseau génératif, mais le fait qu'il y ait deux réseaux rend son utilisation légèrement différente. En effet, la phase d'apprentissage implique, d'un côté, de trouver les paramètres optimaux pour le discriminateur qui maximisent sa précision de classification. De l'autre côté, de trouver les paramètres optimaux d'un générateur qui déroute au maximum le discriminateur. Plus précisément, les fonctions de perte utilisées et le nombre d'itérations sont deux paramètres majeurs où les GAN diffèrent. Pour les GANs, la fonction de perte utilisée ne sera rien de plus qu'une perte binaire ordinaire d'entropie croisée (Youssef et al., 2016). D'autres fonctions de perte peuvent être utilisées, telles que la distance de Wasserstein (Arjovsky et al., 2017), l'écart moyen maximum (Dziugaite et al., 2015), etc.

Concernant le coût de l'apprentissage, il est évalué à l'aide d'une fonction objectif $V(\mathcal{G}_\theta, \mathcal{D}_\omega)$ (avec ω est l'ensemble des paramètres du discriminateur; tandis que, θ est celui du générateur), qui dépend à la fois du générateur et du discriminateur. L'apprentissage consiste que le discriminateur minimise $V(\mathcal{G}_\theta, \mathcal{D}_\omega)$ sur ω à θ fixé et le générateur maximise $V(\mathcal{G}_\theta, \mathcal{D}_\omega)$ sur θ à ω fixé.:

$$\max_{\mathcal{D}} \min_{\mathcal{G}} V(\mathcal{G}_\theta, \mathcal{D}_\omega) \tag{4.4.8}$$

avec

$$\begin{aligned} V(\mathcal{G}_\theta, \mathcal{D}_\omega) &= \mathbb{E}_{p_{data}(x)} log(\mathcal{D}_\omega(x)) + \mathbb{E}_{z \tilde{} Z} log(1 - \mathcal{D}_\omega(\mathcal{G}_\theta(z)) \\ &= \mathbb{E}_{p_{data}(x)} log(\mathcal{D}(x)) + \mathbb{E}_{p_g(x)} log(1 - \mathcal{D}(x)) \end{aligned} \tag{4.4.9}$$

Durant l'apprentissage, les paramètres d'un modèle sont mis à jour, tandis que les paramètres de l'autre sont fixes. Selon (Goodfellow et al., 2014), le discriminateur est entraîné jusqu'à ce qu'il soit optimal par rapport au générateur; puis ce dernier est à nouveau mis à jour. Cependant, en pratique, le discriminateur peut ne pas être entraîné jusqu'à ce qu'il soit optimal, mais peut plutôt n'être entraîné que pour un petit nombre d'itérations, et le générateur est mis à jour simultanément avec le discriminateur. L'apprentissage du GAN est arrêté lorsqu'un équilibre de Nash est atteint, ou lorsqu'un nombre maximum d'itérations soit atteint.

4.5 Réseaux auto-organisateurs

La carte auto-adaptative (ou *Self Organizing Map (SOM)*) ou réseau de Kohonen (Kohonen, 2012) est un type de réseau de neurones non supervisé basé sur un paradigme d'apprentissage compétitif. Cette technique est généralement utilisée pour effectuer des tâches telles que le regroupement, la détection des valeurs aberrantes et la réduction de la dimensionnalité (Kiang et al., 1995). Le but de cette organisation est de réduire l'espace de données d'entrée de grande dimension en une carte de neurones de faible dimension (généralement en deux dimensions) en utilisant des réseaux de neurones auto-organisés tout en conservant leurs structures topologiques.

4.5.1 Principe de fonctionnement

Dans un réseau SOM l'apprentissage ne se fait pas par présentation d'exemple, mais plutôt par une auto-adaptation à des stimuli d'entrée. En termes d'architecture, le réseau est composé de deux couches seulement, une couche d'entrée et une autre de sortie, comme illustré par la Figure 4.12. Les neurones de la couche d'entrée alimentent tous les neurones de la couche de sortie. Cette dernière, connue aussi sous le nom de couche cartographique, comprend un ensemble de neurones $M = \{n_1, \ldots, n_k\}$ entièrement connectés et disposés dans une carte de neurones bidimensionnelle (la carte est généralement représentée graphiquement sous la forme d'un rectangle avec k neurones ($k = x_{dim} \times y_{dim}$)). Chaque neurone, dans cette carte, est associé à un vecteur de poids W connu sous le nom de *code-book* décrivant le profil de données typique à une étape d'apprentissage donnée $t \in \{1, \ldots, T\}$, où T est le nombre total d'itérations durant la phase d'apprentissage.

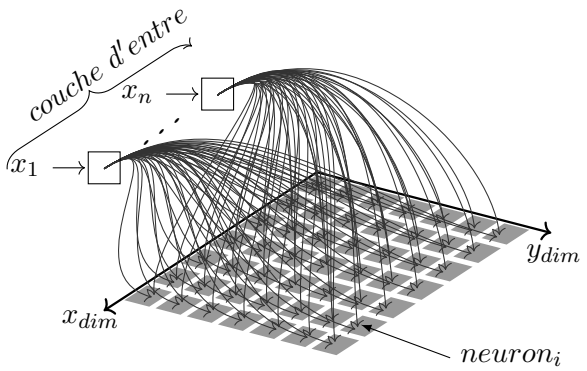


Figure 4.12 – Architecture des réseaux de Kohonen (Smahi et al., 2018)

4.5.2 L'apprentissage dans les réseaux de Kohonen

Le processus d'apprentissage compétitif tente de construire une topologie non linéaire qui assure la mise en correspondance des vecteurs d'entrée $X = \{\overrightarrow{x}(t) | t \in \{1, \ldots, T\}\}$, avec un ensemble de neurones M sur la carte. En termes d'apprentissage, l'algorithme SOM se compose de deux étapes, une étape compétitive et une deuxième de mise à jour:

1. Au cours de la première étape, chaque vecteur de données $\overrightarrow{x}(t)$, pour une phase d'apprentissage particulière t, est mis en relation avec le neurone qui lui correspond le mieux sur la carte. Le neurone sélectionné est noté:

$$n_i(\overrightarrow{x}(t)) = \underset{j \in 1,\ldots,k}{\operatorname{argmin}} || \overrightarrow{x}(t) - \overrightarrow{w}_j(t) || \tag{4.5.1}$$

où chaque neurone n_i code le sous-ensemble de l'espace de données d'entrée, dont les éléments sont plus proches de ce dernier (n_i);

2. Au cours de l'étape de mise à jour, les vecteurs de poids du neurone le mieux adapté n_i et de l'ensemble de ses neurones de voisinage $\Gamma(n_i)$ sont ajustés par rapport au vecteur d'entrée présenté $\overrightarrow{x}(t)$ à l'aide de l'équation suivante:

$$w_j(t+1) = w_j(t) + \eta(t) \cdot h_{i,j}(t) \cdot (x(t) - w_j(t)) \tag{4.5.2}$$

où $j \in \Gamma(n_i)$, $0 < \eta < 1$ est le taux d'apprentissage et $h_{i,j}$ est la fonction d'atténuation. La valeur de $h_{i,j}$ est fonction de la distance $d(i,j)$ du neurone n_i au neurone n_j dans la carte de sortie. Cette fonction renforce l'influence des neurones très proches, et considère les neurones plus éloignés comme inhibiteurs. Cependant, les neurones très éloignés n'exercent aucune influence.

Il faut savoir qu'il existe une autre variante, plus importante, du processus d'apprentissage SOM. Elle est largement utilisée dans les implémentations parallèles (Wittek et al., 2017). En effet, si toutes les données d'apprentissage sont disponibles au préalable, SOM dispose d'une formulation par lot (ou batch) pour la mise à jour de tous les poids:

$$w_j(T) = \frac{\sum_{t'=1}^{T} h_{i,j}(t')x(t')}{\sum_{t'=1}^{T} h_{i,j}(t')} \tag{4.5.3}$$

Cela signifie que les meilleurs neurones correspondants pour tous les vecteurs d'entrée sont sélectionnés; ensuite, leurs poids ainsi que les poids de leurs voisins respectifs sont mis à jour.

4.6 Conclusion

Dans ce chapitre, nous nous somme focalisés sur les différents principes et techniques liés à l'apprentissage automatique ainsi que l'apprentissage profond. En premier lieu, nous avons présenté les origines et les inspirations de la vision connexionniste de l'intelligence artificielle. Nous avons présenté, par la suite, la notion du perceptron considéré comme premier algorithme proposé pour l'apprentissage automatique. Puisque ce dernier présente certaines limites relatives à la résolution de problèmes non linéaires; nous avons présenté le modèle des réseaux de neurones artificiels. Dans cette partie, nous avons mis en avance les notions de fonctions d'activation ainsi que les différents types d'apprentissage utilisés dans le domaine d'apprentissage automatique. Par la suite, nous nous sommes focalisés sur la notion d'apprentissage profond. Nous avons commencé par présenter les réseaux de neurones convolutifs ainsi que les réseaux de neurones récurrents. Ensuite, une partie spéciale était réservée pour les modèles générationnels profond, à savoir, les auto-encodeurs et les réseaux antagonistes génératifs. A la fin de ce chapitre, nous avons présenté un type particulier de réseau de neurones connu sous le nom de carte Kohonen, que nous avons largement utilisé durant notre processus de prédiction de QoS.

Le chapitre suivant présente l'état de l'art des différents travaux autour de notre problématique.

Deuxième partie

Estimation de la QoS pour les services Web

Chapitre 5

Prédiction de la QoS par FC : état de l'art (Partie 1)

Sommaire

5.1 Introduction . . . 66
5.2 Stratégie de recherche documentaire adoptée . . . 67
5.2.1 Processus de recherche bibliographique . . . 67
5.2.2 Sélection d'articles . . . 68
5.3 La recommandation des services Web . . . 69
5.3.1 La prédiction de la QoS . . . 69
5.3.2 Classification des différentes méthodes . . . 70
5.4 Prédiction en séries chronologiques . . . 72
5.4.1 Approches d'analyse statistiques . . . 73
5.4.2 Approches à base de réseaux de neurones récurrents . . . 75
5.4.3 Synthèse des approches sensibles au temps . . . 77
5.5 Prédiction basée sur la mémoire . . . 79
5.5.1 Voisinage utilisateur . . . 79
5.5.2 Voisinage service . . . 81
5.5.3 Méthodes hybrides . . . 82
5.5.4 Synthèse des approches basées voisinage . . . 83
5.6 Conclusion . . . 84

5.1 Introduction

Avec le développement rapide et remarquable enregistré dans le domaine de l'informatique orientée services, combiné aux améliorations dans le domaine des technologies Internet; une très grande augmentation dans le nombre de services Web a été enregistrée. En effet, plusieurs portails Web, registres en ligne et marchés de services Web ont également vu le jour, permettant aux développeurs et fournisseurs d'exposer, de gérer et de faire connaître leurs services. Les utilisateurs finaux de services Web peuvent à présent explorer ces référentiels pour y découvrir les services qui les intéressent. Selon (Adeleye et al., 2019), les services Web sont devenus omniprésents du fait que la majeure partie des applications disponibles actuellement sont proposées sous la forme d'interfaces d'APIs Web.

L'omniprésence et l'adoption croissante des services Web ont complètement révolutionné le développement des applications orientées services (Bouguettaya et al., 2017). Cependant, avec le lancement constant et continu de nouveaux services Web par les grandes, moyennes et petites entreprises, de nombreux services Web ayant des fonctionnalités pratiquement similaires apparaissent sur Internet. Cet

enrichissement continu des services Web ne fournit pas seulement plus d'alternatives pour les développeurs d'applications, mais les rend également embourbés dans l'éventail des options et des choix accessibles. Par conséquent, l'objectif dans ce qui est à venir n'est pas seulement de rendre plus de services Web disponibles, mais plutôt comment réaliser la recommandation des services adéquats aux utilisateurs les plus appropriés.

Actuellement, la recommandation de service est largement étudiée dans le domaine de l'informatique orientée services (Jatoth et al., 2017). Elle vise à fournir un support aux utilisateurs afin de trouver les services Web les plus appropriés en fonction de leurs préférences en matière de QoS; cela, au lieu d'une recherche manuelle, via des moteurs de recherches, à travers un volume important de services Web. Mais puisqu'il existe de nombreux services Web aux fonctionnalités équivalentes, la considération de la fonction d'application associée au service comme seul moyen pour filtrer les services Web ne suffit pas pour un système de recommandation de services Web. Cependant, ce qui distingue un service Web donné d'un autre est sa capacité à travailler dans un environnement complétif avec des contraintes en termes de QoS. En effet, la QoS est un terme très complexe qui comprend un ensemble de caractéristiques spécifiques. Elle est définie comme un groupe de différents attributs non fonctionnels (Ran, 2003), dans lequel chaque attribut représente l'information de QoS du service Web dans un aspect bien spécifique, tel que le temps de réponse, le débit, la disponibilité du service, etc. Ces attributs peuvent être soit du côté utilisateur, par exemple, le temps de réponse et le débit, soit du côté fournisseur, comme le prix et la sécurité.

D'un point de vue pratique, chaque propriété de la QoS se verra attribuer une estimation spécifiée, afin d'indiquer sa valeur, appelée évaluation de la QoS. D'une manière générale, les services Web sont sélectionnés en fonction de leurs valeurs optimales en matière d'attributs de QoS. Il en résulte que la tâche de prédiction des valeurs QoS inconnues des services Web pour un système de recommandation est devenue primordiale. En effet, la problématique de prédiction pour les services Web fera l'objet de ce chapitre.

L'architecture de ce chapitre est la suivante. En premier lieu nous présentons notre stratégie de recherche bibliographique que nous avons établi afin de dresser un état-de-l'art autour de notre thématique de recherche. Ensuite, nous invoquons la notion de recommandation des services Web en présentant les différents challenges et intérêts de la prédiction des valeurs de QoS. Nous dressons, à la fin de cette partie, une classification des différentes méthodes de prédiction de la QoS à base de filtrage collaboratif. Dans la section qui suit, nous nous concentrons sur les travaux appartenant à la catégorie d'approches à base de séries chronologiques. Nous mettons l'accent plus particulièrement aux travaux utilisant les méthodes d'apprentissage profond (réseaux de neurones récurrents). La dernière section est réservée aux travaux utilisant des approches basées sur le voisinage. Il est à noter que les deux dernières catégories de travaux, à savoir, les méthodes basées sur les modèles ainsi que les approches hybrides, seront présentées dans le chapitre suivant.

5.2 Stratégie de recherche documentaire adoptée

Afin d'établir un état-de-l'art assez complet que possible, nous nous sommes basés sur une recherche documentaire portant sur un ensemble d'articles spécialisés autour de notre problématique de recherche. La mise en œuvre de cet état-de-l'art a été effectuée grâce notamment à une combinaison de stratégies de recherche qui a été adoptée pour la *collecte* de ces différents articles.

5.2.1 Processus de recherche bibliographique

Pour l'essentiel, et afin de diminuer le risque de ne pas sélectionner les articles pertinents et importants pour notre état-de-l'art (par conséquent augmenter la chance de les sélectionnés), un ensemble de stratégies, dans un processus itératif, a été appliqué afin d'identifier ces articles. Le processus de filtrage des articles non pertinents est illustré par la Figure 5.1. Pour la mise en place de ce processus, nous avons défini un processus en six étapes comme suit:

1. Le recours aux moteurs de recherche:

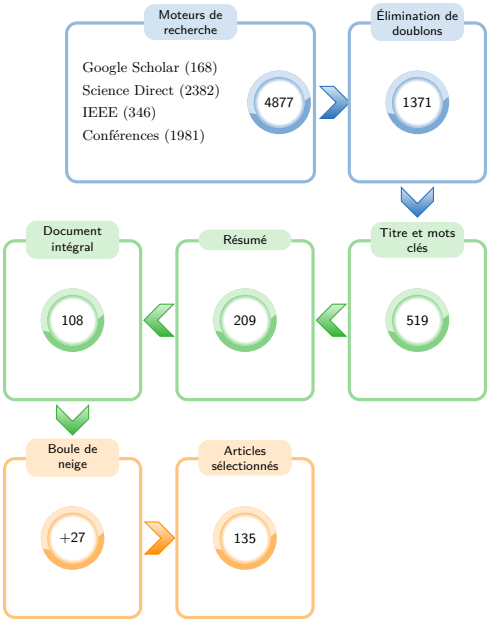


Figure 5.1 – Processus de sélection d'articles bibliographiques

- L'utilisation du moteur de recherche Google Scholar [26] comme outil principal dans notre recherche documentaire;
- Le recours aux moteurs de recherches disponibles au niveau des grandes bases de données d'articles universitaires en ligne, telles que: IEEE [27], Scopus [28], Science Direct [29];
- La consultation des actes de congrès récents de plusieurs conférences de haut niveau dans le domaine des services Web, notamment l'*IEEE International Conference on Web Services* (IEEE ICWS [30]), *Springer International Conference on Web Services* (Springer ICWS [31]);

2. Après avoir récupéré les articles recueillis à travers ces différents canaux, un processus d'examen est appliqué pour filtrer ces articles. Nous commençons, en premier lieu, par la suppression des références en double. Cette étape a été effectuée automatiquement via l'outil de gestion bibliographique Mendeley [32];
3. L'élimination des articles par titre, ce qui nous a permis de sélectionner que les travaux connexes avec notre thématique de recherche;
4. L'étape suivante consiste en une sélection d'article par leur résumé, en retirant les articles que nous avons jugé non pertinents, et qui ne proposaient pas une approche de qualité en tant que contribution;
5. Dans la cinquième étape, une lecture complète des articles restants est effectuée. Dans cette étape, nous avons sélectionné que les articles qui présentaient une approche de qualité;
6. L'application du processus de « boule de neige » (Wohlin, 2014) consistant à consulter la liste des références bibliographiques des articles retenus. Une liste des trois meilleures références est intégrée de nouveau dans le processus de recherche actuel à partir de l'étape 2.

5.2.2 Sélection d'articles

Dans chacune de ces différentes étapes, nous avons appliqué un système de notation afin de déterminer les articles qui seront sélectionnés pour la prochaine étape. Ainsi, nous attribuons une note de

26. https://scholar.google.fr/
27. https://ieeexplore.ieee.org/search/advanced
28. https://www.scopus.com
29. https://www.sciencedirect.com/search
30. https://conferences.computer.org/icws/
31. https://link.springer.com/conference/icws
32. https://www.mendeley.com/

1 si l'article répond à la majorité des critères de sélection, une note de 0.5 s'il les rempli partiellement ou s'il y a des doutes dans la décision et 0 point si l'article ne répond pas aux critères de sélection.

Tableau 5.1 – Formalisation des requêtes de recherche

Objectif	Syntaxe de la requête
Recommandation des services Web	(« web services » **OR**« SOA » **OR**« WS-DL ») **AND** (« recommender system » **OR**« recommendation ») **AND** (« qos » **OR**« quality of service »)
Factorisation matricielle	(« QoS prediction » **OR**« QoS » **OR**« Prediction ») **AND** (« Matrix Factorisation » **OR**« SVD » **OR**« SVM »)
Filtrage collaboratif	(« web services » **OR**« SOA » **OR**« WS-DL ») **AND** (« QoS prediction » **OR**« QoS » **OR**« Prediction ») **AND** (« Collaborative Filtering »)

Il faut savoir que dans la première étape différentes requêtes de recherche ont été appliquées en se basant sur une liste de mots clés: « QoS prediction », « web services », « Collaborative filtering », « deep learning », « machine learning », « WS-Dream ». Le tableau 5.1 présente quelques exemples de combinaison de mots clés que nous avons utilisées afin d'établir ces différentes requêtes.

5.3 La recommandation des services Web

Aux cours des dernières années, une avancée technologique considérable à été enregistrée autour des domaines des systèmes de recommandation ainsi que des services Web. De ce fait, la recommandation de services Web est considérée comme le résultat d'un couplage entre ces deux domaines. La recommandation de services Web est vue comme un processus automatique de découverte et de recommandation d'un certain nombre de services Web à un utilisateur final. Cette recommandation comprend la détermination automatique de l'utilité ou de la pertinence des services Web, la sélection des différents services candidats sur la base des exigences préalables des consommateurs et la recommandation, de façon proactive, d'un sous ensemble de ces services considérés comme les plus appropriés à l'usager final.

5.3.1 La prédiction de la QoS

Différents chercheurs et spécialistes ont tenté de recommander de manière proactive des services Web aux consommateurs en fonction de limites et d'inclinations spécifiques. De cette façon, la recommandation de services web est devenue un processus actif de recherche, de découverte et de sélection de services qui peut être utilisé avec différentes stratégies et techniques informatiques et mathématiques telles que l'analyse de données, l'apprentissage automatique, l'apprentissage profond, la théorie des graphes et les approches probabilistes, afin d'améliorer la précision de la prédiction (Yao et al., 2015).

Malgré le fait que les méthodes de recommandation de services Web actuelles présentent des améliorations; les tâches de recommandation continuent de représenter un grand challenge aux concepteurs et aux développeurs de services en raison de l'expansion rapide et continue du nombre de services Web. Par définition, les systèmes de recommandation fournissent un support pour la sélection d'items et de services conventionnels. Cependant, leur application directe au domaine des services Web n'est pas une tâche facile en raison de plusieurs défis tels que:

— Un besoin d'adaptation des services Web en fonction des différents besoins et exigences des utilisateurs (Liu et al., 2013);

— Le manque de données de rétroaction des utilisateurs (user feed-back) sur les services Web (Lecue, 2010);

— Une difficulté dans la détermination du degré de crédibilité sur les rétroactions des utilisateurs (le retour d'expérience des usagers est souvent subjectif) (Smithamol and Rajeswari, 2019);

— Une rareté de donnée en termes de QoS et en termes d'éléments contextuels (Su et al., 2017);

Les systèmes de recommandation de services Web sont considérés comme une excellente alternative aux moteurs de recherche de services Web (tel que Programmable Web[33]). Largement utilisées, les approches basées sur la recherche par mots clés sont peu performantes en matière de recommandation car, d'un côté, elles ignorent les caractéristiques non fonctionnelles des services Web (Yao et al., 2015). D'un autre côté, ce type d'approches dépend fortement des requêtes complexes et moins correctes des utilisateurs.

Toutefois, il convient de noter qu'il est souvent nécessaire de prévoir les valeurs QoS inconnues des services Web dans l'analyse des recommandations de service Web, ce qui est principalement pour les raisons suivantes :

— *Volume de services Web*: Le nombre de services Web disponibles sur le Web est important et ne cesse de croître, par conséquent, les valeurs QoS connues seront rares. En effet, auprès des fournisseurs de services, l'utilisateur n'invoque que quelques-uns de ces services Web, alors que les valeurs QoS de la plupart des services Web disponibles sont encore inconnues pour lui. En réalité, il est peu pratique pour que chaque utilisateur invoque tous les services Web susceptibles d'être importants afin de connaître leurs valeurs QoS;

— *Manque de ressources côté utilisateur*: Il est difficile d'effectuer des tests de QoS sur tous les services Web disponibles. En raison du grand nombre de services Web disponibles, l'invocation de ces derniers dans le monde réel, est considérée comme une tâche coûteuse en termes de ressources matérielles, financières et de disponibilité de la part des utilisateurs. En outre, l'invocation de services Web impose des coûts aux utilisateurs de services car souvent de nombreux services Web ne sont pas gratuits, et les fournisseurs de services doivent les facturer pour chaque invocation;

— *Nature dynamique de la QoS*: Les données en termes de QoS du côté utilisateur, par exemple le temps de réponse et le débit, varient considérablement d'un utilisateur à un autre; ceci est en en fonction de nombreux facteurs, tels que la position géographique, les conditions du réseau et l'environnement d'exécution côté fournisseur de service (Yu and Huang, 2016). Différents utilisateurs sont susceptibles de percevoir une expérience QoS différente, ce qui signifie que l'expérience QoS d'un utilisateur peut ne pas être significative pour d'autres usagers.

5.3.2 Classification des différentes méthodes

En littérature un large éventail de travaux sur les systèmes de prédiction de la QoS pour les services Web a été développé. Différentes tentatives de classification de ces derniers ont été réalisées. Généralement ces tentatives sont basées sur les différentes classifications proposées dans systèmes de recommandation.

33. https://www.programmableweb.com/

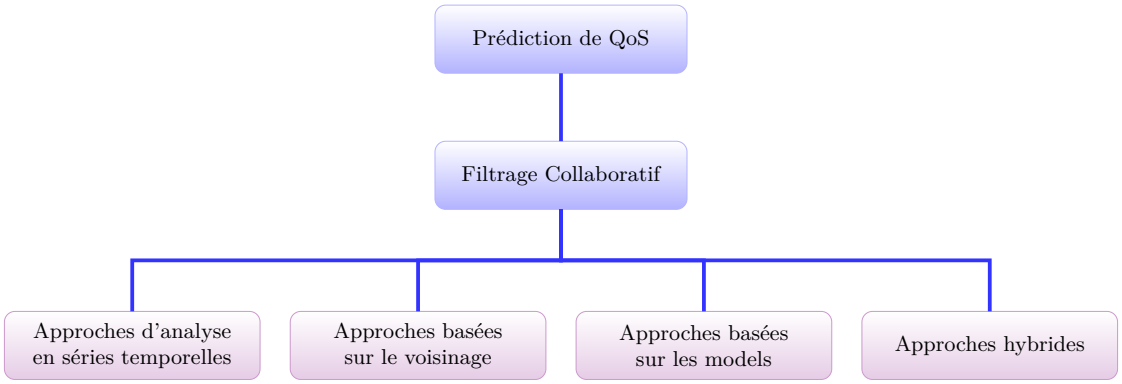


Figure 5.2 – Classification globale des approches de prédiction de la QoS [34].

En ce qui nous concerne, la classification que nous considérons s'inspire de celle adoptée dans le contexte des systèmes de recommandation définie dans (Su and Khoshgoftaar, 2009), et qui est souvent utilisée dans les systèmes purement de collaboration. Cette dernière est couplée avec la classification définie dans (Ghafouri et al., 2020), et qui est une classification spécifique pour la prédiction de QoS à base de filtrage collaboratif. Comme l'illustre la Figure 5.2, différents types de techniques de filtrage collaboratif peuvent être utilisés pour la mise en œuvre de tels systèmes. La plupart de ces travaux peuvent être classées en quatre catégories principales:

— Approches fondées sur l'analyse de séries chronologiques des valeurs de QoS;

— Approches basées sur le voisinage;

— Approches basées sur les modèles

— Approches hybrides.

Les techniques basées sur un filtrage collaboratif, connues aussi sous le nom d'approches statiques, sont divisées en deux groupes distincts : (a) approches basées sur des modèles et (b) approches basées sur la mémoire.

Afin d'améliorer la précision des prédictions, divers types de facteurs sont pris en compte lorsque les valeurs QoS manquantes sont prédites par les différents algorithmes développés en état-de-l'art:

— **Le temps** (Mezni and Fayala, 2018): Les valeurs des différents attributs de QoS, tels que, le temps de réponse, le débit ou le taux de disponibilité des services Web, sont très variables pour le même utilisateur par rapport aux différents moments d'invocation. En effet, cela est dû au fait que ces attributs dépendent essentiellement de l'état du service mais aussi de la qualité actuelle de trafic réseau;

— **La localisation** (Mingdong Tang et al., 2012): Le facteur met en avant la possibilité que les différents utilisateurs et services peuvent être regroupés sur la base de leurs emplacements physiques. En proposant cette démarche de partitionnement sur l'ensemble des utilisateurs et/ou services, ce facteur est souvent exploité afin d'aborder les questions relatives au problème d'évolutivité (*scalability*) des données pour les systèmes de recommandation;

— **La personnalisation** (Jiang et al., 2011): Ce facteur est souvent utilisé dans les algorithmes de FC basés sur la mémoire. Il tient compte de l'influence de la personnalisation des éléments des services Web lorsque le degré de similarité entre les utilisateurs est à calculer. L'idée est basée sur l'observation que les services les plus populaires ou ceux dont la QoS est plus stable d'un utilisateur à l'autre, devraient contribuer moins à la mesure de la similarité des utilisateurs;

— **La réputation** (Xu et al., 2016a): La réputation ou « *trust-aware* » des utilisateurs est généralement considérée comme un facteur important dans les systèmes de recommandation. La plupart de ces derniers supposent que les valeurs de QoS données par les utilisateurs sont valides. Néanmoins, à vrai dire, ce n'est pas toujours correct. En effet, les utilisateurs peuvent soumettre des valeurs QoS peu fiables. Par conséquent, ce manque de fiabilité sur les données constitue l'un des défis majeurs pour un système de prédiction.

Une autre classification définie dans (Zheng et al., 2020), que nous avons jugé intéressante, basée sur les différents facteurs présentés ci-dessus, est illustrée par la Figure 5.3.

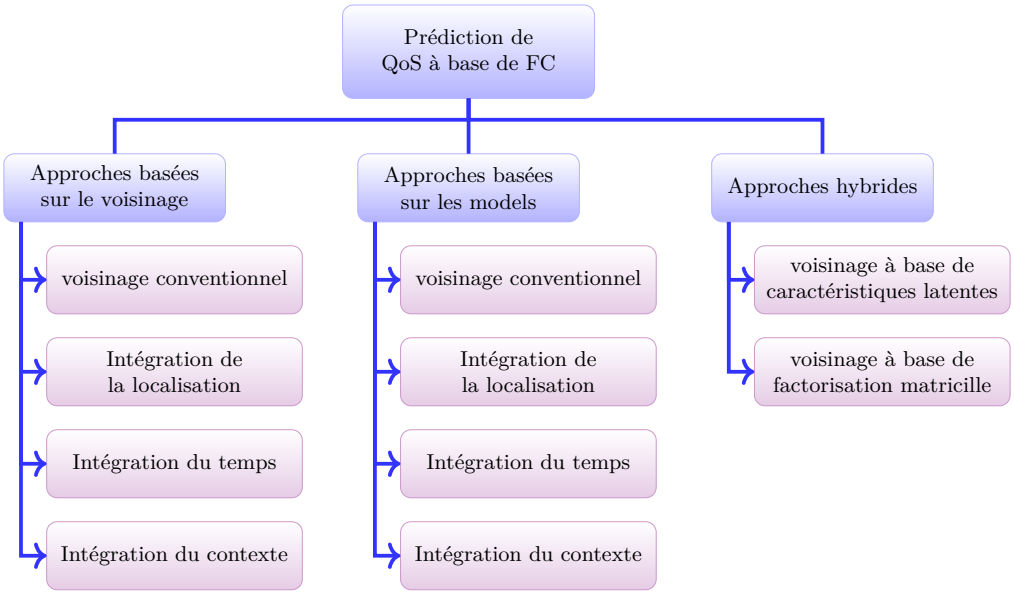
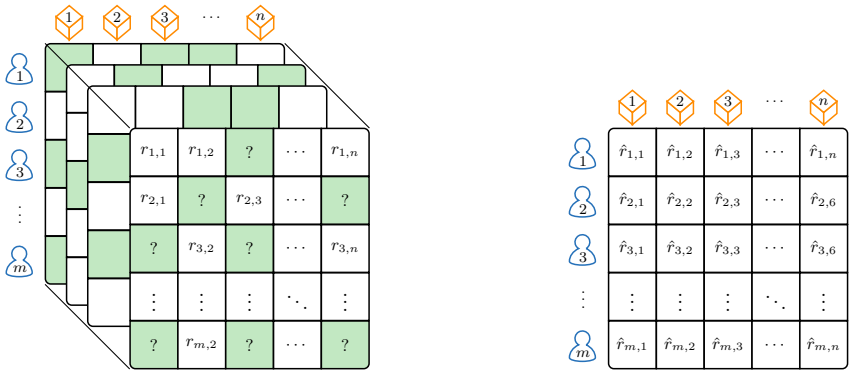


Figure 5.3 – Approches de prédiction de la QoS basées sur le FC (Zheng et al., 2020)

Cette classification présente les différents travaux réalisés en littérature sur la problématique de prédiction de la QoS pour les services Web basée sur la technique de filtrage collaboratif tout en exploitant les différents facteurs définis précédemment. Dans la partie qui suit, nous nous intéressons aux travaux sur la prédiction de la QoS basée sur les séries chronologiques des valeurs de QoS.

5.4 Prédiction en séries chronologiques

Un des problèmes majeurs dans les systèmes de prédiction de QoS est qu'ils considèrent que les valeurs de QoS sont indépendantes les unes des autres. Alors qu'en réalité, les valeurs antérieures de QoS, pour un service donné, ont une incidence directe sur les futures valeurs possibles. L'étape de base consistant à prédire ces futures valeurs de QoS peut être considérée comme un problème de série temporelle, où la prochaine valeur de la série doit être prédite (Amin et al., 2012). L'objectif est donc, de prévoir la prochaine valeur sur cette série temporelle de QoS.



(a) Collection des matrices brutes **(b)** Le nouvelle matrice prédite

Figure 5.4 – Modélisation de la relation utilisateur, service et intervalles de temps

Formellement, la prédiction de la QoS en fonction du temps (ou sensible au temps), connue en anglais sous l'intitulé « *time-aware prediction* » peut être représentée comme un entassement chronologique (historique) de matrices à deux dimensions (utilisateur/service). Où chaque matrice représente

les données de QoS observées, pour un ensemble de services, du point de vue des utilisateurs. Sur la base de cet historique de QoS à k intervalles de temps, à l'instant t actuel, le système de recommandation de services Web doit prédire les valeurs de QoS de toutes les paires utilisateur-service et choisir le service le plus adéquat pour l'utilisateur cible. En d'autres termes, supposant que le système de recommandation se compose de m utilisateurs, n services et k intervalles de temps. La matrice $Q = m * n * k$ illustrée par la Figure 5.4 est utilisée afin de représenter la relation, en termes de QoS, entre le triplet utilisateur, service et intervalle de temps.

5.4.1 Approches d'analyse statistiques

Les valeurs de certains attributs de QoS, tels que, le temps de réponse, la disponibilité ou le débit, ne sont pas invariables dans le temps. Cela est dû principalement à l'état dynamique du réseau et de son trafic très variable. Dans les cas où les valeurs d'une propriété de QoS dépendent du temps, les séries temporelles sont considérées comme un bon outil pour prédire leurs futures valeurs (Syu et al., 2017). En effet, l'analyse et la prévision de séries temporelles ou chronologiques peuvent être effectuées via l'utilisation d'un large éventail d'algorithmes. Pour la prévision des attributs QoS, différentes méthodes peuvent être choisies. En littérature (Wagner et al., 2007), ces techniques sont scindées en deux catégories différentes, comme illustré par la Figure 5.5. D'un côté, les méthodes classiques basées sur des concepts statistiques et mathématiques. De l'autre côté, les méthodes heuristiques modernes basées sur des algorithmes d'apprentissage automatique et d'apprentissage profond.

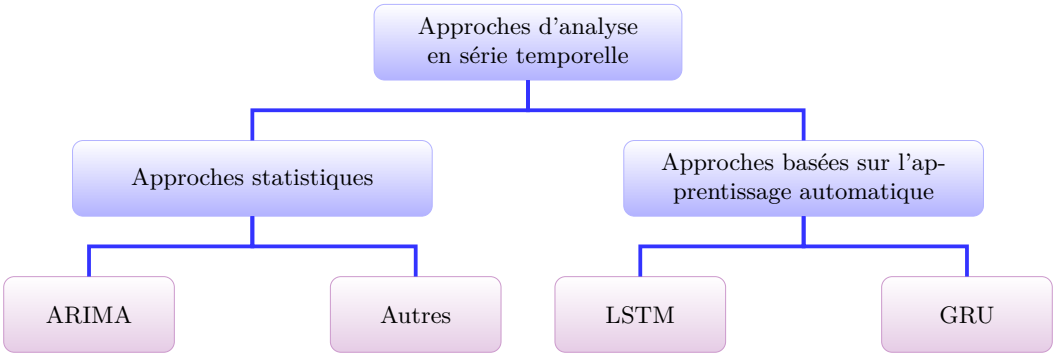


Figure 5.5 – Classification des approches d'analyse en séries temporelles

Dans cette partie du chapitre, nous présentons les différents travaux de références appartenant à cette catégorie d'approche.

Les modèles ARIMA

Le modèle de moyenne mobile intégrée auto-régressive ARIMA, abréviation de *AutoRegressive Integrated Moving Average* est une méthode d'analyse en série temporelle initialement introduits par (Box et al., 2015). ARIMA est la généralisation de modèles autorégressifs et moyenne mobile ARMA de Box et Jenkins (Liu and Hudak, 1992). Ce type de modèles a pour objectif de modéliser sous une forme succincte des données variant dans le temps, et de faire des estimations et des prévisions pour leurs futures valeurs. Ils ont été utilisés pour un large éventail d'applications de détermination de séries temporelles, telles que, à titre d'exemple, les prévisions socio-économiques et les estimations financières, les prévisions sur les flux de véhicules dans les routes, les prévisions météorologiques, etc.

Les modèles ARIMA de prévision comprennent trois parties: partie auto-régressif (AR), partie moyenne mobile (MA) et finalement la partie d'intégration (I) du modèle. Ces composantes peuvent être utilisées ensemble ou indépendamment pour déterminer et faire des prévisions sur des séries chronologiques. Pour chaque composante, il existe un nombre (p pour le modèle AR, d pour I et q pour MA) qui doit être défini pour qu'un modèle ARIMA puisse effectuer des conjectures et des prévisions. Les valeurs adéquates sont déterminées en analysant les données d'apprentissage passées afin de déterminer si l'intégration (I) est nécessaire pour rendre la série temporelle stationnaire (la moyenne

et la variance sont constantes au cours du temps). Lorsque la série temporelle a été rendue stationnaire, les retards AR et MA peuvent être déterminer en établissant un graphique d'auto-corrélation et d'auto-corrélation partielle et en choisissant un nombre approprié de retards. Ce processus peut être automatisé en effectuant une recherche par quadrillage sur les valeurs des composants sur un échantillon de l'ensemble de données d'apprentissage. Formellement, une série temporelle $\{x_t\}$ peut être modélisée par un modèle ARMA d'ordres p et q, désigné par $ARMA(p,q)$, s'il satisfait:

$$x_t = \sum_{i=1}^{p} \phi x_{t-1} + \sum_{j=1}^{q} \theta \epsilon_{t-j} + \epsilon_t$$

où ϕ est le polynôme auto-régressif d'ordre p; θ est le polynôme de moyenne mobile d'ordre q et ϵ_t est l'erreur courante et ϵ_{t-j} est les erreurs antérieures. Si la série temporelle d'origine $\{x_t\}$ est non stationnaire, il faut procéder à des d différences pour la transformer en une série stationnaire $\{\hat{x}_t\}$. Par conséquence, la série $\{\hat{x}_t\}$ serait générée par un modèle ARIMA d'ordre p,d et q, noté $ARIMA(p,d,q)$.

Travaux existants

Dans ce contexte, les auteurs dans (Ding et al., 2018) partent du principe que les valeurs de QoS varient fréquemment au fil du temps et que les approches existantes en matière de recommandation de services Web négligent cette propriété ou la traite d'une manière inadéquate; ce qui se traduit par une recommandation inefficace. De cette constatation, les auteurs proposent une méthode de recommandation de services Web sensible au temps (*a time-aware service recommendation*) en combinant la méthode de FC basée sur les séries chronologiques à celle basée utilisateur. Dans un premier temps, ils proposent une démarche de filtrage collaboratif renforcée par une mesure de similarité entre utilisateurs, tout en traitant le problème de la rareté de données en termes de QoS via la prédiction des valeurs manquantes de la QoS. Ils appliquent ensuite le modèle de prédiction des séries temporelles ARIMA pour la prédiction de la QoS. Il est à noter que ce travail rentre dans le contexte de l'informatique dématérialisée (cloud computing) où des études assez récentes ont révélé que les indicateurs de QoS présentent une forte instantanéité dans cet environnement (Wang et al., 2016).

(Hu et al., 2015) proposent une nouvelle approche de prédiction personnalisée de la qualité de service en combinant une approche de filtrage collaboratif avec une version plus améliorée pour le traitement des séries chronologiques basée sur le filtrage de Kalman [35] afin de compenser les limites enregistrées dans le modèle ARIMA. Donc, cette approche est composée par deux phases distinctes: une phase de prévision locale de la QoS pour chaque service Web individuel basée sur l'analyse des séries temporelles. La deuxième phase utilise un algorithme de filtrage collaboratif modifié centré sur le voisinage utilisateur pour faire une prédiction de QoS personnalisée et adaptée. Afin de compenser les limites du modèle ARIMA, les auteurs l'ont combiné avec le filtrage de Kalman. En effet, ils sont partis du principe que le filtrage de Kalman peut compenser efficacement la déficience des modèles ARIMA en utilisant son mécanisme de rétroaction pour corriger les prévisions du modèle final. Ainsi, leur approche se compose essentiellement de trois étapes: (1) La première étape consistait à l'élaboration d'un modèle ARIMA pour chaque service Web individuel. Ce modèle est construit afin de tenir compte des variations des données antérieurs en matière de QoS; ce qui inclut principalement la détermination du comportement du modèle via l'estimation des différents paramètres du modèle au moyen d'un ensemble de méthodes de calibration bien définies; (2) L'étape suivante s'intéresse à une conversion de manière formelle du modèle ARIMA, construit en première étape, en une représentation de l'espace d'état, comme l'exige le filtrage de Kalman. Ce dernier est utilisé pour corriger les prévisions de QoS en incorporant de nouvelles observations de QoS dans le processus de prédiction; (3) La dernière étape consistait à l'Utilisation d'un algorithme de filtrage collaboratif modifié basé sur le voisinage utilisateur afin de réaliser les prédictions de la QoS. Cette phase s'effectue sur la base des résultats de la prévision de la série temporelle et des observations historiques de la QoS fournies par les voisins qui partagent des expériences de QoS similaires.

35. Pour plus d'information sur les filtres de Kalman, se référer au document (Welch and Bishop, 1995)

5.4.2 Approches à base de réseaux de neurones récurrents

La QoS est un critère important pour évaluer un système de recommandation de services Web. En raison de plusieurs facteurs tels que les diverses conditions du réseau, la nature cyclique des données, etc., les valeurs QoS sont dynamiques et évoluent dans le temps. En règle générale, les données sont trop rares pour s'adapter aux modèles traditionnels de prédiction des séries chronologiques ARIMA. Aussi, de tels modèles traditionnels nécessitent un réglage manuel afin de définir l'irrégularité pour les différents paramètres (White, 2020). De cette constatation, les réseaux de neurones récurrents à mémoire court-terme et long terme (LSTM) (Hochreiter and Schmidhuber, 1997) et réseau de neurones récurrents à portes (GRU) (Cho et al., 2014) sont beaucoup utilisés dans la prédiction de la QoS. Ces deux types de réseaux de neurones ont été développés à partir des réseaux de neurones récurrents (RNN) en améliorant leur architecture de base. Ces types de réseaux ont été exploités dans une grande variété d'applications telles que le traitement du langage naturel (Sutskever et al., 2011), la modélisation du signal de la parole, la climatologie, et beaucoup d'autres (Chen et al., 2013).

Long Short-term Memory (LSTM)

Avec la très grande progression enregistrée dans le domaine des clouds, son intérêt s'est en outre multiplié. Les travaux de (Sahu et al., 2021) s'inscrivent dans ce contexte, où une recommandation à base de la QoS représente un vrai défi pour les fournisseurs de cloud services. La prédiction dans un environnement où les valeurs de QoS ont un caractère dynamique et variant dans le temps, est une tâche complètement différente. En effet, les autres méthodes telles que le filtrage collaboratif basé sur la mémoire ne prennent pas en considération cet aspect temporel. Aussi, les méthodes basées sur des algorithmes d'apprentissage automatique ne sont pas assez flexibles pour intégrer le facteur temporel, bien que certains travaux basés sur les facteurs tensoriels et les modèles factoriels, telles que (Ma et al., 2015; Zhang et al., 2019b), ont pu les incorporer afin de prédire les valeurs de QoS. Mais, selon les auteurs, aucune de ces méthodes n'a été initialement élaborée ou conçue pour les données de QoS chronologiques.
Les auteurs proposent donc un modèle basé sur le réseau LSTM, qui est un type de réseau neuronal récurrent, spécialement conçu pour contenir des dépendances à long terme. Ce dernier sera combiné à la factorisation matricielle afin d'extraire les fonctionnalités internes d'un côté, et de résoudre le problème de démarrage à froid de l'autre côté. Les auteurs ont démontré aussi, via un ensemble d'expériences, que leurs résultats ont été excellentes en les comparants à d'autres travaux existants.

Sous l'impulsion d'une utilisation généralisée des architectures orientées services, les auteurs, dans (Wu et al., 2019b), ont constaté que le nombre de services Web ainsi que de leurs utilisateurs ne cesse d'augmenter. La grande quantité de données et d'information accumulées, via les différentes invocations de services, a rendu possible une recommandation plus appropriée de services que les utilisateurs pourraient invoquer dans l'avenir. Les auteurs constatent que deux problèmes majeurs subsistent pour les systèmes de recommandations actuels. Premièrement, ils ne prennent en considération que la valeur QoS la plus récente, en négligeant les caractéristiques dynamiques de la QoS. En second lieu, ces systèmes de recommandation considèrent les préférences de l'utilisateur et la qualité de service comme deux variables indépendantes, tout en négligeant leur association inerte possible (ou lien latent). Afin de surmonter les limites de ces travaux antérieurs, les auteures proposent un nouvel algorithme de recommandation, nommé QF-RNN « QI-Matrix Factorization Based Recurrent Neural Networks ».

Le processus consistait en deux étapes principales. Tout d'abord, les métriques de QoS sont combinées avec les enregistrements des différentes invocations en suivant un certain nombre de règles afin d'obtenir la matrice QI. Cette dernière permet de refléter d'une manière exhaustive si un service donné est invoqué ou non tout en lui combinant avec ses différentes valeurs de QoS. Par la suite, et à chaque intervalle de temps, un algorithme de factorisation matricielle, basée sur les réseaux de neurones récurrents, est appliqué sur la matrice QI. L'objectif est d'extraire les préférences de l'utilisateur et les caractéristiques du service pour ces différents intervalle de temps. Cet algorithme a pour objectif de prédire les futures valeurs de QoS ainsi que les différentes caractéristiques des services. En se basant

sur les résultats de prédiction du réseau LSTM, la matrice de QI pour l'intervalle de temps suivant peut-être étendue et prédite. A la fin, une liste des Top-n recommandations, avec les valeurs de QoS les plus élevées, sera proposée à l'utilisateur.

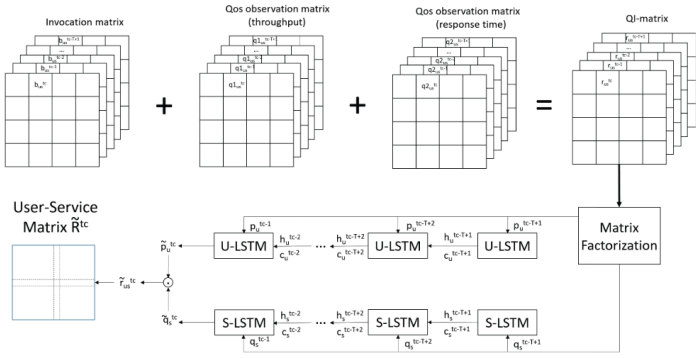


Figure 5.6 – Architecture du modèle QF-RNN (Wu et al., 2019b)

La figure 5.6 représente la Structure du modèle QF-RNN. À chaque intervalle de temps, la matrice QI est générée en intégrant la matrice d'invocation avec celles d'observation de QoS (dans leur car, le débit et le temps de réponse). Ensuite, le processus de factorisation matricielle est utilisé pour extraire les éventuelles représentations latentes. Après un apprentissage suffisant du modèle, QF-RNN obtient les paramètres optimaux du modèle afin de prédire la valeur de QoS pour chaque paire utilisateur/service pour l'intervalle de temps suivant (tc).

Dans (Xiong et al., 2018a), les auteurs constatent que de nombreux travaux sur la prédiction de QoS sensibles au temps ont été proposés et réalisées; et que ces travaux ont atteint des performances de prédiction très encourageantes. Cependant, ces approches proposées présentent un inconvénient majeur car elles ne répondent pas efficacement aux exigences de la prédiction instantanée. En effet, ces algorithmes ne fournissaient pas de mécanismes efficaces de mise à jour sur leurs modèles engendrés; ce qui les obligent de ré-entraîner d'une manière périodique l'ensemble de ces modèles en utilisant un grand volume de données (c'est-à-dire tout l'historique) afin qu'ils puissent traiter les nouvelles données en termes de QoS. Par conséquent, les auteurs considèrent que la mise à jour en temps opportun du modèle de prédiction, afin de prédire une nouvelle valeur instantanée, devient un problème important.

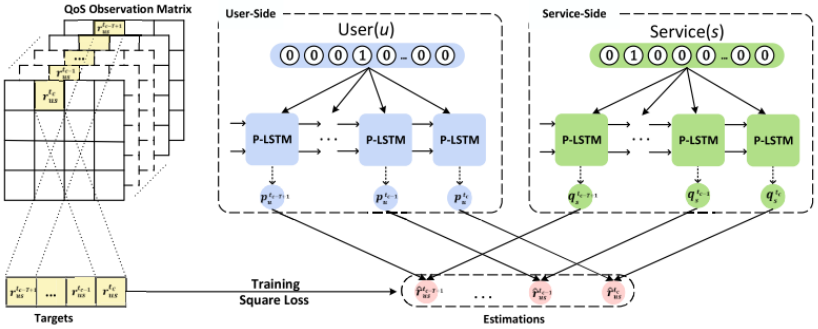


Figure 5.7 – Architecture du modèle PLMF (Xiong et al., 2018a)

Pour répondre à ce problème, les auteurs proposent une nouvelle approche personnalisée de factorisation matricielle basée sur les réseaux LSTM pour la prédiction de la QoS. En effet, les auteurs préconisent que leur modèle 5.7, nommé PLMF, peut caractériser et décrire efficacement les représentations latentes dynamiques de plusieurs utilisateurs et services à la fois. Aussi, le modèle proposé peut être mis à jour en fonction des observations actuelles et des données historiques, ainsi que par certaines informations conservées à long terme afin de réaliser une prédiction efficace.

Gated Recurrent Unit (GRU)

Comme mentionnée précédemment, les GRUs ont été développés à partir de l'architecture de base des RNN en modifiant leur forme fonctionnelle pour faire en sorte que chaque unité récurrente capture de manière adaptative les dépendances sur différents intervalles de temps. Ils ont des performances comparables aux LSTM pour la prédiction de séries temporelles. Le modèle GRU possède des portes synchronisées qui permettent d'omettre ou de mémoriser sélectivement les informations de la séquence temporelle précédente dans une mémoire dynamique à l'intérieur de chaque unité.

Les auteurs (Liang et al., 2021) proposent un filtrage collaboratif basé sur l'architecture RNN pour la prédiction QoS dans le contexte des IoT, nommé RNCF. Plus précisément, le modèle GRU est incorporé au RNCF pour modéliser l'état dynamique de l'environnement en exploitant l'historique d'invocation, contenu dans des matrices, sur un ensemble d'intervalles de temps. Ils considèrent que le GRU multicouches offre la possibilité de partager les enregistrements d'invocation sur les différents intervalles de temps. De plus, la complémentarité des informations d'invocation aide efficacement à atténuer le problème de la rareté des données.

Dans (Hasnain et al., 2020), les auteurs ont tenté de proposer un cadre de détection d'anomalies en utilisant les caractéristiques dynamiques de QoS. Pour cela, ils utilisent trois variantes de RNN, à savoir: le RNN simple, le réseau LSTM et le GRU. Un ensemble d'expériences ont été menées révélant que la méthode proposée a pu détecter, avec une grande précision, les anomalies qui peuvent exister dans les services Web.

5.4.3 Synthèse des approches sensibles au temps

Durant ces dix dernières années, les travaux sur la prédiction sensible au temps des valeurs de QoS dans le domaine des Web services a donné lieu à une riche littérature. Les différents articles que nous avons collectés pour notre recherche bibliographique, sont scindés en trois catégories principales comme l'illustre la figure 5.5. Une catégorie d'articles où les auteurs ont appliqué des approches de filtrage collaboratif. Une deuxième catégorie où des approches en série temporelle ont été appliquées. Et finalement, une minorité de travaux où les auteurs ont opté pour des modèles issus du domaine de l'intelligence artificielle tel que les algorithmes génétiques. Nous précisons que certains travaux se sont appuyés sur une approche hybride en combinant plusieurs types d'approches.

Le tableau 5.2 [36] [37] décrit l'ensemble des articles de recherche que nous avons examinés en dressant une comparaison entre les approches utilisées. Le tableau présente les informations générales sur ces articles, telles que le type de publication, journal (J) ou conférence (C), et l'année de publication. En plus de ces dernières, le tableau ci-dessous décrit, plus spécifiquement, le type d'approches utilisée, le domaine d'application relative ainsi que la nature du jeu de données utilisé durant la phase expérimentale pour tous les travaux.

Il ressort clairement que la plupart des approches portent sur les services Web. De plus, nous pouvons observer que certains auteurs traitent le problème de recommandation dans les contextes des Clouds, Internet des objets, et les réseaux, mais avec un nombre moins important.

Nous pouvons constater aussi que la plupart des travaux ont utilisé une approche hybride entre deux, voire trois catégories de méthodes. En statistiques, les approches linéaires classiques basées sur les modèles à moyenne mobile et auto-régressifs, avec leurs variants, ont été les plus utilisées. Nous avons remarqué que la plupart de ces approches sont fondées sur les méthodes de séries chronologiques communes, telles que ARIMA, SARIMA, ARMA, ARIMA-GARCH, Holts-Winters, genetic algorithms, etc. Pour plus d'information, ces méthodes sont recueillies et introduites dans (Wagner et al., 2007). En apprentissage automatique, des extensions non linéaires de ces modèles, basées essentiellement sur des réseaux de neurones récurrents, ont été proposées. En filtrage collaboratif, nous remarquons que la plupart de ces travaux se sont basés un FC à base de modèle tels que la factorisation matricielle, les facteurs latents et le partitionnement (ou le clustering).

36. Les statistiques sur les citations sont récupérés depuis Google Scholar, date de consultation 10/10/2021
37. Les détails sur les jeux de données sont présentés dans la section 7.2

Tableau 5.2 – Synthèse d'état de l'art sur les méthodes sensibles au temps.

Article			Approches basées		Facteur	Contexte	Données[37]
Référence	Éditeur	Cit.[36]	Statistiques	App. automatique			
Yan et al. (2022)	J - IEEE	0	ARIMA	SVD	-	EC	WS-D
Hasnain et al. (2020)	J - CMC	2	-	LSTM & GRU	-	SOA	simulés
Chen et al. (2019a)	C - ICAIBD	7	-	stacked LSTM	-	SOA	WS-D
White et al. (2018b)	C - IEEE	32	-	LSTM	-	EC	Autres
Zhang et al. (2018)	C - MATEC	0	-	GRU	-	-	Spécifique
Ding et al. (2018)	J - Elsevier	67	ARIMA	-	-	Cloud	WS-D
Zhang et al. (2017)	C - ICWS	10	MTS	-	-	SOA	multiples
Wang et al. (2016)	J - ACM	74	Lasso	-	Spatio-temporel	SOA	WS-D
Hu et al. (2015)	C - ICWS	43	ARIMA& KF	-	-	SOA	WS-D, PW
Li et al. (2012)	J - Springer	20	ARIMA	-	-	SOA	autres
Amin et al. (2012)	C - ICWS	130	ARIMA-GARCH	-	-	SOA	autres[38]
Godse et al. (2010)	C - ICWS	60	ARIMA	-	-	SOA	-
Syu and Wang (2021)	J - IEEE	3			Survey		
Syu et al. (2018)	J - Springer	6			Survey		
Mezni and Fayala (2018)	J - JWS	6			Survey		
Syu et al. (2017)	J - IEEE	30			Survey		

38. Benchmark of Web service QoS Cavallo et al. (2010)

5.5 Prédiction basée sur la mémoire

Alors que de récentes recherches sur la prédiction des valeurs de QoS, montrent que les approches basées sur des modèles [39] sont supérieures, en termes de précision des résultats, à celles basées sur la mémoire; une nouvelle compréhension émergente préconise que cela ne garantit pas une expérience efficace et satisfaisante pour les différents utilisateurs. Selon (Herlocker et al., 2004), une prédiction basée sur la mémoire offre plus de sérendipité à l'expérience des utilisateurs. En d'autres termes, ce type d'algorithmes aide généralement l'utilisateur à trouver un service qu'il l'intéresse et qu'il n'aurait probablement pas autrement découvert par lui seul.

Les approches basées sur la mémoire, également connues sous le nom de méthodes de voisinage, sont considérées parmi les premiers algorithmes développés dans le domaine du FC. Ces méthodes sont intuitives et relativement simples à mettre en œuvre, elles ne nécessitent pas un apprentissage coûteux (en termes de temps et d'espace) et elles sont faciles à stabiliser et à mettre à jour lors d'un ajout de nouvelles données. Ces approches se divisent en trois familles principales. La famille de méthodes basées sur le voisinage utilisateur, les approches qui se focalisent seulement sur les services et en dernier lieu les algorithmes hybrides qui sont considérés comme une combinaison des deux premières approches.

Récemment, plusieurs recherches ont été effectuées dans le domaine de FC pour la sélection et la recommandation de services Web. Ces recherches ont abouti au développement d'un très grand nombre d'algorithmes dans ce domaine. Dans un processus de recommandation de services Web via un FC basé voisinage, l'étape de calcul de similarité est considérée comme étant une étape cruciale et fondamentale pour ce processus. A titre d'exemple, les mesures de corrélation de Pearson (Peerzade, 2017) et de Cosinus (Wang et al., 2017b) sont parmi les stratégies les plus utilisées pour le calculer de la similarité. Cependant, certains travaux ont étudié les lacunes et les déficiences de ces dernières en proposant de nouvelles techniques de calcul de similarité, telles que « *Normal recovery* » (Sun et al., 2013) et « *Ratio based* » (Wu et al., 2017a).

Puisque la plupart des travaux pour cette catégorie de méthodes utilisent des métriques de similarité; nous nous sommes basés sur le travail présenté dans (Khojamli and Razmara, 2021). Ce dernier dresse une synthèse complète sur les différentes métriques de similarité présentées en littérature. L'objectif était donc, de trouver le maximum des travaux de base sur la prédiction de la QoS via un FC basée sur le voisinage en utilisant ces différentes métriques.

5.5.1 Voisinage utilisateur

L'idée de base sur le FC fondé sur le voisinage utilisateurs est de sélectionner un ensemble approprié d'utilisateurs afin de les incorporés dans le processus de prédiction. Tandis que le fondement de cette sélection d'utilisateurs similaires est la sélection d'un modèle de calcul de similarité le plus approprié que possible. Pour la suite de cette sous-section, nous détaillerons trois des travaux de base classés par ordre chronologique (allant du plus récent au moins récent).

Les auteurs, dans (Chen et al., 2020), préconisent que lors de la conception du modèle de similarité, trois questions doivent être considérées. (1) *évaluation commune*: les auteurs confirment que le taux de similarité est souvent surestimé dans le cas où les utilisateurs ne sont pas similaires mais qu'ils ont une expérience de QoS plutôt similaire sur les services Web co-invoqués; (2) *étendue des valeurs de QoS*: selon les auteurs, la présence d'une grande plage de valeurs pour l'intervalle de données de QoS entrave considérablement l'exactitude des résultats pour les calculs de similarités. Par exemple, la similarité de cosinus pour les vecteurs de notation $(0.3, 0.4)$ et $(3, 4)$ vaut 1, alors que qu'ils ont des plages de données différentes (plus l'intervalle de valeurs est réduit plus les résultats seront exactes); (3) *Problème de valeurs égales*: les auteurs font référence au fait que les utilisateurs peuvent avoir les mêmes valeurs de QoS pour les différents services invoqués. Dans ce cas, un modèle de calcul de similarité traditionnel, par exemple, la corrélation de Pearson, ne peut pas être appliqué.

39. Ces travaux seront discutés en détails dans le chapitre suivant (section 6.2.1)

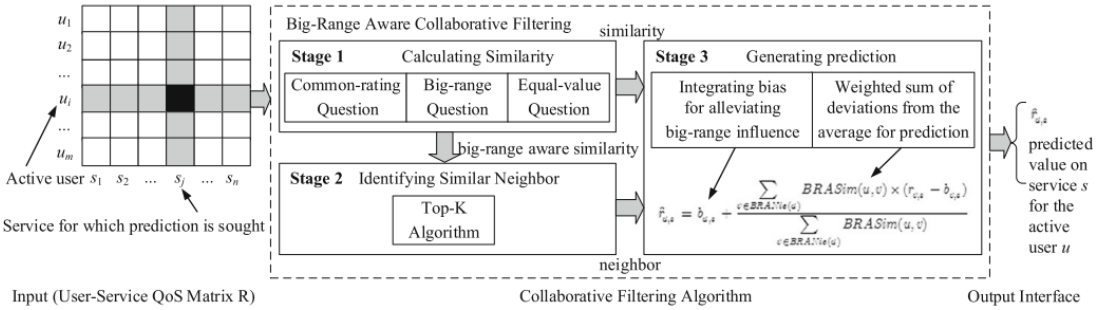


Figure 5.8 – Architecture du modèle BRAFC (Chen et al., 2020)

La figure 5.8 présente l'architecture de leur système baptisé BRAFC (pour *Big-Range Aware Collaborative Filtering*), et qui qui comprend trois étapes: (1) *Calculer la similarité*, tout en respectant les trois considérations présentées précédemment, l'objectif, pour cette étape, est de calculer le degré de similitude entre les différents utilisateurs sur la base des valeurs de QoS connues des services qu'ils ont co-invoqués. L'intérêt ici est double, d'un côté, cette phase permet d'identifier les voisins similaires, de l'autre côté, elle permet de donner une certaine pondération aux voisins dans le processus de prédiction. Il est à noter que la similarité entre les utilisateurs u et v est calculée via la formule suivante:

$$BRASim(u,v) = Jaccard(u,v) \times JacMinMax(u,v) \tag{5.5.1}$$

(2) *Identifier les voisins similaires*, l'objectif est de sélectionner des utilisateurs qui partagent les mêmes modèles de notation avec l'utilisateur actif sur la base de la valeur de similarité calculée entre les différents utilisateurs. Généralement, un algorithme Top-k est utilisé pour identifier ces voisins similaires; (3) *Générer les prédictions*, une fois les deux premières étapes achevées, l'objectif, à présent, est de prédire la valeur QoS sur le service Web cible pour les utilisateurs actifs. Pour cette étape, les auteurs proposent de calculer une somme pondérée des écarts par rapport à la moyenne pour générer les différentes prédictions. Nous retenons que les auteurs ont intégré lors de la phase des calculs les informations de biais des utilisateurs et des services afin d'améliorer la précision de la prédiction.

Les auteurs dans (Wu et al., 2015) constatent que les performances d'un système de prédiction de QoS sont considérablement réduites si des valeurs de QoS et/ou des utilisateurs non fiables ont été utilisés dans ce processus de prédiction. La problématique est donc de déterminer et détecter ces données non fiables, avant d'entamer les calculs des prédictions de valeurs QoS manquantes. C'est dans ce contexte, qu'ils proposent une méthode baptisée CAP (pour *redibility-aware QoS prediction*). La première étape de leur démarche consiste à l'application d'un algorithme de partitionnement à deux phases afin d'identifier les utilisateurs non fiables d'un côté, et d'épurer les données de QoS de l'autre côté. Afin d'éliminer les utilisateurs non fiables, ils appliquent la règle qui stipule qu'un candidat non fiable a tendance à être dans un cluster avec un nombre minimum d'éléments et plus un candidat est fiable, il a tendance a appartenir à plusieurs clusters. La seconde phase s'intéresse à la prédiction des valeurs de QoS manquantes en se basant sur la crédibilité des données qui ont été récupérées lors de la phase précédente.

Enfin, le dernier travail que nous citerons dans cette catégorie de FC est l'approche proposée dans (Shao et al., 2007). Cet algorithme de base est considéré parmi les travaux les plus cités en littérature [40]. Dans leur approche, les auteurs traitent les corrélations positives et négatives des utilisateurs séparément, puis elles sont combinées en une moyenne pondérée. Ils proposent une approche pour la prédiction des valeurs de QoS pour les services Web, sur la base d'une comparabilité effectuée sur les expériences des différents utilisateurs. L'idée fondamentale de cette approche est que les utilisateurs, qui ont presque les mêmes expériences historiques sur certains services, auraient presque les mêmes expériences sur d'autres services. La mise en œuvre de leur méthode consiste en deux phases. En premier lieu, l'algorithme détermine le degré de similarité qui existe entre deux utilisateurs à partir de

40. Un total de 555 citations selon Google Scholar et 241 citations selon IEEExplore. Statistiques consultées le 19 Octobre 2021.

leurs données historiques de QoS. Dans cette étape, les auteurs utilisent la métrique de corrélation de Pearson pour le calcul des similarités. En second lieu, la méthode prédit les valeurs de QoS manquantes pour un utilisateur sur la base des données disponibles des autres utilisateurs qui lui sont similaires.

5.5.2 Voisinage service

Pour les systèmes avec un nombre d'utilisateurs supérieurs à celui des services, les approches basées sur les services peuvent résoudre certains problèmes relatifs aux algorithmes de la catégorie précédente. Avec un plus grand nombre d'utilisateurs que de services, ces derniers ont tendance à avoir plus d'évaluations que celles des utilisateurs, de sorte que l'évaluation moyenne d'un service ne change généralement pas rapidement. Le principe dans cette catégorie d'algorithmes est que les similarités entre les différents services sont d'abord calculées, puis les différentes évaluations par les différents utilisateurs pour un service sélectionné seront prédites en évaluant les valeurs des services similaires.

Dans cette étude (Chen et al., 2019c), les auteurs abordent le problème de prédiction des valeurs QoS, tout en considérant l'influence et l'impact de l'étendu des données QoS sur la performance de cette prédiction (problème déjà expliqué dans la section précédente) dans un contexte de FC. Plus précisément, ils proposent un modèle, à la fois simple et efficace, pour le calcul de similarité entre services qui tient compte de l'étendue des données en termes de QoS, appelé $JacMinMax$. De plus, et après application de cette mesure de similarité, les auteurs proposent deux techniques de sélection de voisinage, une sélection basée sur un algorithme Top-k et une autre basée sur un filtrage à seuil. A la fin, le processus de prédiction des valeurs QoS manquantes sera déclenché, où ces voisins seront systématiquement intégrés, à la fois, dans un modèle de FC à base de voisinage et un autre basé sur une factorisation matricielle.

Dans (Chen et al., 2017b) les auteurs ont observé que l'utilisation des informations spatiales, relatives aux services Web, a une influence significative dans la prédiction des valeurs de QoS. Un phénomène qui a été vérifié empiriquement sur un ensemble de données de QoS du monde réel WS-Dream [41] (Zheng et al., 2013a). Leur hypothèse initiale est comme suit: afin de construire un modèle de prédiction plus précis à base d'une factorisation matricielle, des services Web voisins devront être sélectionnés soigneusement sur des bases d'information de localisation et partitionnés, par la suite, sur des clusters distincts, via un partitionnement hiérarchique ascendant, et qui seront injectés, au final, dans ce modèle.

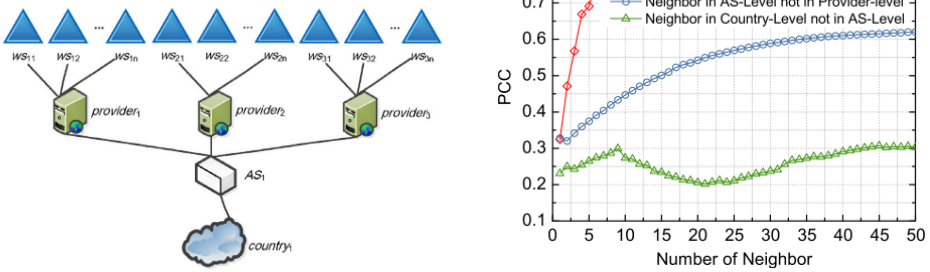
La mise en œuvre de leur démarche consistait en trois étapes successives: (1) Les auteurs analysent empiriquement les caractéristiques spatiales des services Web du point de vue de leur localisation à l'aide d'un ensemble de données public du monde réel. Il a été constaté que la corrélation est toujours positive entre la note d'un service avec la note moyenne de ses voisins, cela quel que soit le niveau de la localisation. (2) Afin de sélectionner « soigneusement » les voisins, les auteurs ont proposé un algorithme de clustering hiérarchique ascendant dans lequel les service sont regroupés selon trois critères de localisation relatifs aux services Web, à savoir, le fournisseur du service, le système autonome [42] où le serveur du fournisseur est logé et en fin du pays où se trouve ce dernier (voir Figure 5.9a). Le clustering est renforcé par l'application d'une mesure de similarité (PCC) afin de filtrer les différents services (voir Figure 5.9b). (3) Les services de chaque cluster sont intégrés d'une façon systémique dans un modèle de factorisation matricielle classique unifié. Ce modèle prédictif est considéré comme étant un problème d'optimisation puis est appris à l'aide d'un algorithme itératif.

En dernier lieu (par ordre chronologique), le travail dans (Sarwar et al., 2001), avec un nombre de citations assez considérable [43], est considéré comme étant l'approche de référence dans cette catégorie de FC. Dans cet article, les auteurs proposent une analyse, assez détaillée, sur les différents algorithmes de génération de recommandations basés sur le voisinage service. Ils examinent les métriques les plus

41. Une présentation de ce jeu de donnée est donnée dans la Section 7.2

42. Wikipédia (date de consultation: 10/5/2021): Un Autonomous System, ou système autonome, est un ensemble de réseaux informatiques IP intégrés à Internet et dont la politique de routage interne est cohérente

43. Un total de 10452 citations selon Google Scholar et 4267 citations selon ACM. Statistiques consultées le 19 Octobre 2021.



(a) Relation hiérarchique entre les niveaux: fournisseur, système autonome et pays.

(b) Corrélation de coefficients de Pearsnon un service et ses voisins par rapport aux différents niveaux.

Figure 5.9 – Analyse empirique des caractéristiques géographiques des services Web présentée par (Chen et al., 2017b)

utilisées pour le calcul de la similarité entre items dans le contexte de FC, à savoir, la *corrélation de Pearson* (O'Mahony et al., 2005; Resnick et al., 1994) et la *similarité du cosinus* (Bao et al., 2013; Breese et al., 1998) et *le cosinus ajusté* (Schafer et al., 2007). Il est important de préciser que les études expérimentales menées par les auteurs, comparant ces trois mesures, ont montré que la métrique du *Cosinus ajusté* est la plus performante en termes de pertinence des résultats de prédiction. Enfin, les auteurs présentent une évaluation expérimentale de leurs résultats avec une comparaison directe avec les approches à base de voisinage utilisateur.

5.5.3 Méthodes hybrides

Dans cette catégorie de méthodes, on traite à la fois les similarités qui peuvent exister entre les utilisateurs et les services. Par la suite, un processus hybride utilise ces deux calculs de similarité pour prédire les valeurs QoS manquantes. Selon (Ma et al., 2007) les algorithmes de FC basés sur un voisinage hybride ont obtenu de meilleurs résultats que les approches individuelles axées sur les utilisateurs ou les services.

Les auteurs dans (Wu et al., 2017b) ont conçu un algorithme de FC pour calculer une similarité relative et filtrer les valeurs des attributs de QoS, améliorant ainsi la précision dans l'identification des voisins. Plus en détail, les principales contributions dans leur travail sont en premier lieu, la proposition d'une méthode basée sur les ratios pour calculer la similarité (RBS: *ratio-based similarity*) entre les utilisateurs ou entre les services. Cette dernière est calculée en comparant directement les valeurs des attributs de QoS comme suit:

$$RBSim(u,v) = \frac{\sum_{i \in \mathcal{I}} (min(r_{u,i}, r_{v,i}) / max(r_{u,i}, r_{v,i}))}{|\mathcal{I}|} \tag{5.5.2}$$

; avec $r_{u,i}$ et $r_{v,i}$ sont les valeurs QoS relatives à l'invocation du service i par, respectivement, les utilisateurs u et v; $min(r_{u,i}, r_{v,i})$ et $max(r_{u,i}, r_{v,i})$ sont la valeur minimale et la valeur maximale de ces invocations. La similarité entre les services est calculée aussi comme suit:

$$RBSim(i,j) = \frac{\sum_{u \in \mathcal{U}} (min(r_{u,i}, r_{u,j}) / max(r_{u,i}, r_{u,j}))}{|\mathcal{U}|}$$

; avec $r_{u,i}$ et $r_{u,j}$ sont les valeurs QoS obtenues après invocation des services i et j par l'utilisateur u; $min(r_{u,i}, r_{u,j})$ et $max(r_{u,i}, r_{u,j})$ sont les valeurs maximales et minimales des QoS. Enfin les ensembles $\mathcal{U}$ sont les utilisateurs qui ont invoqué les services i et j, cependant l'ensemble $\mathcal{I}$ représente tous les services qui ont été invoqués par les utilisateurs u et v.

En second lieu, sur la base des résultats obtenues de l'étape précédente, les auteurs présentent un processus de prédiction de valeurs inconnues de QoS basé sur les méthodes classiques existantes en état-de-l'art.

Les travaux[44] de (Zheng et al., 2009) et (Zheng et al., 2011), abrégés par UIPCC (*User Item Pearson Correlation Coefficient*), sont considérés comme étant le travail de référence pour le FC basé sur un voisinage hybride. Dans ces travaux, les auteurs présentent une approche hybride, baptisée WSRec, combinant le principe des deux approches précédentes, à savoir, le voisinage utilisateur et le voisinage service. Afin de performer les calculs de la similarité, ils ont utilisé une version améliorée de la métrique des coefficients de corrélation de Pearson

$$WSRecSim(u,v) = \frac{2 \times |I_u \cap I_v|}{|I_u| + |I_v|} PCCSim(u,v)$$

; avec $|I_u|$ et $|I_v|$ sont les ensembles de services invoqués par respectivement, les utilisateurs u et v et $PCCSim(u,v)$ est la similarité de Pearson calculer par l'équation 3.3.3. La même équation est appliquée afin de déterminer les similarités entre les différents services. Cependant, pour la prédiction des valeurs QoS manquantes, les auteurs ont intégré deux coefficients appelées *poids de confiance* qui seront calculés en considérant les similarités des voisins similaires. Ce coefficient est calculé via la formule suivante:

$$con_{user}(u) = \sum_{v \in S(u)} \frac{WSRecSim(u,v)}{\sum_{v \in S(u)} WSRecSim(u,v)} \times WSRecSim(u,v) \tag{5.5.3}$$

; avec $S(u)$ est l'ensemble des voisins similaires à u. Il est à noter que la même équation est utilisée pour les services.

5.5.4 Synthèse des approches basées voisinage

Les approches de FC basées sur le voisinage sont connues par leur très grand niveau d'efficacité. En effet, elles ont obtenu un succès généralisé dans des applications réelles. Comme mentionné précédemment, ces algorithmes sont scindés en trois catégories, par rapport à des techniques basées soit sur l'utilisateur, le service ou soit sur une hybridation entre les deux (voir Figure 5.10).

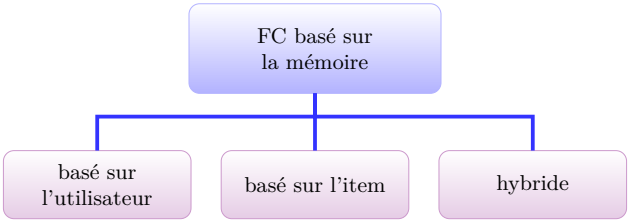


Figure 5.10 – Classification des travaux basés sur la mémoire

L'idée pour la première catégorie d'algorithmes est fondée sur la sélection de l'ensemble des utilisateurs similaires à l'utilisateur actuel afin de prédire la qualité de service des services candidats. Tandis que dans la deuxième catégorie, les similarités entre les différents services sont calculées, puis l'évaluation de l'utilisateur pour un service donné sera prédite en évaluant les valeurs des services similaires à celui-ci. Il est important de mentionner que dans les systèmes qui ont plus d'utilisateurs que de services ces algorithmes sont plus adaptés et peuvent résoudre certains des problèmes des algorithmes précédents. Dans une approche hybride, les similarités entre les services ainsi qu'entre les utilisateurs sont calculées. Ensuite, un algorithme hybride, utilisant ces deux similarités, prend en charge la prédiction des valeurs de la QoS.

44. Les deux articles cumulent un total de 19408 citations selon Google Scholar et 738 citations selon IEEExplore. Statistiques consultées le 19 Octobre 2021.

En analogie avec la synthèse sur les méthodes basées sur les séries chronologiques (Section 5.4.3), le Tableau 5.3 [45] présente un recueil de publications scientifiques que nous avons collecté et examiné. Nous remarquons que ces méthodes utilisent une panoplie de métriques de similarité. Nous remarquons aussi, que la plupart d'entre elles utilisent un FC à base de voisinage hybride. Et afin d'améliorer la précision des prédictions, nous constatons que divers types de facteurs sont employés par ces méthodes, tels que, la réputation, la localisation et le temps.

5.6 Conclusion

Dans ce chapitre, nous avons examiné les différentes approches actuelles de l'état de l'art pour la prédiction de la QoS. L'analyse s'est portée sur deux catégories de méthodes, à savoir, le FC basé sur les séries temporelles, et celui basé sur le voisinage. Mais avant cela, et en début du chapitre, nous avons présenté notre processus de sélection des publications scientifiques qui nous facilitera la tâche pour dresser un état-de-l'art autour de la prédiction de la QoS. Nous avons enchaîné, ensuite, par la présentation de notre classification de méthodes de prédiction de la QoS à base de filtrage collaboratif que nous avons adopté dans cette thèse.

La première partie du chapitre suivant, fera la continuité avec les deux catégories de méthodes, à savoir, les approches de FC basées sur les modèles ainsi que les algorithmes hybrides. Tandis que le reste du chapitre sera consacré à la présentation de l'architecture de notre approche proposée ainsi que nos différentes contributions pour une prédiction de QoS plus efficace.

45. Les statistiques de citations sont récupérés depuis Google Scholar, date de consultation 22/10/2021

Tableau 5.3 – Synthèse d'état de l'art sur le FC basé voisinage.

Article			User	Service	Similarité	Facteur	Contexte	Données
Référence	Éditeur	Cit.[45]						
Zhang et al. (2021a)	J - IEEE	17	✓	-	RBS (eq.5.5.2)	-	Cloud	Autres
Shrivastava and Sharma (2021)	C - ICIPTM	0	✓	✓	RBS	-	SR	GP
Ngaffo et al. (2021)	J - Springer	3	✓	-	KRCC [46], Jaccard	Réputation, Temps	Cloud	WS-D
Chen et al. (2020)	J - Springer	5	✓	-	BRASim (eq. 5.5.1)	-	SOA	WS-D
Song (2020)	J - PLOS	0	✓	✓	Euclidienne	-	SOA	WS-D
Qi et al. (2019)	J - Elsevier	108	✓	✓	LSH	Temps	SOA	WS-D
Wang et al. (2019a)	C - Springer	1	✓	-	Cosinus	-	Cloud	Autres
Kalaï et al. (2018)	J - Elsevier	49	✓	-	Jaccard	Réputation, Temps	SOA	Autres
White et al. (2018a)	C - IEEE	36	✓	✓	PCC	-	IoT	WS-D
Zou et al. (2018)	J - Springer	17	✓	✓	RBS, PCC	-	SOA	WS-D
Kuang et al. (2018)	J - MDPI	43	✓	✓	PCC	Spatial, Réputation	CPS	WS-D
Wu et al. (2017b)	J - IEEE	76	✓	✓	RBS	-	SOA	WS-D
Qi et al. (2017)	J - ACM	8	✓	-	Cosinus Ajustée	Réputation	SOA	Autres
Liu et al. (2016)	J - IEEE	127	✓	✓	PCC	Spatial, Personnalisation	SOA	WS-D
Ma et al. (2016)	J - IEEE	91	✓	✓	PCC	-	SOA	Hybride
Saranya et al. (2016)	J - INDJST	31	✓	-	PCC, Jaccard	-	SOA	WS-D
Wenqiang Li et al. (2016)	C - IEEE	16	✓	-	Plusieurs [47]	-	SR	Autres

46. KRCC: Kendall Rank Correlation Coefficient Zheng et al. (2013b)
47. Cos, Jaccard, PCC

Tableau 5.3 – *Synthèse d'état de l'art sur le FC basé voisinage.*

Article			User	Service	Similarité	Facteur	Contexte	Données
Référence	Éditeur	Cit.						
Karim et al. (2015)	C - IEEE	16	-	✓	Jaccard	-	Cloud	Autre
Tang et al. (2015)	J - wiley	34	✓	✓	PCC	Spatial	Cloud	WS-D
Wang et al. (2015)	J - IEEE	117	✓	-	PCC	Réputation	SOA	WS-D
Deng et al. (2014b)	J - Spring	49	✓	✓	PCC	Réputation	SOA	WS-D
Hu et al. (2014)	C - ICWS	69	✓	✓	PCC	Temps	SOA	WS-D
Huang (2013)	J - ACM	28	✓	-	-	Réputation	Cloud	Autres
Sun et al. (2013)	J - IEEE	103	✓	✓	Normal Recovery	-	SOA	WS-D
Cao et al. (2013)	J - Springer	99	✓	✓	PCC	Personnalisation	SOA	WS-D
Qiu et al. (2013)	C - ICWS	61	✓	✓	PCC	Réputation	SOA	WS-D
Mingdong Tang et al. (2012)	C - ICWS	209	✓	✓	PCC	Spatial	SOA	WS-D
Zheng et al. (2009)	C - ICWS	572	✓	✓	PCC	-	SOA	Autre
Shao et al. (2007)	C - ICWS	556	✓	-	PCC	-	SOA	Autre
Ghafouri et al. (2020)	J - IEEE	11	Survey					
Zheng et al. (2020)	J - IEEE	11	Survey					

Chapitre 6

Prédiction de QoS par apprentissage profond

Sommaire

6.1 Introduction 87
6.2 Prédiction de la QoS par FC: état de l'art (Partie 2) 88
6.2.1 Prédiction basée sur les modèles 88
6.2.2 Prédiction basée sur les modèles hybrides 95
6.2.3 Avantages et limites des approches de FC 100
6.3 Estimation de la QoS par apprentissage profond 101
6.3.1 Les motivations sur nos choix de conception 102
6.3.2 Architecture du modèle 103
6.3.3 Partitionnement de données 105
6.3.4 Prédiction de la QoS 107
6.4 Problème du démarrage à froid 110
6.5 Conclusion 112

6.1 Introduction

Le filtrage collaboratif permet de prédire les évaluations que peut attribuer un utilisateur à un ensemble de services Web, en tenant compte des évaluations données par les autres utilisateurs à cet ensemble de services. Comme mentionné dans la classification précédente (Section 5.3.2), le FC est généralement décomposé en trois catégories principales: FC basé sur les séries chronologique, FC basé sur la mémoire, et celui basée sur un modèle. Une quatrième catégorie est ajoutée, et qui représente un FC hybride entre au moins deux méthodes appartenant à ces différentes catégories.

Dans le chapitre précédent, nous avons dressé un état-de-l'art sur les travaux autours de la prédiction de la QoS, en se focalisant seulement sur les approches de FC basées sur le voisinage (Section 5.5), ainsi que les approches à base de séries temporelles (Section 5.4). Dans la première partie du chapitre actuel, nous présentons un état-de-l'art sur les deux catégories de FC qui nous restent de notre classification, à savoir, le FC basé sur les modèles ainsi que l'hybride.

Dans la deuxième partie du chapitre, nous nous intéressons à la présentation de notre approche de solution. En effet, nous décrivons notre environnement de base, que nous avons mis en place, pour préciser la portée de nos contributions. Ensuite, nous mettons en évidence les motivations concernant les choix effectués pour la conception de ce modèle d'architecture. Enfin, nous présentons en détail tout le processus de prédiction de la QoS que nous avons adopté en étalant les différents algorithmes que nous avons proposés.

6.2 Prédiction de la QoS par FC: état de l'art (Partie 2)

6.2.1 Prédiction basée sur les modèles

Dans l'ensemble, les travaux présentés dans les catégories d'approches précédentes sont relativement simples à mettre en œuvre et donnent des résultats satisfaisants lorsque la matrice d'évaluation n'est pas creuse. Cependant, ces algorithmes présentent plusieurs problèmes qui ont un impact direct sur la précision des résultats de la prédiction, tels que, le problème de rareté de données, le démarrage à froid, l'évolutivité ou le passage à l'échelle et la réputation sur les valeurs de QoS (Kuang et al., 2018; Xu et al., 2016a). Par conséquent, les approches de filtrage collaboratif fondées sur les modèles, sont introduites, principalement, pour résoudre ces différents problèmes.

Un FC basé sur un modèle est très efficace pour prédire des valeurs de QoS manquantes. En effet, cette efficacité de prédiction repose sur l'hypothèse d'existence d'un nombre d'associations locales entre les différentes données de QoS du système. Ce type de méthodes a pour objectif d'établir un modèle représentant une image réduite de la matrice d'évaluation. Contrairement à la famille de FC précédente, où les évaluations des utilisateurs sont directement impliquées dans le processus de prédiction; les approches basées sur les modèles, utilisent plutôt ces évaluations pour apprendre un modèle prédictif. Ce dernier sera utilisé par la suite pour un processus de prédiction. Dans cette partie du chapitre, nous présentons les algorithmes et techniques de prédiction de la QoS des services Web basés sur des modèles.

Méthodes de partitionnement

La méthode de partitionnement (ou *clustering*) fait référence à un procédé dans lequel des données similaires sont regroupées dans des partitions séparées. Il existe de multiples algorithmes et méthodes pour effectuer une tâche de partitionnement. Dans ce contexte, un nombre important d'algorithmes de filtrage collaboratif est basé sur le principe du partitionnement.

Ces stratégies, qui s'appliquent, soit sur l'ensemble des utilisateurs, soit sur les services, permettent d'atténuer l'impact de la rareté des données en réduisant la quantité de données à manipuler, ce qui aura un impact direct sur le volume des calculs à effectuer. Le regroupement améliore l'évolutivité des méthodes de prédiction en réduisant l'espace de données des services et des utilisateurs. De plus, ces méthodes ont une meilleure réponse, en termes de solution, pour le problème du démarrage à froid. Ceci est dû au fait qu'un nouvel utilisateur et/ou service peut facilement être attribué à une des différentes partitions déjà créées. Il est important de noter que souvent, les algorithmes de partitionnement sont utilisés comme une étape d'analyse intermédiaire, dans un processus de prédiction de QoS. C'est la raison pour laquelle ils sont généralement utilisés en combinaison avec une autre méthode de filtrage collaboratif. Aussi, ces méthodes se basent sur plusieurs critères relatifs aux caractéristiques des utilisateurs et/ou services, tels que les facteurs temporels, de localisation et la réputation.

Méthodes de factorisation matricielle

La factorisation matricielle (MF), ou le modèle à facteurs latents, est l'approche la plus utilisée pour la prédiction de la QoS parmi les diverses approches basées sur les modèles. Elle est réputée comme la méthode la plus efficace dans les systèmes de prédiction/recommandation ces dernières années (Koren et al., 2009; Xu et al., 2013a). La principale idée derrière l'extraction des variables latentes est que les techniques de réduction de dimensionnalité arrivent à exploiter certaines informations, relatives aux préférences de l'utilisateur, qui sont dissimulées dans les données brutes et que les autres techniques n'arrivent pas à les faire extraire.

La mise en œuvre de tel modèle consiste à entraîner un modèle en se servant des valeurs QoS disponibles dans la matrice d'évaluation utilisateur-service (c'est-à-dire les valeurs QoS historiques apportées par différents utilisateurs) pour prédire les valeurs de QoS manquantes dans cette matrice. Ainsi, le problème de factorisation matricielle est considéré comme un problème d'optimisation qui consiste à décomposer la matrice d'évaluation utilisateur-service $\mathcal{R} = \{r_{i,j}\} \in \mathbb{R}^{m \times n}$ en un produit de deux matrices réduites. La première est la matrice de caractéristiques des utilisateurs $U \in \mathbb{R}^{d \times m}$,

de sorte que m est le nombre d'utilisateurs. La seconde concerne la matrice de caractéristiques des services $S \in \mathbb{R}^{d \times n}$, où n est le nombre de services. La variable d, qui doit être largement inférieure à m et n, représente le nombre de facteurs latents. La reconstruction de la matrice initiale estime la correspondance entre les caractéristiques de l'utilisateur et du service: $\mathcal{R} \approx \hat{\mathcal{R}} = U^T \times S$.

La factorisation matricielle a pour objectif de résoudre le problème de minimisation définit par la fonction objectif 3.3.13, où une adaptation de celle-ci est donnée par (Xu et al., 2016b), comme suit:

$$\mathcal{L}(U, S) = \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} (r_{i,j} - \hat{r}_{i,j})^2 + \frac{\lambda_u}{2} ||U||_F^2 + \frac{\lambda_s}{2} ||S||_F^2 \tag{6.2.1}$$

où:

— $I_{i,j}$ est un indice qui vaut 1 si la valeur $r_{i,j}$ existe, sinon il est à 0;

— λ_u et λ_s sont des paramètres de régularisation introduits pour éviter le problème du sur-apprentissage;

— $||.||$ est la norme de Frobenius.

Le principal avantage de cette technique est qu'elle permet de résoudre, en partie, les problèmes liés à la rareté et à l'évolutivité des données (Salakhutdinov and Mnih, 2009).

Plusieurs travaux ont été proposés en littérature, dans lesquels divers modèles de FM, tels que la factorisation matricielle probabiliste (PMF), la décomposition en valeurs singulières (SVD) et l'analyse en composantes principales (ACP), peuvent être utilisées pour prédire les valeurs de QoS. Nous citons certains de ces travaux dans la partie suivante.

Ainsi, le travail dans (Xu et al., 2016b) part de l'hypothèse que les services fournis par le même prestataire de services sont susceptibles de partager le même environnement de fonctionnement et les mêmes ressources, telles que, les performances de calculs du serveur, la bande passante du réseau, la capacité de stockage, etc. Ils ont proposé d'utiliser un modèle de factorisation matricielle d'ensemble linéaire, nommé LE-MF, en combinant les informations de localisation du côté de l'utilisateur ainsi que celui du service. Dans le cas où le service n'a pas suffisamment de voisins, il sélectionne au hasard un deuxième ensemble de voisins pour le service en question à partir de tous les fournisseurs de services qui se trouvent dans le même pays du premier service.

Les auteurs dans (Yao et al., 2018) utilisent une technique de FM probabiliste combinée à une stratégie de régularisation de corrélation implicite pour suggérer des services candidats dans le contexte de la composition de service Web. Ils ont utilisé un modèle de variables latentes pour découvrir les différentes corrélations entre les services en analysant le modèle de co-invocation.

Dans la référence (Wu et al., 2018a), les auteurs proposent un modèle de factorisation matricielle unifié sensible au contexte, en termes de la configuration des serveurs, l'état du service et les conditions Internet ainsi que les utilisateurs et les services. Ils partent de la constatation que l'intégration et la fusion de facteurs spatio-temporels pour les méthodes existantes, reste un problème difficile à résoudre, en raison de leurs limites inhérentes d'évolutivité et de flexibilité; ce qui se traduit par des performances moins importantes. De ce fait, ils proposent une approche générale de factorisation matricielle sensible au contexte, baptisée CSMF, qui modélise simultanément les interactions qui peuvent exister entre les utilisateurs et les services ainsi qu'entre les différents environnements, tout en exploitant, au maximum, les facteurs contextuels implicites et explicites. En pratique, un service sera sollicité par différents usagers, et les utilisateurs ayant des modèles de comportement similaires peuvent être regroupés en termes de services similaires. Tandis que la collaboration entre services est utilisée pour modéliser les interactions comportementales des utilisateurs entre les services.

Les algorithmes de prédiction basés sur la factorisation matricielle présentent de nombreux avantages, tels que, leur grande précision dans les résultats de prédiction, leur meilleure extensibilité ainsi que leur grande flexibilité. Ils permettent de transformer les caractéristiques des utilisateurs ainsi que des services en espaces latents implicites. La prédiction s'effectue, ensuite, en calculant la corrélation sémantique implicite entre ces différents utilisateurs et services. La décomposition en valeurs singulières (SVD) est considérée comme la première méthode à être appliquée à l'algorithme de recommandation basés sur le filtrage collaboratif (Billsus et al., 1998). Cette technique est caractérisée par sa simplicité

de mise en œuvre. Toutefois, elle présente deux inconvénients majeurs relatives, premièrement, aux valeurs manquantes dans la matrice d'évaluation et qui doivent être remplies à l'avance. En deuxième lieu, le processus de décomposition lui-même prend beaucoup de temps, en particulier pour les matrices à grande échelle. La technique de facteurs latents via l'utilisation de la factorisation matricielle non négative (NMF) répond parfaitement aux contraintes de non-négativité. En effet, la non-négativité est une propriété importante dans le contexte des données de QoS (Guan et al., 2012).

D'une manière générale, le principal avantage des techniques basées sur les facteurs latents est leur faculté à gérer les problèmes de la rareté des données et du démarrage à froid, par rapport aux autres méthodes, telles celles basées sur la mémoire. Pour cette stratégie, l'estimation des valeurs de la QoS est anticipée par un processus d'apprentissage appliqué sur un modèle construit à partir des valeurs historiques existantes. Cependant, malgré les avantages susmentionnés, ces techniques souffrent de certains problèmes (Chen et al., 2014). En supposant qu'un nouvel utilisateur ou un nouveau service soit ajouté à l'ensemble initial, le modèle doit être régénéré et entraîné à nouveau. Par conséquent, le coût de réactualisation du modèle pour ces méthodes peut être très considérable. En outre, la phase d'entraînement dans les méthodes de FM, et qui n'existe pas dans certaines techniques, représente un autre inconvénient majeur en termes de coût et de temps. A cela s'ajoute le problème de paramétrage, où certains paramètres doivent être définis de manière appropriée car leurs valeurs ont un effet direct et significatif sur la précision de la prédiction. Le dernier inconvénient des modèles à facteurs latents est que ces méthodes n'ont pas la capacité d'expliquer ou d'interpréter les résultats de la prédiction. Bien que les méthodes à facteurs latents puissent traiter efficacement de grandes matrices d'évaluation utilisateur-services, elles ne parviennent toujours pas à résoudre le problème du démarrage à froid.

En résumer, et bien que ces méthodes aient amélioré la précision de la prédiction sous un ensemble d'aspects, les méthodes de FC existantes basées sur la MF ont toujours une limitation selon laquelle seules les caractéristiques explicites sont extraites ou apprises. De plus, certaines de ces approches ne sont pas en mesure d'apprendre les caractéristiques profondes des utilisateurs et/ou services.

Méthodes basées sur l'apprentissage automatique

Récemment, une nouvelle catégorie de méthodes de filtrage collaboratif basée sur les modèles est apparue. En effet, les techniques d'apprentissage automatique, en générale, et d'apprentissage profond, en particulier, sont actuellement très exploitées dans le contexte de FC pour la prédiction de la QoS. Cette catégorie d'approches considérée comme une alternative efficace, et à laquelle notre approche appartient, a pour objectif de capturer, via un processus d'apprentissage, la fonction d'interaction non linéaire à partir de l'ensemble de données.

Contrairement à la grande quantité de travaux relative aux méthodes de factorisation matricielle, il existe, relativement, peu de travaux autour de l'utilisation des réseaux de neurones profonds (DNN) pour la prédiction de la QoS dans le cadre des architectures orientées services. Ainsi, selon notre recherche bibliographique, les travaux de (He et al., 2017) sont considérés parmi les premières tentatives qui ont appliqué des techniques d'apprentissage profond pour les systèmes de recommandation. Il s'agit d'une approche avancée d'apprentissage profond qui combine une architecture de MLP et MF pour la prédiction de la qualité de service.

Les auteurs dans (Yin et al., 2020) proposent un modèle de factorisation matricielle (MF) intégrant un réseau de neurones de type CNN. L'idée de base derrière l'utilisation d'un réseau d'apprentissage profond est de pouvoir déduire les caractéristiques cachées qui peuvent exister entre les différents utilisateurs ainsi que les services. Leur travail s'inscrit dans le contexte de la prédiction de la QoS dans l'informatique en périphérie de réseau (Edge computing). Leur approche, baptisée JCM, a pour objectif de créer deux matrices de caractéristiques relatives, respectivement, aux l'utilisateurs et aux services. Ces dernières seront intégrées dans un processus de factorisation matricielle pour réaliser la prédiction, comme l'illustre la Figure 6.1.

La phase d'apprentissage de ces deux matrices suppose l'utilisation de deux mesures de similarité. En effet, partant du principe que les métriques traditionnelles, telle que la distance euclidienne, souffrent du problème de surestimation; les auteurs proposent, une mesure de similarité basée sur la fréquence inverse d'invocation des utilisateurs (IIFU), pour *Inverse User Invocation Frequency* et

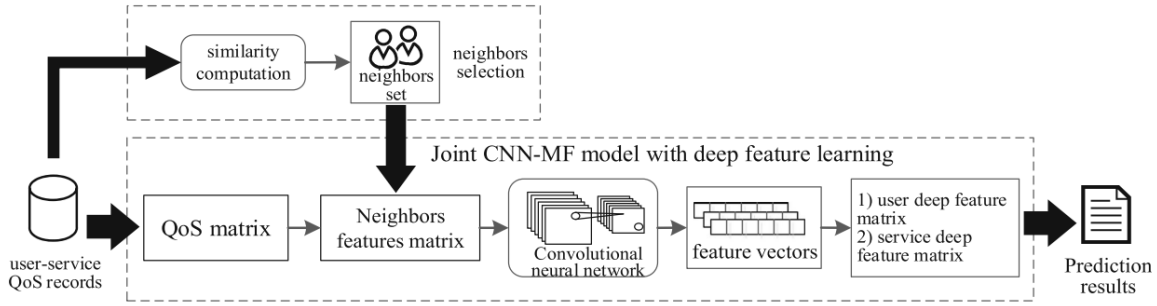


Figure 6.1 – Framework pour la prédiction de la QoS du modèle JCM Yin et al. (2020)

une deuxième basée sur la fréquence inverse d'invocation des services (IIFS), pour *Inverse Service Invocation Frequency*. Cela permettra de sélectionner les différents ensembles de voisins les plus similaires (pour les utilisateurs et/ou services). Dans leur modèle, le processus de convolution consiste à apprendre des caractéristiques profondes à partir de ces ensembles de voisins, et à construire, ensuite, les matrices de caractéristiques correspondantes.

L'approche basée sur l'apprentissage profond proposée dans (Zhang et al., 2019c), parte du principe que le CF traditionnel n'exploite généralement que les interactions linéaires avec de faible dimension entre les utilisateurs et les services; et il est souvent confronté au problème de la rareté des données. Pour résoudre ces problèmes, cet article propose, grâce à l'apprentissage profond, un nouveau modèle de FC pour la recommandation de services Web, baptisé LDCF pour *location-aware deep CF*. En se basant sur les données spatiales, les auteurs présentent un modèle d'apprentissage profond via l'utilisation d'un perceptron multicouche entièrement connecté conçu pour apprendre les interactions non linéaires et à haute dimension. Pour plus de précision dans les performances de la prédiction, les auteurs ont intégré un correcteur adaptatif (AC) de similarité, conçu pour déterminer le degré de similitude, via l'intégration des caractéristiques de localisation, entre les utilisateurs et les services. En effet, l'AC a pour objectif d'incorporer les différents calculs de similarité entre les utilisateurs et les services dans le processus de propagation vers l'avant dans le réseau de neurones. Les auteurs préconisent que l'AC est adaptatif aux diverses métriques de calcul de similarité et permet d'ajuster les valeurs de prédiction.

Les travaux des auteurs dans (White and Clarke, 2020; White et al., 2019) sont réalisés dans le cadre de l'informatique en périphérie de réseau (ou *Edge Computing*). En effet, les auteurs proposent un auto-encodeur empilé (ou *Stacked autoencoder*) (Vincent et al., 2010) ou un auto-encoder profond. Ils démontrent comment qu'une telle architecture peut être utilisé pour réduire le temps d'apprentissage par rapport aux algorithmes existants de factorisation matricielle, tout en maintenant la précision de la prédiction. L'auto-encodeur est implémenté avec quatre couches entièrement connectées. La partie codage du réseau est composée de deux couches a 20 et 10 neurones. Tandis que le réseau de décodage a deux couches de 10 et 20 neurones.

Les algorithmes basés sur l'apprentissage automatique et/ou profond ont la capacité d'apprendre des relations non linéaires et complexes qui peuvent exister entre les utilisateurs et/ou les services. Ils ont la faculté d'être généralisés de sorte qu'ils peuvent prédire des données qui ne sont pas perceptibles à partir de ce qu'ils ont appris des entrées initiales. Contrairement aux autres méthodes de prédiction, ces algorithmes n'imposent pas de restrictions sur les données d'entrée, que ce soit dans la phase d'apprentissage ou lors de la prédiction. En fin, Ils ont une grande capacité pour la modélisation des données à haute variabilité. Cependant, ce type de méthodes présent deux inconvénients, à savoir, il est difficile d'extraire des caractéristiques ou d'interpréter les différents résultats. Aussi, ces techniques affichent un temps de calcul très élevé pour leur phase d'apprentissage.

Synthèse des approches basées modèles

Pour les algorithmes de CF basés sur des modèles, l'utilisation intégrale des valeurs de QoS, à partir de la matrice utilisateur-service, pour construire un modèle global, permet de produire de meilleures

estimations sur la structure globale qui concerne tous les utilisateurs ainsi que les services. Dans cette section, nous avons fait un survole sur les approches de prédiction de la QoS des services Web basées sur des modèles.

Le Tableau 6.1 [48] présente une collection de travaux scientifiques, autour des méthodes de FC basées sur les modèles, que nous avons collecté et examiné. Nous constatons que la plupart de ces approches utilisent un FC basé sur la FM ou sur les méthodes d'apprentissage automatique. Les travaux basés sur un FC via des méthodes de partitionnement ne sont pas nombreux car leur utilisation est toujours en association avec d'autres méthodes, c'est-à-dire, elles sont définies comme méthodes hybrides. Nous remarquons aussi, qu'afin d'améliorer la précision des prédictions, la plupart de ces algorithmes associent à leur démarche de divers facteurs, tels que, la localisation, le temps et réputation. En effet sur les 26 articles collectés, 38,46% utilisent les facteurs temporels, le même pourcentage est pour l'utilisation des attributs de localisation, tandis que 7.50% de ces travaux incorporent les facteurs de réputation. Aussi, nous constatons la tendance dans les trois dernières années est beaucoup plus vers les modèles de filtrage collaboratif basés sur l'apprentissage automatique et profond.

48. Les statistiques sur les citations sont récupérés depuis Google Scholar, date de consultation 10/12/2021

Tableau 6.1 – Synthèse d'état de l'art sur le FC basé modèles.

Article			Modèles			Facteur	Contexte	Données
Référence	Éditeur	Cit.[48]	MF	Clust.	ML			
Su et al. (2021)	J - IEEE	1	MF	-	-	Temps	-	Spécifique
Li et al. (2021)	J -IEEE	0	-	-	DNN	-	SOA	WS-D
Wu and Luo (2020)	J - IEEE	5	MF	-	-	-	RS	Autres
Singh et al. (2020)	J - Elsevier	3	-	-	ANN	Temps	SOA	WS-D
Keshavarzi et al. (2020)	J - Springer	7	-	Clust.	-	Temps	Cloud	WS-D
FanJiang et al. (2020)	J - IEEE	1	-	-	AG[49]	Temps	SOA	autres[50]
Luo et al. (2019a)	J - IEEE	125	MF	-	-	Temps	SOA	WS-D
Zhang et al. (2019c)	J - IEEE	60	-	-	MPL	Location	SOA	WS-D
White et al. (2019)	C - IEEE	18	-	-	Deep Autoencodeur	Temps	EC	WS-D
Zhou et al. (2019)	J - Elsevier	12	-	-	Deep ANN	Spatio-temporel	SOA	WS-D
Luo et al. (2019b)	J - IEEE	77	MF	-	-	-	SOA	WS-D
Yao et al. (2018)	J - IEEE	40	MF	-	-			WS-D
Wu et al. (2018a)	J - Elsevier	70	MF	-	-	Location		
Yang et al. (2018)	J - IEEE	9	MF	-	-	Location	SOA	WS-D
Xiong et al. (2018b)	J - Elsevier	69	-	-	DL	-	SOA	Autres
Wu et al. (2018b)	J - IEEE	32	-	-	Deep ANN	Location	SOA	WS-D
Zhu et al. (2017)	J - IEEE	75	MF	-	-	Temps	Cloud	WS-D

49. Algorithme génétique
50. benchmark of Web service QoS Cavallo et al. (2010)

Tableau 6.1 – *Synthèse d'état de l'art sur le FC basé modèles.*

Article			Modèles			Facteur	Contexte	Données
Référence	Éditeur	Cit.	MF	Clust.	ML			
Thinh and Tu (2017)	J - Autre	2	MF	-	-	Réputation	SOA	WS-D
Tian et al. (2017)	J - Springer	41	MF	-	-	Temps	SOA	PW
Xu et al. (2016b)	J - Elsevier	100	MF	-	-	Location		
Tang et al. (2016)	J - IEEE	60	MF	-	-	Location	SOA	Hybride
Xu et al. (2016a)	J - IEEE	70	MF	-	-	Réputation		
Xie et al. (2016)	C - ICWS	21	MF	-	-	-	SOA	WS-D
Xu et al. (2013b)	C - Springer	55	PMF	-	-	Location		
Lo et al. (2012b)	C - ICWS	149	MF	-	-	Location	SOA	WS-D
Zhang et al. (2011b)	C - IEEE	249	MF	-	-	Temps	SOA	WS-D
Zheng et al. (2020)	J - IEEE	11			Survey			
Ghafouri et al. (2020)	J - IEEE	11			Survey			

6.2.2 Prédiction basée sur les modèles hybrides

Une technique de prédiction hybride (Cao et al., 2020) est une combinaison entre deux ou plusieurs techniques différentes. L'objectif est d'améliorer la qualité et les performances de la prédiction, tout en évitant certains inconvénients relatifs à une utilisation des algorithmes de prédiction traditionnels purs; c'est-à-dire sans hybridation. Le but est de tirer parti des avantages de chacune des deux techniques pour obtenir une meilleure performance. Dans certains cas, l'objectif pourrait être aussi de minimiser ou supprimer carrément, les inconvénients d'une technique quand elle est utilisée séparément. Différentes manières d'hybridation sont citées dans la littérature (Jannach et al., 2010):

1. **Hybridation monolithique**: Le modèle monolithique consiste en une permutation entre plusieurs stratégies de prédiction, en intégrant leurs différents aspects dans une seule implémentation d'hybridation. Le but est de modifier le comportement de base de plusieurs méthodes de prédiction pures pour en faire une autre qui peut utiliser les avantages de toutes les autres méthodes. De l'autre côté, en permutant les techniques, l'hybridation monolithique permet d'éviter les insuffisances propres à une technique particulière.
2. **Hybridation parallélisée**: Dans cette approche, les résultats de prédiction de plusieurs algorithmes purs qui fonctionnent de façon indépendante et parallèle, sont consolidés dans le but de produire un seul résultat final. Dans ce cas, chaque service a plusieurs résultats, en termes de prédiction, provenant des diverses techniques utilisées. Les performances générales dans ce type d'hybridation sont obtenues selon différentes techniques, telles que mixte, pondéré, commuté ou le vote à majorité.
3. **Hybridation en cascade**: Cette approche d'hybridation consiste à une application d'une succession de techniques de prédiction, de telle sorte que, la sortie, en termes de résultats, de chaque technique est considérée comme une entrée à celle qui la suit. Cette stratégie est considérée comme très efficace car dans ce processus les résultats de prédiction vont être affinés pour chaque itération.

Dans la prédiction-recommandation des services Web, plusieurs approches hybrides ont été proposées. Dans notre contexte d'études, nous nous sommes concentrés seulement sur les approches impliquant que les techniques de FC dans leur processus d'hybridation.

Les auteurs dans (Li et al., 2020) proposent un modèle de recommandation de services Web à deux niveaux. Le premier consiste en un processus d'élimination des services invalides générés lors des différentes invocations utilisateur/service. Dans la deuxième étape, les auteurs appliquent un modèle d'apprentissage profond sur le sous-ensemble de services générés lors de la première étape. Afin de raffiner leurs résultats de prédiction, les auteurs ont impliqué les informations spatiales relatives aux utilisateurs ainsi qu'aux services dans cette deuxième étape. Pour les auteurs, l'invalidité fait référence aux différents services qui enregistrent une instabilité remarquable dans leurs valeurs de la QoS. Cette dernière est utilisée comme indicateur pour éliminer les services invalides, ce qui réduit considérablement la taille de la matrice d'évaluation et élimine dans une certaine mesure l'impact négatif des services invalides sur performances de la prédiction. Le modèle d'apprentissage profond proposé est une machine de factorisation pour apprendre les caractéristiques des interactions entre les différents services. Ce modèle est un algorithme supervisé, qui a été introduit en premier lieu par (Rendle, 2010), et qui peut être utilisé dans des taches de classification, régression et de classement. Ce dernier est consolidé par un mécanisme d'attention qui est une technique qui imite l'attention cognitive (Niu et al., 2021), en focalisant la plus grande puissance de calcul possible, du réseau de neurones profond, aux parties importantes des données (qui sont généralement petites). De différentes expérimentations ont été effectuées sur le jeu de données WS-D [51], où 70% des données ont été utilisées pour la phase d'apprentissage, 20% pour la validation et 10% pour les tests.

Les auteurs dans (Su et al., 2017) ont proposé une méthode de prédiction personnalisée de QoS, basée sur la confiance (ou la réputation) des utilisateurs. Pour ce faire, ils ont combiné l'utilisation de l'algorithme de partitionnement K-moyen avec une méthode de distribution de probabilité basée sur la loi Béta. Cette dernière est utilisée pour calculer d'abord les degrés de réputation des utilisateurs. En

51. Le data set WS-Dream sera présenté dans la section 7.2

effet, ils ont utilisé l'algorithme non-supervisé K-moyen pour regrouper les utilisateurs avec les plus grands degrés de confiance formant ainsi la partition la plus « honnête ». Ils ont ensuite calculé les degrés de réputation des différents utilisateurs en évaluant leur écart par rapport au groupe d'utilisateurs honnêtes. Ces degrés sont utilisés, ensuite, comme poids de la matrice de QoS originale. Afin de rendre la prédiction plus précise, une deuxième phase de partitionnement, basée sur l'algorithme K-moyen, est effectuée pour sélectionner les services similaires. Finalement les valeurs de QoS manquantes sont prédites, pour chaque cluster, sur la base de la matrice d'évaluation pondérée.

Les deux travaux cités ci-dessus, résolvent principalement le problème des utilisateurs peu fiables qui influencent négativement sur la précision de la prédiction de la QoS. Cependant, elles ne résolvent pas les problèmes de rareté des données et du démarrage à froid. De plus, pour la première méthode, nous considérons que la taille de la base de test choisie dans la partie des expérimentations, est considérable par rapport à la taille de la base de test; ce qui augmente le risque d'avoir une situation de sur-apprentissage.

Le tableau 6.2 [52] précise pour chaque article que nous avons examiné à partir de l'état-de-l'art, ses catégories d'appartenance, ainsi que les facteurs qui ont été utilisé afin d'améliorer la qualité de la prédiction.

52. Les statistiques sur les citations sont récupérés depuis Google Scholar, date de consultation 15/12/2021

Tableau 6.2 – Synthèse d'état de l'art sur le FC hybride. (Per: Personnalisation, Loc: Localisation, Rép: Réputation, Tps: Temps)

Article			Séries chronolo.		Voisinage		Modèles			Facteur	Données
Référence	Éditeur	Cit.[52]	Stat.	ML	User	Serv.	FM	Clust.	ML		
Sahu et al. (2021)	J - Elsevier	1	-	-	✓	✓	-	-	✓	-	WS-D
Sahu et al. (2021)	J - Springer	0	-	✓	-	-	✓	-	-	-	WS-D
Liang et al. (2021)	J - IEEE	0	-	✓	✓	✓	-	-	✓	-	WS-D
Keshavarzi et al. (2021)	J - Springer	7	✓	-	-	-	-	✓	-	-	WS-D
Zhang et al. (2021b)	J - IEEE	72	-	-	✓	✓	✓	-	-	-	WS-D
Zhu et al. (2021)	J - IEEE	23	-	-	✓	-	✓	-	-	Loc	WS-D
Chattopadhyay et al. (2021)	arXiv	0	-	-	-	-	✓	-	✓	Loc	WS-D
Li et al. (2020)	J - IEEE	3	-	-	✓	✓	-	-	✓	Loc	WS-D
Yin et al. (2020)	J - Springer	173	-	-	✓	✓	✓	-	✓	-	WS-D
Cui et al. (2020)	J - IEEE	110	-	-	✓	-	- ✓	-	-	Per	GP
Zhou et al. (2020)	C - Autres	0	-	✓	-	-	✓	-	-	-	WS-D
Yuan et al. (2020)	C - ECAI	0	✓	-	✓	-	✓	-	-	-	WS-D
Li et al. (2019)	J -	7	-	-	-	-	✓	✓	-	Loc, Rép	Hybride
Wu et al. (2019b)	C - IEEE	7	-	✓	-	-	✓	-	-	-	WS-D
Wang et al. (2019b)	J - Elsevier	10	-	-	✓	✓	✓	-	-	Tps	WS-D
Zhang et al. (2019b)	J - Elsevier	8	-	✓	-	-	✓	-	-	-	WS-D
Chen et al. (2019b)	J - Hindawi	6	-	-	✓	✓	✓	-	✓	Loc	WS-D
Liu and Chen (2019)	J - Elsevier	27	-	-	✓	-	-	✓	-	Rép	WS-D
Chen et al. (2019c)	J - Elsevier	8	-	-	-	✓	✓	-	-	-	WS-D
Jin et al. (2019)	J - autre	8	-	-	✓	✓	-	-	✓	-	WS-D
Anithadevi and Sundarambal (2019)	J - Springer	12	-	-	-	-	-	✓	✓	Loc	WS-D
Wu et al. (2019a)	C - Springer	12	-	-	-	-	✓	✓	-	Loc, Tps	WS-D
Yin et al. (2019)	J - IEEE	49	-	-	✓	✓	- -	-	✓	-	WS-D
Zhang et al. (2019a)	J - IEEE	8	-	✓	-	-	-	✓	✓	-	hybride
Ryu et al. (2018)	J - IEEE	21	-	-	✓	✓	✓	-	-	Loc	WS-D

TABLEAU 6.2 – *Synthèse d'état de l'art sur le FC hybride.*

Article			Séries chronolo.		Voisinage		Modèles			Facteur	Données
Référence	Éditeur	Cit.	Stat.	ML	User	Serv.	FM	Clust.	ML		
Yuan et al. (2018)	J - Springer	30	-	-	✓	-	-	-	✓	Rép	Autres
Xiong et al. (2018a)	C - ICWS	21	-	✓	-	-	✓	-	-	-	WS-D
Li et al. (2018)	J - IEEE	14	✓	-	✓	-	-	-	-	-	WS-D
Ding et al. (2018)	J - Elsevier	67	✓	-	✓	-	-	-	-	-	WS-D
Li et al. (2018)	J - IEEE	14	✓	-	-	-	✓	-	-	-	WS-D
Ren and Wang (2018)	J - Elseiver	59	-	-	-	-	✓	✓	-	-	WS-D
Ding et al. (2018)	J - Elsevier	67	✓	-	✓	-	-	-	-	Tps	WS-D
Chen et al. (2017b)	J - Elsevier	32	-	-	-	✓	✓	-	-	Loc	WS-D
Chen et al. (2017c)	J - Elsevier	24	-	-	✓	✓	✓	✓	-	Loc	WS-D
Xiong et al. (2017)	C -ICWS	11	-	✓	✓	✓	✓	-	-	Loc, Tps	WS-D
Chen et al. (2017a)	C - IEEE	12	-	-	✓	✓	-	✓	-	Loc, Rép	WS-D
Su et al. (2017)	J - Elsevier	82	-	-	✓	✓	✓	✓	-	Rép	WS-D
Wu et al. (2017a)	J - Elsevier	23	-	-	✓	✓	✓	-	✓	Loc	WS-D
Chen et al. (2017d)	J - Sprnger	4	-	-	-	-	✓	✓	-	Loc	
Wu et al. (2017c)	C - Springer	22	-	-	-	-	✓	-	✓	-	WS-D
Luo et al. (2016b)	J - IEEE	204	-	-	-	-	✓	-	✓	-	WS-D
Su et al. (2016)	J - JIFS	23	-	-	-	✓	✓	-	-	-	WS-D
Chen et al. (2016)	C - IEEE	9	-	-	✓	✓	✓	-	-	-	WS-D
Chen et al. (2016)	C - IEEE	9	-	-	✓	✓	✓	-	-	-	WS-D
Yu and Huang (2016)	J - Springer	83	-	-	✓	✓	-	✓	-	Loc, Tps	WS-D
Luo et al. (2016a)	J - Elsevier	68	-	-	✓	✓	-	-	✓	-	WS-D
Wu et al. (2016)	C - ICWS	11	-	-	✓	✓	-	✓	-	Loc, Tps	WS-D
Hu et al. (2015)	C - ICWS	43	✓	-	✓	-	-	-	-	-	Hybride
Wu et al. (2015)	C - ICWS	72	-	-	✓	-	-	✓	-	Rép	WS-D
Lo et al. (2015)	J - Elsevier	48	-	-	✓	-	✓	-	-	Loc	WS-D
He et al. (2014)	C - IEEE	12	-	-	-	-	✓	✓	-	Loc	!

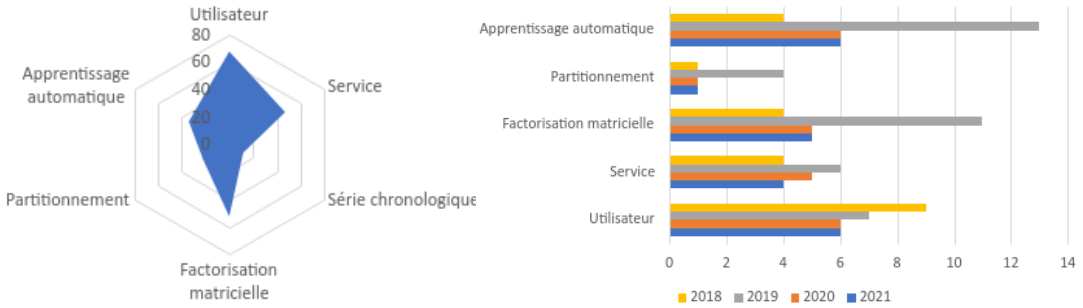
TABLEAU 6.2 – *Synthèse d'état de l'art sur le FC hybride.*

Article			Séries chronolo.		Voisinage		Modèles			Facteur	Données
Référence	Éditeur	Cit.	Stat.	ML	User	Serv.	FM	Clust.	ML		
Yin et al. (2014)	J - Elsevier	60	-	-	✓	-	✓	-	-	Loc	WS-D
Yu et al. (2014)	C - IEEE	55	-	-	-	✓	✓	-	-	Loc	WS-D
Deng et al. (2014a)	J - Elsevier	187	-	-	✓	✓	✓	-	-	Rép	Autre
Chen et al. (2014)	J - IEEE	191	-	-	✓	✓	-	✓	-	Per	WS-D
Xu et al. (2013a)	C - IEEE	19	-	-	-	✓	✓	-	-	Loc	WS-D
Wu et al. (2013)	J - IEEE	216	-	-	✓	✓	-	✓	-	-	WS-D
Lo et al. (2012a)	C - ICWS	103	-	-	✓	✓	✓	-	-	Loc	WS-D
Zhu et al. (2012)	C - ICWS	60	-	-	✓	✓	-	✓	-	-	WS-D
Zhang et al. (2012)	C - IEEE	54	-	-	✓	-	-	✓	-	-	Autres
Kuang et al. (2012)	C - ICWS	53	-	-	✓	-	-	✓	-	Rép, Tps	QWS
Tao et al. (2012)	J - Elsevier	33	-	-	✓	-	-	✓	-	Rép	WS-D
Zhang et al. (2011a)	C - IEEE	180	-	-	✓	✓	✓	-	-	-	WS-D
Chen et al. (2010)	C - ICWS	307	-	-	✓	-	-	✓	-	Loc	W-SD
Syu and Wang (2021)	J - IEEE	3				Survey					
Zheng et al. (2020)	J - IEEE	11				Survey					
Ghafouri et al. (2020)	J - IEEE	11				Survey					
Syu et al. (2018)	J - Springer	6				Survey					
Mezni and Fayala (2018)	J - JWS	6				Survey					
Syu et al. (2017)	J - IEEE	30				Survey					

6.2.3 Avantages et limites des approches de FC

Dans le chapitre précédent, ainsi que la première partie du chapitre actuel, nous avons dressé une synthèse assez détaillée sur les différentes méthodes utilisées dans la prédiction des valeurs de la QoS dans le domaine des services Web, ainsi que d'autres domaines (mais à degré moins). Pour chacune des catégories d'approches, nous nous sommes focalisés sur les approches considérées comme approche de base en littérature. Aussi, nous avons essayé de sélectionner les travaux les plus récents que possible sur l'ensemble des existants.

Avant d'étaler les différents avantages et inconvénients des méthodes de filtrage collaboratif, nous présentons, via le Figure 6.2, une vue statistique sur les différents travaux que nous avons examinés. En effet, sur un ensemble de 130 articles, nous avons un cumule de 236 techniques de FC utilisées [53]. Nous constatons que la technique de FC basé sur le voisinage représente un taux 48,73%, où 28,81% de ces travaux sont relatifs au FC basé sur les utilisateurs et 19,92% pour ceux d'un FC basé sur les services. Aussi, un taux de 22,46% de ces travaux ont opté pour les techniques de factorisation matricielle et 14,41% pour des modèles basés sur des techniques d'apprentissage automatique (voir Figure 6.2a). Tandis que le reste est partagé entre un FC basé sur le partitionnement et les séries chronologiques avec des taux de, 9.32% et 5,08% respectivement. Nous remarquons, aussi, que la tendance dans les articles récents (de 2018 à 2021) est dans l'utilisation des techniques basées sur l'apprentissage automatique et sur la factorisation matricielle, avec des taux de 12,29% et 10,59% respectivement 6.2b.



(a) Total des travaux synthétisés

(b) Évolution d'utilisation des algorithmes de FC

Figure 6.2 – Vue statistique sur les travaux de l'état-de-l'art examinés.

Le tableau 6.3 récapitule les avantages et les limites de ces différentes approches que nous les avons déjà invoqués pour chacune des méthodes à part. Les algorithmes basés sur le voisinage ont une capacité perceptuelle élevée et leurs résultats sont facilement interprétables. Cependant, ils présentent certains inconvénients relatifs aux problèmes de rareté de données, le démarrage à froid et l'évolutivité ou le passage à l'échelle. L'évolutivité ou la *scalability* est souvent considérée comme l'un des principaux facteurs restrictifs pour cette catégorie d'algorithmes. En effet, avec la croissance actuelle du Web, on s'attend à ce qu'il y ait des systèmes avec des millions d'utilisateurs et de services; ce qui rendrait le coût du calcul des similarités, afin de déterminer les différents voisins (pour les utilisateurs et les services), exceptionnellement élevé. Les approches fondées sur des modèles ont été conçues pour résoudre une partie de ces difficultés relatives aux approches précédentes. Elles ont pu surclasser ces dernières et ont montré des améliorations en ayant la possibilité de gérer et traiter efficacement les problèmes du démarrage à froid et de la rareté des données. De plus, cette catégorie d'approches peut également gérer et manipuler un très grand nombre d'utilisateurs et de services. Cette faculté est due aux caractéristiques des facteurs latents qui impliquent qu'il n'y a qu'un petit nombre de valeurs à manipuler pour trouver la similarité entre les différents utilisateurs et/ou services. Toutefois, ces modèles nécessitent un temps d'apprentissage plus important pour générer ces caractéristiques latentes. De plus, ils et doivent être mises à jour et ré-entraînés à nouveau dans le cas où un nouvel utilisateur ou service est introduit dans le système (Chen et al., 2014).

53. Cumule des techniques pour tous les articles.

Tableau 6.3 – Avantages et limites des méthodes de filtrage collaboratif.

Catégorie du FC		Avantage	Limite
Basé sur le voisinage	Basé utilisateur	Simple à mettre en oeuvre	Extensibilité limitée
		Résultats interprétables	Temps de calculs important
		Performances de prédiction élevées si la matrice est dense	Performances réduites avec des données éparses
	Basé service		Il est toujours combiné avec d'autres techniques
	Hybride	Performances plus élevées	Plus de temps de calculs
			Extensibilité plus limité
Basé sur les modèles	Factorisation matricielle	Extensibilité plus élevée	Temps de calculs élevé
		Efficace pour le problème du démarrage à froid	Résultats non interprétables et dépendent des paramètres choisis
		Performances de prédiction élevées si la matrice est sparce	Extensibilité faible
	Partitionnement	Réduire le problème du démarrage à froid	Doit être combiné avec d'autres techniques
		Réduire le volume le temps de calculs	Temps de calculs élevé lors de la création des partitions
		Réduire le volume de données	
		Extensibilité plus élevée	
	Apprentissage automatique	Modéliser des caractéristiques plus complexes	Beaucoup de paramètres à manipuler
		Efficace pour le problème du démarrage à froid	Résultats non interprétables
		Efficace pour le problème de la parcimonie des données (si combiné avec les différents facteurs: temps, location et réputation)	Temps de calculs très élevé dans la phase d'apprentissage
Hybride		meilleure précision de prédiction	Complexité plus élevée pour la mise en œuvre
		Meilleures performances face au problème de la parcimonie de données	

6.3 Estimation de la QoS par apprentissage profond

La prédiction de la QoS constitue une étape cruciale dans le processus de construction d'un système de recommandation des services Web. Dans la pratique, les valeurs de la QoS changent constamment en raison de certaines contraintes environnementales, telles que, l'infrastructure informatique et la charge du réseau, ce qui fait de la prédiction de la QoS une tâche très difficile (Chen et al., 2014). En outre, il est évident que les services disponibles ne seront pas tous invoqués par les différents utilisateurs finaux. Par conséquent, les données, en termes de QoS, sont susceptibles d'être insuffisantes ou rares (Huang et al., 2004), ce qui peut, dans l'ensemble, influencer la précision de la prédiction de la QoS.

Comme nous l'avons présenté dans la section précédente, ainsi que dans le chapitre précédent, de nombreux travaux se sont basés sur le filtrage collaboratif pour résoudre le problème de prédiction de la QoS. Ces approches reposent sur un processus d'exploration des données historiques de QoS des services Web, et qui sont capturées lors des différentes interactions (invocations). Ils utilisent classiquement une structure matricielle, utilisateur-service, pour afficher toutes les données QoS valides. Dans notre thèse, nous nous sommes concentrés spécialement sur les méthodes de FC à base de modèles. En effet, ces d'algorithmes ont la faculté d'apprendre l'ensemble de caractéristiques latentes, qui peuvent excitées entre les utilisateurs et les services, à partir de la matrice QoS pour faire des prédictions sur les valeurs manquantes de cette matrice.

La plupart des études mentionnées précédemment, en particulier ceux basées sur les modèles, sont utiles pour prédire la valeur de la QoS pour les utilisateurs dans une certaine mesure. Mais la plupart de ces méthodes souffrent des problèmes relatifs à la dimension élevée des données, d'une grande complexité temporelle et d'un coût élevé. En effet, nos propositions dans cette thèse, et en plus des problèmes classiques, tels que, la parcimonie des données et le démarrage à froid, traitent ces différents problèmes afin d'améliorer les performances de la prédiction de la QoS.

Dans cette section, nous présentons le modèle de conception de l'approche proposée impliquant des méthodes d'apprentissage en profondeur, des techniques de factorisation matricielle et des facteurs spatiaux. Ensuite, nous mettons en évidence les motivons concernant les choix effectués pour la conception de ce modèle d'architecture. Enfin, nous présentons en détail tout le processus de prédiction de la QoS que nous avons adopté en étalant les différents algorithmes proposés.

6.3.1 Les motivations sur nos choix de conception

Nos différentes contributions sont basées *exclusivement* sur l'utilisation d'un ensemble de modèles issus du domaine d'apprentissage profond. En effet, ces derniers sont connus par leur puissance et leur capacité d'apprendre des caractéristiques profondes. L'objectif est de prédire les valeurs de la QoS des services Web via un filtrage collaboratif à base d'apprentissage profond, en intégrant, dans ce processus, les différentes informations contextuelles disponibles.

Processus de prédiction

Les stratégies de prédiction utilisant les facteurs latents, telle que la factorisation matricielle, utilisent le produit interne entre le vecteur caractéristique de l'utilisateur et celui du service. Ainsi, la méthode considère que la relation entre l'utilisateur et le service est linéaire. Cependant, en réalité, cette relation peut ne pas être uniquement linéaire, et la méthode d'apprentissage profond peut théoriquement s'adapter à toute autre fonction non linéaire.

Dans ce contexte, le modèle des auto-encodeurs est probablement l'un des modèles les plus puissants pour capturer les principales caractéristiques latentes des données. Il est un type de réseau de neurones pour les tâches d'apprentissage non supervisé, tels que, la réduction de la dimension, le codage efficace et la modélisation générative (Goodfellow et al., 2016). Récemment, l'utilisation des auto-encodeurs a amélioré les performances des systèmes de recommandation et a apporté plus d'opportunités dans la reconstruction des expériences des utilisateurs (Anwar et al., 2020; Jiang et al., 2020; Zhang et al., 2020).

Pour une tache de prédiction, l'auto-encodeur aide le système à mieux comprendre la structure entre les utilisateurs et les services en apprenant efficacement la relation de non linéarité qu'existe entre eux et en encodant des abstractions complexes dans les représentations de données en termes de QoS. Par ailleurs, le modèle auto-encodeur peut aussi atténuer l'impact de la rareté des données. Enfin, la capacité à gérer les bruits pour les auto-encodeurs est meilleure que les approches traditionnelles (Li et al., 2015).

Les auto-encodeurs classiques se déclinent souvent en des réseaux d'identité et ne parviennent pas à apprendre les relations latentes entre les données (Glorot and Bengio, 2010). Dans nos différentes contributions dans cette thèse, nous avons abordé ce problème en utilisant, en premier lieu, un auto-encodeur profond à plusieurs couches cachées, combinée avec des entrées corrompues amenant le réseau

à les débruiter en sortie (Vincent et al., 2008). En second lieu, nous combinons l'utilisation des auto-encodeurs avec les réseaux antagonistes génératifs (GAN). En effet, la partie décodage de notre auto-encodeur profond jouera le rôle d'un générateur pour le réseau antagoniste génératif; ceci en faisant correspondre le vecteur des facteurs latents de l'auto-encodeur avec la partie génératrice d'un réseau GAN.

Processus de partitionnement

Dans notre contexte d'études, nous considérons qu'une parcimonie plus élevée entraîne des performances de prédiction moins précises. Pour résoudre le problème de la rareté ou la parcimonie des données, en termes de QoS, ainsi augmenter la précision de notre modèle de prédiction, nous avons appliqué un algorithme de partitionnement sur la matrice d'évaluation originale. L'objectif est d'obtenir un ensemble de matrices de tailles réduites partageant, dans le cas idéal, un sous-ensemble de caractéristiques contextuelles (telles que des informations de localisation ou des fournisseurs de services). Ensuite, un auto-encodeur individuel (et/ou un adversarial auto-encodeur) est formé sur chacune de ces sous-matrices. En effet, nous partitionnons les différents utilisateurs et/ou services en plusieurs groupes en utilisant l'algorithme des cartes auto-organisatrices de Kohonen (SOM). Le choix de cette méthode est justifié par différentes raisons, notamment les suivantes:

— La méthode SOM a la faculté de préserver la densité et la structure topologique (la forme) des données d'origine;

— Le partitionnement basé sur cet algorithme nous sera utile pour faire face au problème du démarrage à froid. Plus précisément, dans le cas d'un nouveau service (ou utilisateur), un traitement initial, sur la base des informations de localisation, est effectué afin d'estimer des valeurs initiales, en termes de QoS, relatives à ce service (ou utilisateur);

— Le réseau SOM assure une représentation plus fidèle (en termes d'attributs contextuels, tels que le pays, le fournisseur et les systèmes autonomes) des services (ou des utilisateurs); autrement dit, les services (ou les utilisateurs) ayant des attributs de localisation similaires sont plus susceptibles d'appartenir à la même partition;

— Enfin, avec un mécanisme d'initialisation adéquat, l'algorithme SOM garantit un comportement déterministe pour la phase du partitionnement.

Influence des facteurs de localisation

Selon les travaux dans (Chen et al., 2017b; Tang et al., 2016), les valeurs de QoS sont directement influencées par les caractéristiques de localisation des utilisateurs ainsi que des services. En effet, plus des spécificités contextuelles semblables sont partagées, plus les valeurs de QoS sont susceptibles d'être similaires. Dans la pratique, les caractéristiques de localisation pour les services Web peuvent être groupées en trois ensembles imbriqués. (1) Le fournisseur du service ($\mathcal{P}$), (2) le système autonome où le serveur du fournisseur est logé ($\mathcal{AS}$), et enfin (3) l'emplacement ou la localisation géographique ($\mathcal{C}$), de telle sorte que $\mathcal{P}(i) \subseteq \mathcal{AS}(i) \subseteq C(i)$, où $i \in \{service, user\}$. Ceci implique que plus le groupe est restreint, plus les valeurs de qualité de service sont susceptibles d'être similaires.

6.3.2 Architecture du modèle

La figure 6.3 représente l'architecture globale de l'approche proposée. Premièrement, nous supposons que notre système est composé de m utilisateurs et n services Web. Les valeurs de QoS sont collectées à partir des invocations des services Web par les différents utilisateurs. La provenance de ces données peut être multiple, i.e. les réseaux sociaux, fournisseur de services, utilisateurs, etc. Ces valeurs collectées constituent une matrice d'évaluation à deux dimensions $m \times n$, nommée $M = \{r_{u,s}\}_{m \times n}$. Nous supposons, aussi, que cette matrice contient des valeurs manquantes car il est impossible d'invoquer tous les services par tous les utilisateurs. Afin de prédire ces valeurs, nous proposons une approche qui se compose de trois palliés que nous allons les décrire ci-dessous.

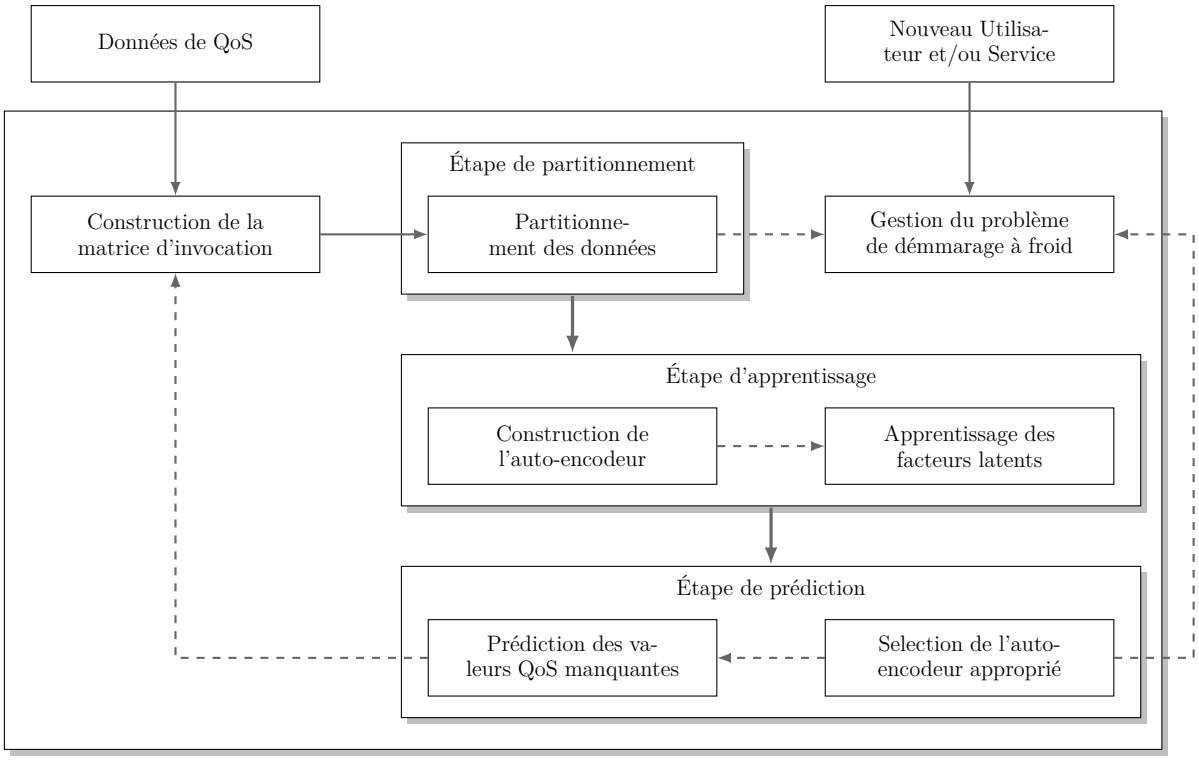


Figure 6.3 – Aperçu de l'architecture du système de prédiction

Partitionnement des données La première étape dans notre approche consiste en une division de la matrice d'invocation initiale par rapport aux attributs spatiaux des utilisateurs et/ou services. Plus précisément, en premier lieu, nous effectuons un partitionnement basé sur l'algorithme SOM afin de créer une cartographie de neurones. Une fois que cette étape soit terminée, nous appliquons l'algorithme K-moyenne [54] sur cette cartographie de neurones afin de générer les différentes partitions d'utilisateurs et/ou de services. Il est important de mentionner que le réseau de Kohonen que nous avons appris nous permettra de créer des partitions dont les membres sont en relation étroite par rapport à leurs attributs contextuels. Cette étape vise à produire un ensemble de partition partageant les mêmes caractéristiques contextuelles. De plus, elle permet de diminuer le problème dû à la densité de données dans la matrice d'évaluation QoS. En effet, au lieu du l'utiliser en intégralité, nous sélectionnons que les sous matrices relatives aux différentes partitions que nous avons générées. L'impact des données spatiales sera largement discuté à la Section 7.3.2 du chapitre suivant.

Apprentissage des modèles La deuxième phase dans notre processus de prédiction de QoS consiste en une phase d'apprentissage des facteurs latents via l'utilisation de plusieurs modèles de réseaux neuronales. En effet, nous avons proposé trois modèles de réseaux de neurones, comme l'illustre la Figure 6.4.

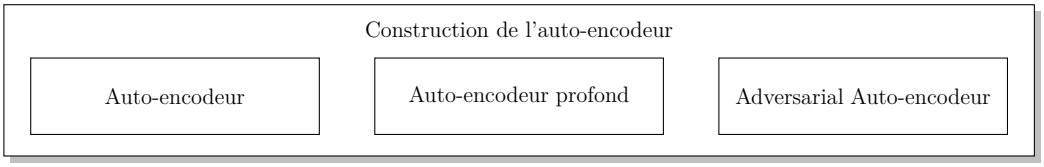


Figure 6.4 – Modèles de réseaux de neurones proposés.

54. K-moyenne est une approche de partitionnement de données (Hartigan and Wong, 1979).

1. La première proposition (Smahi et al., 2018) est un modèle de type auto-encodeur avec une seule couche cachée où l'apprentissage et la prédiction s'effectueront sur l'ensemble de la matrice d'évaluation.
2. La deuxième proposition (Smahi et al., 2021) consiste en une utilisation d'un auto-encodeur à dénotation profonde, nommé *Deep AE*, où l'apprentissage et la prédiction s'effectueront aussi sur l'ensemble de la matrice d'évaluation. Cependant et puisque nous avons effectué un processus de partitionnement sur la matrice d'évaluation initiale, nous appliquons pour chaque partition un *Deep AE* adapté, où la performance globale des résultats de la prédiction correspond à la moyenne pondérée des performances locales relatives à chaque partition. Les choix des différents hyperparamètres ainsi que le processus d'adaptation du *Deep AE* pour chaque partition seront discutés dans le chapitre suivant (Section 7.3.1, 7.3.1 et 65).
3. Un troisième modèle, nommé *Adversarial AE*, consiste en une combinaison entre l'architecture de l'auto-encodeur avec celle d'un réseau antagoniste génératif où le générateur de ce dernier jouera le rôle d'un décodeur. Il est important de préciser que même pour ce modèle l'apprentissage et la prédiction se feront sur, à la fois, toute la matrices d'évaluation et sur les sous-matrices des différentes partitions générées lors de la phase de partitionnement.

Prédiction de la QoS La phase de prédiction consiste en une simple application d'un des modèles proposés afin de prédire les valeurs de QoS manquantes afin de les stockées dans la matrice d'évaluation initiale.

6.3.3 Partitionnement de données

La première étape dans notre contribution pour la prédiction de la QoS est concrétisée par l'Algorithme 3. Il consiste en un traitement de partitionnement qui s'effectue sur l'ensemble des données afin de créer des partitions homogènes d'utilisateurs et/ou de services . L'objectif de cette phase est double. D'un côté il nous permet de réduire la dimensionnalité de la matrice d'évaluation $M = \{r_{u,s}\}_{m \times n}$. De l'autre côté, ce partitionnement nous sera utile pour notre proposition de solution au problème du démarrage à froid qui sera discuté dans la Section 6.4.

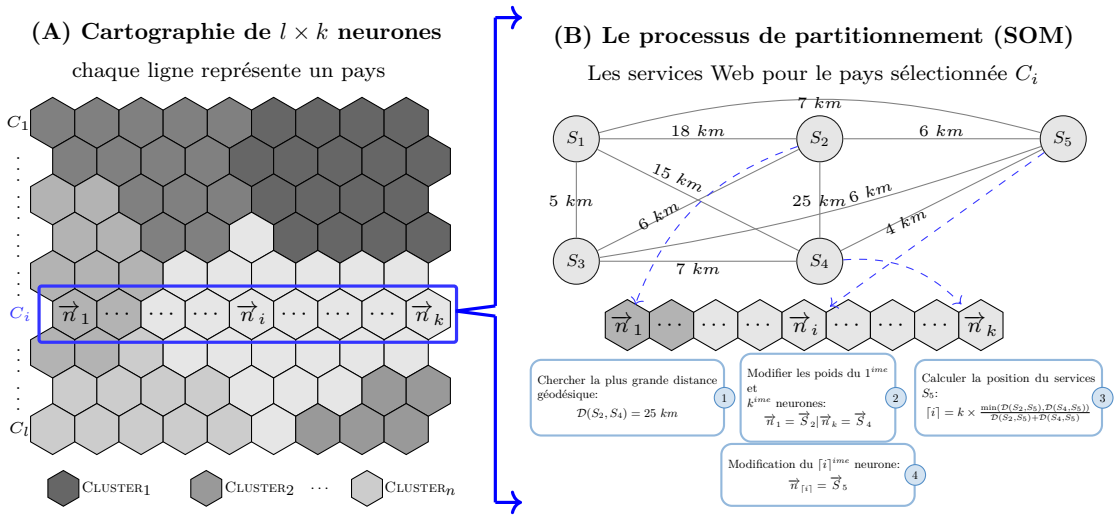


Figure 6.5 – Exemple illustratif pour la phase de partitionnement de données

Dans la première étape (de la ligne 1 à 7), nous initialisons les vecteurs de poids des différents neurones de notre cartographie. À cette fin, nous utilisons un mécanisme d'initialisation personnalisé pour assurer un comportement déterministe et préserver les relations de voisinage entre les différents données (utilisateurs et/ou services). La Figure 6.5 représente un exemple illustratif sur la phase d'initialisation de ce réseau de neurones pour un partitionnement effectué sur les services. En effet,

notre carte auto-organisatrice est composée de l lignes où chacune d'elle correspond à un pays figurant dans notre jeu de données. Nous affectons aux neurones qui se trouvent aux deux extimités d'une ligne donnée C_i le vecteur d'utilisateur (et le vecteur service dans le cas d'un partitionnement effectué sur les utilisateurs). Cette affectation est assurée via l'application de la fonction objectif $argmax(\mathcal{D}(u_1, u_2))$, $\forall u_1, u_2 \in Country_i$ dans le cas où nous effectuons un partitionnement d'utilisateurs et $argmax(\mathcal{D}(s_1, s_2))$, $\forall s_1, s_2 \in Country_i$ (ligne 2) pour le cas contraire. $\mathcal{D}$ désigne une fonction qui retourne la distance géodésique qu'existe entre les deux zones géographiques où les services s_1 et s_2 sont situés. En se référant à notre exemple, la distance maximale trouvée est entre les services s_2 et s_4. Ensuite, pour chaque vecteur de service s_m nous lui cherchons sa position, $\lceil index \rceil$, sur cette ligne via l'application de l'équation de ligne 4. L'intérêt de ce choix sera démontré empiriquement dans le chapitre suivant (Section 7.3.2).

Algorithme 3 : Partitionnement des données avec SOM

Inputs : $S = \{s_1, \cdots, s_n\}$, ▷ L'ensemble des vecteurs de services
$C = \{C_1, \cdots, C_k\}$, ▷ L'ensemble des pays
max ▷ Nombre de partitions à générer
Data : $W = \{w\}_{k,k}$ ▷ Cartographie de $k \times k$ neurones

`/* Phase d'initialisation */`
1 **foreach** $c_l \in C$ **do**
2 $(w_{l,1}(s_i), w_{l,k}(s_j)) = \underset{s_i \in c_l \ et \ s_j \in c_l}{\text{argmax}} (\mathcal{D}(s_i, s_j))$
3 **foreach** $s \in c_l, s \neq s_i, s \neq s_j$ **do**
4 $\lceil index \rceil = k \times \frac{\min(\mathcal{D}(w_{l,1}, s), \mathcal{D}(w_{l,k}, s))}{\mathcal{D}(n_{l,1}, s) + \mathcal{D}(n_{l,k}, s)}$
5 $w_{index} = s$
6 **end foreach**
7 **end foreach**
`/* Phase d'apprentissage */`
8 **for** $epoch : 1 \rightarrow N_{epochs}$ **do**
9 **foreach** $s \in S$ **do**
10 Sélection du meilleur neurone $n(s) = \underset{i \in 1,\ldots,k \ et \ j \in 1,\ldots,k}{\text{argmin}} ||s - w_{i,j}||$
11 **end foreach**
12 **for** $j : 1 \rightarrow k$ **do**
13 Récupérer les paramètres de l'equation 4.5.3
14 **end for**
15 **foreach** $w \in W$ **do**
16 Modification du w selon l'equation 4.5.3
17 **end foreach**
18 **end for**
`/* Construction des classes avec l'algorithme k-means */`
19 $< C_1, \cdots, C_{max} >= KmeansCluster(W, max)$
20 **return** $< C_1, \cdots, C_{max} >$ ▷ La liste des clusters

Après cette étape cruciale d'initialisation, nous déterminons pour chaque vecteur $\mathcal{V}(s) = \{qos_1, \cdots, qos_m\}$ de service en entrée son neurone correspondant sur la carte de neurones (de la ligne 8 à la ligne 11). Dans les étapes suivantes, de la ligne 12 à 18, nous ajustons les vecteurs de poids de toutes les neurones correspondants ainsi que leurs neurones voisins, selon la version *batch* de l'algorithme SOM, via l'application de l'équation 4.5.3. Ce processus est répété autant de fois que nécessaire où selon le nombre d'époques spécifié dans la ligne 8. Une fois que la cartographie des neurones établie nous générons les clusters finaux en appliquant l'algorithme K-moyenne sur notre cartographie finale, en respectant le nombre de partitions que nous voulons générer, et qui est spécifié comme paramètre d'entrée pour cet algorithme.

6.3.4 Prédiction de la QoS

Auto-encodeur profond partitionné

Dans l'Algorithme 4, nous nous lançons dans un processus d'apprentissage afin d'apprendre un modèle obtenant les meilleures performances de prédiction possibles pour les valeurs de QoS manquantes. L'objectif est de déduire le meilleur modèle d'apprentissage, Model$^{k^\star}$, pour notre auto-encodeur profond. Cet algorithme apprend un autoencodeur profond pour chaque cluster $C_i, i \in \{1, \dots max\}$, comme illustré par la Figure 6.6.

En appliquant le principe de la validation croisée à k-plis (ou k-fold cross-validation) (Refaeilzadeh et al., 2016), nous scindons les données de la partition C_i actuelle (la ligne 3) à k échantillons différents. Avec cette technique nous divisons aléatoirement l'échantillon initial en données d'entraînement et celles de test. Pour chaque pli, nous nous limitons à un pourcentage de données spécifié par le paramètre *density*. Cette limite représente la proportion des données sélectionnée pour la phase d'entraînement; par conséquent, le reste sera réservé pour la phase des tests. Par exemple, pour une densité égale $x\%$, nous sélectionnerons aléatoirement ce taux de données pour la phase d'apprentissage et un taux de $1 - x\%$ pour les données qui seront réservées pour la phase des tests.

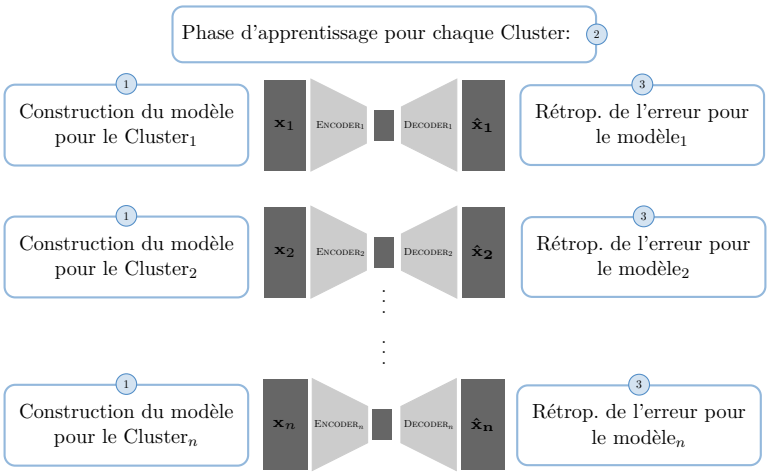


Figure 6.6 – Prédiction de la QoS par l'auto-encodeur profond

Ensuite, dans la ligne 7, et par rapport à tous les ensembles d'entraînement ($\forall S \in T$), nous apprenons un auto-encodeur profond pour chaque pli de la partition en cours. L'objectif de cette phase, est de minimiser l'erreur quadratique qui existe entre les valeurs réelles et celles prédites par l'auto-encodeur.

Il est essentiel de préciser que les données, dans la phase d'entraînement, sont conditionnées par un taux de bruit appliqué sur les données d'entrée de l'auto-encodeur (voir équation 4.4.4). Dans le cas où le taux de bruit est différent de 0% nous passerons d'une architecture d'auto-encodeur profond à celle d'un auto-encodeur profond de débruitage. À la ligne 8, nous déterminons l'erreur de l'auto-encodeur profond (ou de débruitage), obtenue sur le jeu de données de validation. À la ligne 11, nous calculons la moyenne des résultats sur les k-plis afin de produire une seule erreur de validation (mve). Enfin, nous retournons le meilleur modèle d'entraînement ainsi que l'erreur de validation moyenne. À noter que le meilleur modèle trouvé dans cette phase (ligne 10) sera utilisé pour gérer le problème du

démarrage à froid (Section 6.4).

Algorithme 4 : L'apprentissage de l'auto-encodeur profond partitionné

Inputs : $C = \{C_1, \cdots, C_{max}\}, density,\ noise,\ k\text{-}fold, max$

1 **for** $i \rightarrow max$ **do**
2 $\quad T = C_i$ ▷ Data set d'apprentissage
3 $\quad < folder_1, ..., folder_{k\text{-}fold} >= Partition(C_i,\ density,\ k\text{-}fold)$
4 $\quad$ **for** $k : 1 \rightarrow k\text{-}fold$ **do**
5 $\quad\quad T = T - folder_k$
6 $\quad\quad V = folder_k$ ▷ Data set de validation
7 $\quad\quad Model^k = \underset{\forall S_m \in T}{\operatorname{argmin}} \left(\sqrt{\frac{1}{|T|} \sum_{m=1}^{|T|} |DAE(S_m, noise, |C_i|) - S_m|^2} \right)$
8 $\quad\quad err^k = \sqrt{\frac{1}{|V|} \sum_{m=1}^{|V|} |Model^k(S_m) - S_m|^2}$
9 $\quad$ **end for**
10 $\quad Model^{k^\star} = \underset{\forall k \in k\text{-}fold}{\operatorname{argmin}} (err^k)$ ▷ Récupérer le k^{me} meilleur modèle en termes de err_k
11 $\quad mve = \frac{1}{k\text{-}fold} \sum_{k=1}^{k\text{-}fold} err_k$
12 $\quad \Omega[i] =< Model^{k^\star}, mve >$ ▷ Sauvegarde du meilleur modèle
13 **end for**
14 **return** Ω ▷ Retourner tous les meilleurs modèles

A noter que le processus d'apprentissage pour l'auto-encodeur profond (sans introduire le partitionnement) est presque le même que le précédent comme l'illustre l'algorithme 5. En effet, et par rapport à l'algorithme précédent, l'apprentissage s'effectue sur l'ensemble des données en même temps (de la ligne 3 à la ligne 8); et l'algorithme doit retourner le meilleur modèles ainsi que l'erreur d'apprentissage moyenne.

Algorithme 5 : L'apprentissage de l'auto-encodeur profond

Inputs : $C, density,\ noise,\ k\text{-}fold$

1 $T = C$ ▷ Data set d'apprentissage
2 $< folder_1, ..., folder_{k\text{-}fold} >= Partition(C,\ density,\ k\text{-}fold)$
3 **for** $k : 1 \rightarrow k\text{-}fold$ **do**
4 $\quad T = T - folder_k$
5 $\quad V = folder_k$ ▷ Data set de validation
6 $\quad Model^k = \underset{\forall S_m \in T}{\operatorname{argmin}} \left(\sqrt{\frac{1}{|T|} \sum_{m=1}^{|T|} |DAE(S_m, noise, |C|) - S_m|^2} \right)$
7 $\quad err^k = \sqrt{\frac{1}{|V|} \sum_{m=1}^{|V|} |Model^k(S_m) - S_m|^2}$
8 **end for**
9 $Model^{k^\star} = \underset{\forall k \in k\text{-}fold}{\operatorname{argmin}} (err^k)$ ▷ Récupérer le k^{me} meilleur modèle en termes de err_k
10 $mve = \frac{1}{k\text{-}fold} \sum_{k=1}^{k\text{-}fold} err_k$
11 **return** $Model^{k^\star}, mve$ ▷ Retourner le meilleur modèle

Adversarial Auto-encodeur

Pour notre troisième proposition, nous nous sommes inspirés des travaux (Wang et al., 2018, 2017a) [55]. En effet, nous construisons une nouvelle architecture afin d'améliorer les performances de nos résultats de prédiction. Nous combinons l'utilisation de l'architecture des réseaux antagonistes génératifs (GAN) à celle des auto-encodeurs. Comme présenté dans la Section 4.4.2; un réseau GAN confronte deux sous réseaux; l'un est générateur, tandis que le second est discriminateur. L'association du réseau « générateur » avec un « discriminateur » permet de paramétrer au mieux le générateur. Partant de ce principe, l'idée de base est de considérer le générateur du réseau GAN comme étant le décodeur de notre auto-encodeur profond, comme illustrer par la Figure 6.7. Au lieu d'apprendre un auto-encodeur profond, l'apprentissage s'effectuera sur le réseau GAN où le générateur doit tromper le discriminateur qui doit distinguer les vecteurs réels de ceux obtenus par l'auto-encodeur.

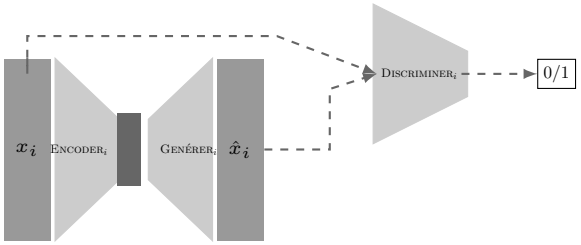


Figure 6.7 – Prédiction de la QoS par l'adversarial auto-encodeur

Dans ce cas de figure, le générateur du réseau GAN jouera le rôle du décodeur de notre auto-encodeur profond. Nous considérons le réseau encodeur comme étant l'inverse du décodeur (c'est-à-dire, il est représenté par l'inverse de la matrice des paramètres du réseau générateur). Ce choix est pris afin de réduire le temps d'apprentissage pour cette contribution. Par conséquent, la phase d'entraînement ne concerne que le réseau GAN, où nous modifions que les poids de ce dernier afin d'atteindre un équilibre de Nash par descente de gradient. Le générateur ainsi que les données réelles (vecteurs des utilisateurs et/ou services) alimentent le réseau discriminateur, et ce dernier produit la sortie que nous essayons d'affecter. La fonction de perte pénalise le générateur pour avoir produit un échantillon que le réseau discriminateur classe comme faux (c'est-à-dire le vecteur $\hat{x}_i$ et largement différent à celui des données réelles x_i).

Nous entraînons alors le générateur selon une suite d'instructions illustrée par l'Algorithme 6. En effet, les paramètres d'entrée de celui-ci sont les mêmes que l'Algorithme 4 (les différentes partitions, le taux de densité de données à considérer, le taux d'erreur à appliquer, le nombre de plis pour la validation croisée et enfin, le nombre maximal d'itérations). Après la détermination des différents plis (ligne 3), nous initialisons, aléatoirement, les paramètres de réseau GAN, à savoir, le générateur (ligne 4) et le discriminateur (ligne 5). Dans la ligne 6, nous initialisons aussi la matrice des paramètres de l'encodeur. L'apprentissage du réseaux GAN se fait sur la fonction objectif représentée par l'équation 4.4.8 (ligne 10 et 11). En effet, pour chaque itération, nous récupérons le vecteur des facteurs latents (la sortie de l'encodeur) afin de produire la sortie du générateur. Par la suite nous devons obtenir une classification discriminante « vrai » ou « faux » pour la sortie du discriminateur. L'étape suivante consiste à calculer la perte à partir de la classification du discriminateur. Une rétro-propagation par montée du gradient s'effectue pour modifier les différents poids du discriminateur. Tandis qu'une descente du gradient s'effectue pour le générateur. Par conséquent les poids du décodeur ainsi que de l'encodeur seront misent à jours. Les instructions qui restent à la fin de l'algorithme, de la ligne 12 à la ligne 16, sont équivalents à celles de l'algorithme de l'auto-encodeur profond (Algorithme 4).

55. Nous notons que ces travaux, ne sont pas dédiés aux tâches de filtrage collaboratif

Algorithme 6 : L'apprentissage de Adversarial auto-encodeur

Inputs : $C = \{C_1, \cdots, C_{max}\}, density,\ noise,\ k\text{-}fold, max$

1 **for** $i \to max$ **do**

2 $\quad T = C_i$ ▷ Data set d'apprentissage

3 $\quad < folder_1, ..., folder_{k\text{-}fold} >= Partition(C_i,\ density,\ k\text{-}fold)$

`/* Initialisation aléatoire des paramètres du réseau */`

4 $\quad (\mathcal{G}_{\Theta_{(w,b)}} \equiv \mathcal{D}\text{ecoder}_\Theta) \leftarrow A \sim \mathcal{U}(0,1)$ ▷ Paramètres du générateur

5 $\quad \mathcal{D}_{\Phi_{(w,b)}} \leftarrow A \sim \mathcal{U}(0,1)$ ▷ Paramètres du discriminateur

6 $\quad \mathcal{E}\text{ncoder}_\Theta \longleftarrow \mathcal{G}_\Theta^{-1}$

7 $\quad$ **for** $k : 1 \to k\text{-}fold$ **do**

8 $\quad\quad T = T - folder_k$

9 $\quad\quad V = folder_k$ ▷ Data set de validation

`/* L'apprentissage du GAN */`

10 $$\mathcal{D}_\Phi^* = \underset{\forall S_m \in T}{\arg} \max_{\mathcal{D}} \min_{\mathcal{G}} \frac{1}{m} \sum_{i=1}^{m} \Big(\log \mathcal{D}_\Phi(S_m) + log(1 - \mathcal{D}_\Phi(\mathcal{G}_\Theta(z^{(i)}))) \Big) \text{ avec}$$
$$z^{(i)} = \mathcal{E}\text{ncoder}_{\Theta^{-1}}(S_i, noise)$$

11 $$\mathcal{G}_\Theta^* = \underset{\forall S_m \in T}{\arg} \min_{\mathcal{G}} \max_{\mathcal{D}} \frac{1}{m} \sum_{i=1}^{m} \Big(\frac{1}{m} \sum_{i=1}^{m} log(1 - \mathcal{D}_\Phi(\mathcal{G}_\Theta(z^{(i)}))) \Big) \text{ avec}$$
$$z^{(i)} = \mathcal{E}\text{ncoder}_{\Theta^{-1}}(S_m, noise)$$

12 $$err^k = \sqrt{\frac{1}{|V|} \sum_{m=1}^{|V|} |\mathcal{D}\text{ecoder}_\Theta(\mathcal{E}\text{ncoder}_{\Theta^{-1}}(S_m)) - S_m|^2}$$

13 $\quad$ **end for**

14 $\quad Model^{k^\star} = \underset{\forall k \in k\text{-}fold}{\text{argmin}} (err^k)$ ▷ Récupérer le k^{me} meilleur modèle en termes de err_k

15 $\quad mve = \frac{1}{k\text{-}fold} \sum_{k=1}^{k\text{-}fold} err_k$

16 $\quad \Theta[i] =< Model^{k^\star}, mve >$ ▷ Sauvegarde du meilleur modèle

17 **end for**

18 **return** Ω ▷ Retourner tous les meilleurs modèles

6.4 Problème du démarrage à froid

La question du démarrage à froid est considérée comme un problème très répondu dans les systèmes de recommandation (Lika et al., 2014; Schein et al., 2002). Il survient lorsque le système peine à fournir des prédictions dû à un manque d'informations, et qui est dans notre cas relatif à l'ajout de nouveaux utilisateurs et/ou de services.

Pour traiter ce problème, et en tenant compte du fait que les valeurs de QoS sont fortement influencées par les caractéristiques spatiales (comme nous l'avons discuté à la fin de la section 6.3.3), nous devons être en mesure de calculer des valeurs initiales de QoS pour chaque nouvel utilisateur et/ou service en fonction de ses attributs en termes de données de localisation via l'application de toutes les étapes de l'algorithme 7 que nous avons établi.

La Figure 6.8 présente un exemple illustratif relatif à notre proposition pour faire face au problème du démarrage à froid.

En effet, l'algorithme consiste en ensemble de tests imbriqués afin de déterminer le vecteur initial en termes de valeurs de QoS, $\mathcal{V}(s) = \{qos_1, \cdots, qos_m\}$, d'un nouveau service s. Nous testons, en premier lieu, si le fournisseur du nouveau service s se trouve dans notre data set (ligne 1). Si c'est le cas, nous calculons le taux de représentativité $\alpha_{c,s}$ du fournisseur pour chaque partition $c \in \{C_1, \cdots, C_{max}\}$, où max est le nombre de partition créées après l'application de l'algorithme 3. Ce taux de représentativité sera combiné avec les vecteurs centroïdes de chaque partition afin de calculer le vecteur $\mathcal{V}(s)$ (ligne 3).

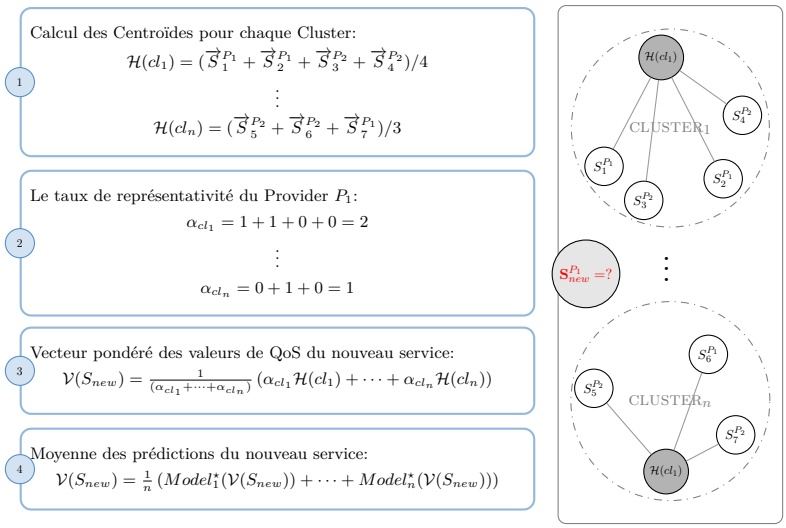


Figure 6.8 – Processus à appliquer pour résoudre le problème du démarrage à froid.

Dans le cas contraire, nous vérifions si l'AS du service se trouve dans notre data set (ligne 4). Dans le cas échéant, nous appliquons le même principe pour calculer le vecteur pondéré $\mathcal{V}(s)$ du nouveau service en se basant sur le taux de représentativité du système autonome (l'$\mathcal{AS}(s)$) du service (ligne 5), par rapport à l'ensemble des partitions, ainsi que les vecteurs centroïdes de ces derniers (ligne 6). Dans le cas contraire, c'est-à-dire, ni le fournisseur $\mathcal{P}(s)$ ni le système autonome $\mathcal{AS}(s)$ ne figurent pas dans notre data set, nous passons à vérifier si le pays du service $\mathcal{C}(s)$ figure dans notre data set; et nous appliquerons le même principe pour calculer le vecteur $\mathcal{V}(s)$ (ligne 9) dans le cas échéant. Pour le cas contraire nous sélectionnons le provider le plus proche en distance géodésique avec le provider du nouveau service (ligne 11). Une fois ce dernier est sélectionné, nous appliquerons le même procédé pour en calculer le vecteur des valeurs de QoS du service s (ligne 12 et ligne 13).

Une fois cette étape de tests terminée, nous utilisons les nouvelles valeurs de QoS (générées lors de la première étape) comme entrée de tous les auto-encodeurs déjà entraînés via l'algorithme 4 où nous calculons la moyenne pondérée des résultats obtenus (ligne 14).

6.5 Conclusion

Les attributs de QoS sont une composante essentielle pour la création d'applications orientées services. Cependant, la plupart des travaux existants sur la prédiction de la QoS rencontrent des problèmes de diverses natures, tels que, la rareté des données et le problème du démarrage à froid. Dans ce chapitre, nous avons abordé les problèmes susmentionnés en adoptant une conception basée sur les modèles d'apprentissage profond. En effet nous avons proposé un modèle basé sur les auto-encodeurs en exploitant de nombreux variants afin d'optimiser les capacités de prédiction. Nous avons également introduit une étape de partitionnement basée sur l'algorithme des cartes auto-organisatrices pour le prétraitement avant la phase d'apprentissage du modèle. Cette étape était utilisée pour réduire l'effet de l'éparpillement de la matrice de qualité de service et à améliorer la précision de la prédiction.

Dans le chapitre suivant, nous nous intéressons au paramétrage de nos modèles tout en comparant leurs performances avec celles des méthodes issues de l'état-de-l'art.

Algorithme 7 : Traitement du problème du démarrage à froid

Input : s ▷ nouveau service
$C = \{C_1, \cdots, C_{max}\}$ ▷ liste des clusters de services
$\mathcal{H}(C) = \{\mathcal{H}(C_1), \cdots, \mathcal{H}(C_{max})\}$ ▷ Les centroïdes de clusters
Θ ▷ Tableau des meilleurs modèles

Data : $\mathcal{P}$, $\mathcal{AS}$, $\mathcal{C}$ ▷ Liste des Providers, des ASNs et des pays disponibles dans notre data set

if $(\mathcal{P}(s) \in \mathcal{P})$ **then**

$$\alpha_{c,s} = \frac{1}{|c|} \sum_{j=1}^{|c|} \begin{cases} 1, & \text{if } \mathcal{P}(s) = \mathcal{P}(s_j) \\ 0, & \text{sinon} \end{cases}$$

$$\mathcal{V}(s) = \frac{1}{\sum\limits_{c \in C} \alpha_{c,s}} \sum_{c \in C} \alpha_{c,s} \mathcal{H}(c)$$

else if $(\mathcal{AS}(s) \in \mathcal{AS})$ **then**

$$\alpha_{c,s} = \frac{1}{|c|} \sum_{j=1}^{|c|} \begin{cases} 1, & \text{if } \mathcal{A}(s) = \mathcal{A}(s_j) \\ 0, & \text{sinon} \end{cases}$$

$$\mathcal{V}(s) = \frac{1}{\sum\limits_{c \in C} \alpha_{c,s}} \sum_{c \in C} \alpha_{c,s} \mathcal{H}(c)$$

else if $(\mathcal{C}(s) \in \mathcal{C})$ **then**

$$\alpha_{c,s} = \frac{1}{|c|} \sum_{j=1}^{|c|} \begin{cases} 1, & \text{if } \mathcal{C}(s) = \mathcal{C}(s_j) \\ 0, & \text{sinon} \end{cases}$$

$$\mathcal{V}(s) = \frac{1}{\sum\limits_{c \in C} \alpha_{c,s}} \sum_{c \in C} \alpha_{c,s} \mathcal{H}(c)$$

else

$\mathcal{P}(s) = \mathcal{P}(\underset{\forall s_i \in S}{\operatorname{argmin}}(\mathcal{D}(s, s_i)))$ ▷ Récupérer le fournisseur de service le plus proche

$$\alpha_{c,s} = \frac{1}{|c|} \sum_{j=1}^{|c|} \begin{cases} 1, & \text{if } \mathcal{P}(s) = \mathcal{P}(s_j) \\ 0, & \text{sinon} \end{cases}$$

$$\mathcal{V}(s) = \frac{1}{\sum\limits_{c \in C} \alpha_{c,s}} \sum_{c \in C} \alpha_{c,s} \mathcal{H}(c)$$

$\mathcal{V}(s) = \frac{1}{|\Theta|} \sum\limits_{Model^\star \in \Theta} Model^\star(\mathcal{V}(s))$ ▷ Prédiction des valeurs de QoS initiales

return $\mathcal{V}(s)$ ▷ service initial vector

Chapitre 7

Expérimentations et évaluation des performances

Sommaire

7.1 Introduction . . . 114
7.2 Jeux de données et métriques d'évaluation . . . 115
7.2.1 Bases de test existantes . . . 115
7.2.2 Base de test utilisée . . . 116
7.2.3 Métriques d'évaluation utilisées . . . 117
7.3 Comparaison des performances . . . 118
7.3.1 Impact des paramètres d'apprentissage . . . 118
7.3.2 Impact des informations de localisation . . . 119
7.3.3 Choix de modèles . . . 122
7.3.4 Résultats des expérimentations & comparaison des performances . . . 125
7.4 Problème du démarrage à froid . . . 128
7.5 Menaces à la validité (Threats To Validity) . . . 129
7.5.1 Validité interne . . . 129
7.5.2 Validité externe . . . 129
7.6 Conclusion . . . 130

7.1 Introduction

Dans les chapitres précédents (chapitre 5 et chapitre 6), nous avons présenté les défis des approches actuelles ainsi que les caractéristiques et les décisions de conception qui ont été prises pour relever ces défis. Dans ce chapitre, l'impact des différentes décisions de conception que nous avons entrepris pour surmonter ces défis est évalué. L'approche d'évaluation est conçue pour répondre aux questions de recherche posées en introduction de cette thèse, et qui se concentrent le paramétrage des différents modèles, le démarrage à froid et la précision et la performance de la prédiction.

Dans ce chapitre, nous présentons la partie expérimentation des différentes contributions étaler dans le chapitre précédent. Ces expériences ont été menées sur la plate-forme MESO@LR [56] de l'Université de Montpellier, France. Pour les réaliser, nous avons utilisé quatre nœuds (avec 14 cœurs) de 128 Go de RAM. Tous les programmes d'apprentissage ont été implémentés en Python Tensorflow [57], où ce dernier représente un outil open source d'apprentissage automatique développé par Google.

Nous commençons le chapitre par la description du jeu de données ainsi que les différentes métriques d'évaluation que nous avons utilisées pour mener nos expériences. Ensuite, nous présentons,

56. https://meso-lr.umontpellier.fr/
57. https://www.tensorflow.org/learn?hl=fr

via des résultats expérimentales, l'impact des différents hyperparamètres, des modèles proposés, sur les performances de la prédiction. Enfin, nous montrons les différentes expérimentations menées tout en discutant les résultats obtenus. Ce chapitre est conclu par une discutions sur les différentes menaces possibles à la validité.

7.2 Jeux de données et métriques d'évaluation

7.2.1 Bases de test existantes

Comme nous l'avons présenté dans le chapitre précédent, diverses approches ont été proposées et développées, d'une manière exhaustive, afin de traiter la problématique de la prédiction de QoS des services Web. Pour comparer leur performance de prédiction, la plupart de ces approches utilisent un ensemble de données QoS de services Web réels à grande échelle. Certaines approches, comme celles définies dans (Ma et al., 2016; Qi et al., 2017; Shrivastava and Sharma, 2021), utilisent simplement un ensemble de données de classement des films, MovieLens (Herlocker et al., 2004) [58], pour les études expérimentales, ce qui n'est pas suffisamment convaincant.

Les auteurs, dans (Zheng and Lyu, 2010), ont invoqué un total de 100 services Web en utilisant une multitude d'ordinateurs. Ils ont utilisé exactement 150 nœuds répartis dans le monde entier. Cet ensemble de données est composé de 150 fichiers (un par utilisateur), où chaque enregistrement comprend 10 000 invocations de services des 100 services Web disponibles. Au total, il y a plus de 1,5 million invocations de services Web. Le jeu de données est connu sous le nom de WS-Dream1 [59]

Dans (Zheng et al., 2010), les auteurs ont également effectué des évaluations sur la QoS réelle observée par les utilisateurs sur un ensemble de 5 825 services Web à partir d'emplacements répartis. Dans ce jeu de données, que nous le nommons WS-Dream2, les auteurs se sont intéressés à deux QoS différentes, à savoir, le temps de réponse et le débit. Ces deux métriques sont évaluées par 339 utilisateurs de services distribués.

Les mêmes auteurs ont présenté dans (Zhang et al., 2011b) un jeu de données, WS-Dream3, comprenant des valeurs de QoS en termes de temps de réponse et de débit pour un ensemble de 4 532 services Web invoqués par 142 utilisateurs de service à 64 intervalles de temps. Cet ensemble de données est utilisé dans des expériences conçues pour évaluer les performances des prédictions des différentes approches sensibles au temps. Le tableau 7.1 résume ces ensembles de données disponibles sur la qualité de service des services Web.

Tableau 7.1 – Jeux données de QoS existants

Jeu de données	abréviation	Nbr. d'utilisateurs	Nbr. de services	Nbr. d'invocations
GroupLens	GP	6 040	3 952	1 000 209
WS-Dream1	WSD_1	150	100	1 542 884
WS-Dream2	WSD_2	339	5 825	1 974 675
WS-Dream3	WSD_3	142	4 532	30 287 611

Plusieurs approches utilisent d'autres ensembles de données. Par exemple, dans (Zhang et al., 2021a), le jeu de données utilisé dans les différentes expériences, nommé EdgeQoS, est un jeu de données obtenu par la fusion de deux jeux de données: WS-Dream-3 et Telecom dataset (Xu et al., 2019). Nous avons aussi le jeu de données QWS pour *Quality of Web Service*, qui ne concernait,

58. MovieLens, mis en place par GroupLens, contient un total de 1 000 209 évaluations de films par les utilisateurs sur un ensemble de 6 040 utilisateurs et 3 952 films. http://grouplens.org/datasets/movielens/1m/

59. WS-Dream: http://www.wsdream.net. À noter que le chiffre 1 a été ajouté pour différencier les différentes versions de cet ensemble de données.

malheureusement, qu'un seul utilisateur sur 2 507 services Web; ce qui limite l'applicabilité de cet ensemble de données.

7.2.2 Base de test utilisée

Afin d'évaluer notre approche, nous avons mené un ensemble d'expériences sur un référentiel de QoS pour les services Web, à grande échelle, nommé WS-Dream (Zheng et al., 2014). Ce jeu de données est composé par deux ensembles de données: WS-Dream2 et WS-Dream3. À noter que les invocations, sur ces deux jeux de données, considèrent deux critères de QoS pour les services Web ($c \in \{rt, tp\}$). La première invocation se porte sur le critère de temps de réponse rt. Tandis que le deuxième critère représente le débit du service Web tp. Le tableau 7.2, résume certaines caractéristiques essentielles pour ces deux jeux de donnée WS-Dream, telles que, la moyenne globale de valeurs de QoS, l'intervalle d'invocation, le pourcentage de valeurs manquantes, etc., et cela pour les deux critères de QoS.

Tableau 7.2 – Détails d'information sur le jeu de données WS-Dream (RT : temps de réponse, TP: débit)

Statistiques	WS-Dream1		WS-Dream2	
	RT	TP	RT	TP
Échelle de données	0-20 s	0-20 s	0-20 s	0-20 b/s
Moyenne des valeurs	0.908 s	47.561 s	3.165 s	9.608 b/s
Nbr. d'utilisateurs	339	339	142	142
Nbr. de services	5 826	5 826	4 500	4 500
Intervalles d'invocation	1	1	64	64
Nbr. total d'invocations	1 874 177	1 831 592	30 286 687	30 286 687
Nbr. total de valeurs manquantes	100 837	143 422	10 609 313	10 609 313
% de valeurs manquantes	5.10%	7.26%	25.94%	25.94%

Pour le bon déroulement de nos expériences, nous avons apporté quelques modifications, d'ordre empirique, sur ces deux jeux de données:

— *WS-Dream2*: L'objectif derrières ces mises à jour est de réduire le nombre de valeurs manquantes sur les différents attributs de services Web, en termes d'informations de géolocalisation, tels que, le numéro AS [60], l'adresse IP, la latitude et la longitude du service, etc. Pour ce faire, nous utilisons les informations de géolocalisation issues des bases de données GeoIP [61] et IP2Location [62].

— *WS-Dream3*: Nous avons remarqué que le jeu de données, concernant le deuxième data set, contient à peu près 26% d'invocations non réalisées. Ce qui représente un taux assez élevé de données manquantes. Ce problème, spécifique au jeu de données actuel, peut nous poser certains problèmes lors de la phase des expérimentations. En effet, ce niveau de *sparsity* augmente le risque d'avoir un clustering invalide lors de la phase de partitionnement de données. En d'autres termes, cette phase peut nous engendrer un cluster regroupant que les services Web non invoqués, ce qui poserai un problème de division par zéro lors des différents calculs. Afin de réduire, au maximum, ce risque; nous appliquons les deux règles suivantes sur les matrices d'invocations $M_{c,t} = \{QoS_{u,s}\}_{m \times n}$; où $c \in \{rt, tp\}$ sont les deux critères de QoS, $t \in \{1, 2, \ldots, 64\}$ représente le moment d'invocation (une invocation à chaque 15 minutes), m est le nombre d'utilisateurs et n est le nombre de services:

60. Un AS (ou *Asynchronous system*) est soit un réseau unique, soit un groupe de réseaux contrôlés par un administrateur de réseau commun (ou un groupe d'administrateurs) pour le compte d'une entité administrative unique (université, entreprise commerciale, etc.).

61. Maxmind GeoIP2 Geolocational Databases. Consulter et télécharger en 2019 à partir de http://dev.maxmind.com/geoip/geoip2/geolite2/

62. IP2Location LITE Databases. Consulter et télécharger en 2019 à partir de http://lite.ip2location.com

1. Pour chaque valeur $QoS_{s,u}^{c,t}$ de la matrice d'invocation $M_{c,t}$ actuelle; si cette valeur est manquante, alors elle sera remplacée par la moyenne des valeurs valides $QoS_{s,u}^{c,t'}$ déjà invoquées, et qui sont mentionnées dans les matrices précédentes à l'intervalle de temps t actuel: $QoS_{s,u}^{c,t} = \frac{1}{t-1}\sum_{t'=1}^{t-1} QoS_{s,u}^{c,t'}$
2. Pour chaque valeur $QoS_{s,u}^{c,t}$ invalide existante dans la matrice d'invocation $M_{c,t}$ actuelle, si nous ne nous sommes pas en mesure de récupérer les invocations précédentes, alors cette valeur sera remplacée par la valeur 0 ($QoS_{s,u}^{c,t} = 0$).

Tableau 7.3 – Les différentes améliorations enregistrées sur le jeu de données

Attributs	% de valeurs manquantes (Avant)	% de valeurs manquantes (Après)
QoS values	26%	23%
AS number	21%	15%
IP Address	24%	13%
Latitude/Longitude	22%	15%

Le tableau 7.3 illustre les différentes améliorations enregistrées sur l'ensemble du jeu de données initial. Ces améliorations auront un impact important pour le déroulement correct des différentes expériences, ainsi que sur les performances en termes de prédiction de valeurs de QoS manquantes, par rapport à notre approche proposée.

7.2.3 Métriques d'évaluation utilisées

Avant de détailler les performances de prédiction de notre proposition, nous présentons, dans cette partie, les différentes métriques d'évaluation que nous nous utilisons afin de valider les performances de nos contributions. Pour évaluer ces performances, nous exploitons deux métriques fréquemment utilisées dans le domaine du filtrage collaboratif (Section 3.5). En effet, nous calculons les écarts relatifs moyens absolus (MAE Mean Absolute Error) et quadratiques (RMSE pour Root Mean Square Error), données, respectivement, par les équations 3.5.1 et 3.5.2. Ces dernières seront adaptées au contexte des services Web en remplaçant la variable N par $|V|$, ce qui représente le data set utilisé pour la phase de validation, et s pour désigner un service au lieu d'un item i.

Étant donné que nous avons utilisé le principe d'une validation croisée, les évaluations des performances s'effectueront également sur la moyenne MAE et RMSE. La moyenne MAE_{AVG} des écarts relatifs moyens absolus est calculer comme suit:

$$MAE_{AVG} = \frac{1}{\left|\bigcup_{allV_{k\text{-}fold}}\right|} \sum_{allV_{k\text{-}fold}} (|V_{k\text{-}fold}| \times MAE_{V_{k\text{-}fold}}) \tag{7.2.1}$$

Tandis que celle de la moyenne $RMSE_{AVG}$, qui est également désigné par mve dans l'étape 11 de l'Algorithme 4, correspond aux écarts relatifs moyens quadratiques est calculée en appliquant l'équation suivante:

$$RMSE_{AVG} = \frac{1}{\left|\bigcup_{allV_{k\text{-}fold}}\right|} \sum_{allV_{k\text{-}fold}} (|V_{k\text{-}fold}| \times RMSE_{V_{k\text{-}fold}}) \tag{7.2.2}$$

où $V_{k\text{-}fold}$ représente le data set de validation set.

7.3 Comparaison des performances

7.3.1 Impact des paramètres d'apprentissage

Le modèle des auto-encodeurs regorge d'hyperparamètres, ce qui rend la tâche du choix de ces derniers relativement complexe. Cela nécessite une très grande compétence avec un certain niveau d'expertise, ainsi que de nombreux essais et erreurs pour arriver à sélectionner des valeurs optimales.

Dans cette perspective, nous considérons que le réglage des hyperparamètres est un élément important et qui a un impact énorme sur les performances de notre approche. De ce fait, nous nous sommes concentrés sur deux hyperparamètres que nous avons jugés importants: la taille de chaque couche pour notre auto-codeur profond et le choix de la fonction d'activation à appliquer sur ces couches. Dans ce qui suit, nous présentons une série d'expériences comparatives effectuée afin d'obtenir les paramètres optimaux relatives à notre modèle.

Impact relatif au nombre de couches

La taille de la couche commune (couche de facteurs latents) est un hyperparamètre que nous devons définir bien avant la phase d'apprentissage de notre auto-encodeur profond. Comme tous les réseaux de neurones à propagation avant, il a été prouvé que les auto-encodeurs profonds donnent une bien meilleure compression que les auto-codeurs superficiels ou linéaires correspondants (Hinton and Salakhutdinov, 2006). Ainsi, des couches supplémentaires peuvent apprendre des représentations complexes en approximant un nombre important de correspondances entre l'entrée et le code. Pour cette raison, et en dehors de la couche des facteurs latents, nous utilisons un réseau de neurones à trois couches pour les phases d'encodage et de décodage. Ce choix est basé sur la puissance du réseau neuronal à trois couches, démontrée expérimentalement dans (Hecht-Nielsen, 1992). De ce fait, l'architecture finale de notre auto-encodeur que nous avons adoptée est composée de quatre couches entièrement connectées (trois couches plus la couche des facteurs latents) pour les réseaux d'encodage et de décodage.

Afin de déterminer le nombre optimal de neurones pour chaque couche, de nombreuses règles empiriques ont été prises en compte (Panchal et al., 2011). En effet, et selon ces recommandations de bonnes pratiques, la taille de la couche des facteurs latents ainsi que celle des couches cachées doit être: (A) comprise entre la taille de la couche d'entrée et celle de la couche de sortie, (B) correspondre à deux tiers de la taille de la couche d'entrée, plus celle de la couche de sortie, et enfin, (C) correspondre à moins de deux fois la taille de la couche d'entrée. Ces trois recommandations ont été considérées comme un point de départ afin de déterminer la limite supérieure de la taille de la couche des facteurs latents.

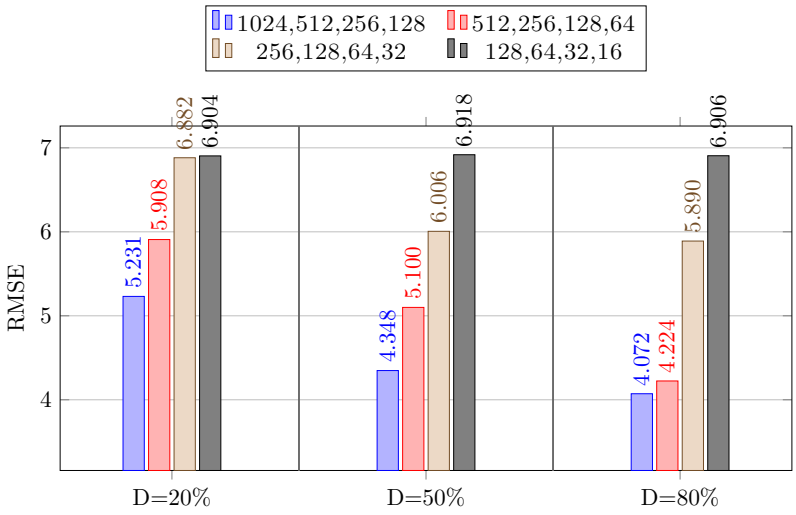


Figure 7.1 – Influence du choix de la taille des couches. Apprentissage par mini-batch avec un taux d'apprentissage égal à 0.05 et un total de vingt époques (ou cycle) d'apprentissage.

La figure 7.1 illustre les différents résultats pour quatre valeurs différentes que nous avons choisies, et qui agissent sur les tailles des couches de l'auto-codeur. Comme nous pouvons le constaté, la configuration qui donne de meilleurs résultats par rapport à notre modèle à 4 couches est 1024 neurones pour la couche d'entrée, suivie de deux couches cachées à, respectivement, 512 et 256 neurones et en fin une couche à 128 neurones. Notez que cette configuration dépend de la dimension de notre entrée (avec 4500 services d'entrée) pour le modèle d'auto-encodeur. Pour l'auto-encodeur clustérisé, cette configuration est mise à jour par la règle de trois sur la taille d'entrée (qui correspond à la taille exacte du cluster). À noter que les expériences, afin de déterminer les valeurs optimales de la taille des différentes couches, sont effectuées sur des mini-batchs à densités variables (de 20% à 80%, en passant par 50%).

Impact relatif au choix de la fonction d'activation

Le deuxième hyperparamètre important que nous devons le déterminer afin qu'il soit utilisé dans la phase d'apprentissage de notre auto-encodeur profond est la fonction d'activation. Pour inspecter l'impact de cet hyperparamètre, nous utilisons certains des choix les plus populaires existants dans le domaine d'apprentissage profond. Ainsi, nous examinons expérimentalement l'utilisation la fonction sigmoïde comme unité d'activation standard, l'unité exponentielle linéaire (ELU) (Clevert et al., 2016), l'unité de rectification linéaire (ReLU) (Sun et al., 2015) et enfin, la fonction (Leaky ReLU) (Xu et al., 2015) qui représente une amélioration de la fonction d'activation précédente.

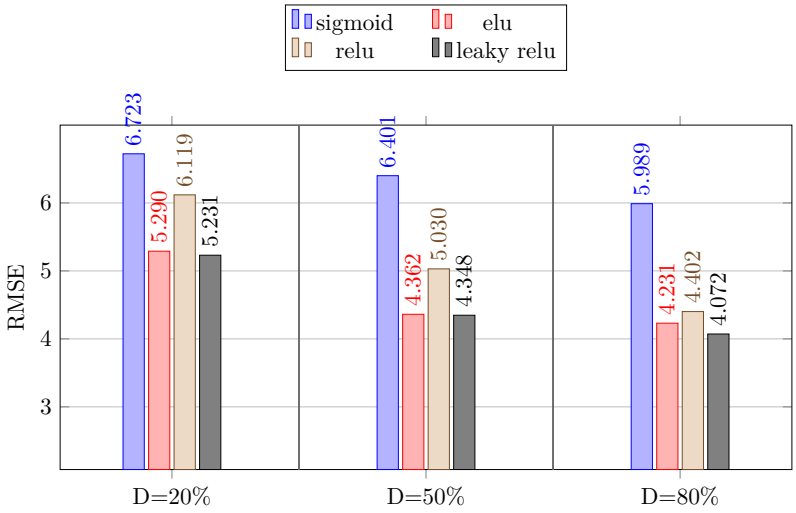


Figure 7.2 – Influence du choix de la fonction d'activation. Apprentissage par mini-batch avec un taux d'apprentissage égal à 0.05 et un total de vingt époques (ou cycle) d'apprentissage.

L'effet des différentes fonctions d'activation sur la métrique d'entraînement RMSE est illustré par la figure 7.2. En se basant sur cette constatation empirique, et afin d'accélérer la vitesse de convergence dans la phase d'apprentissage, nous optons pour une fonction d'activation non saturée, à savoir, Leaky ReLU.

7.3.2 Impact des informations de localisation

Nous nous intéressons dans cette sous-section à l'utilisation des cartes auto-adaptatives en démontrant leurs intérêts par rapport à notre approche. En effet, ces réseaux, connues aussi sous le nom de cartes auto-organisatrices, représentent une catégorie de réseaux de neurones artificiels fondée sur des méthodes d'apprentissage non-supervisées. L'utilisation de ce type de réseau a un double avantage pour notre démarche. D'un côté, il nous permet de faire face aux problèmes liés à la rareté de données, en effectuant la prédiction sur des jeux de données réduits avec un nombre restreint de valeurs manquantes; ce qui aura un impact positif sur nos performances de prédiction. De l'autre côté, il nous permet aussi à traiter efficacement le problème du démarrage à froid.

Comme nous l'avons déjà noté précédemment, les résultats finaux en termes de métriques d'évaluation, représentent la moyenne générale sur les métriques d'évaluation relatives à chaque cluster (voir figure 7.7). En effet, l'application d'une « quantification vectorielle » de l'espace de données, via l'utilisation d'un réseau de Kohonen nous a permet de créer un nombre total de huit clusters. Pour chacun d'eux, nous avons associé un auto-encodeurs profond, où chacun d'eux est construit en fonction de sa taille vectorielle d'entrée (nombre de services). L'histogramme présenté par la figure 7.3 montre le détail sur les résultats d'apprentissage de chaque auto-encodeur pour chacun de ces clusters. En fait, ce graphique représente les résultats en termes de métriques RMSE et MAE de huit auto-encodeurs que nous avons créés. Les valeurs de la moyenne pondérée des métriques RMSE et MAE sont tracées avec des lignes bleues et rouges respectivement. À noter que l'axe des abscisses correspond à la taille des données d'entrée de ces auto-encodeurs.

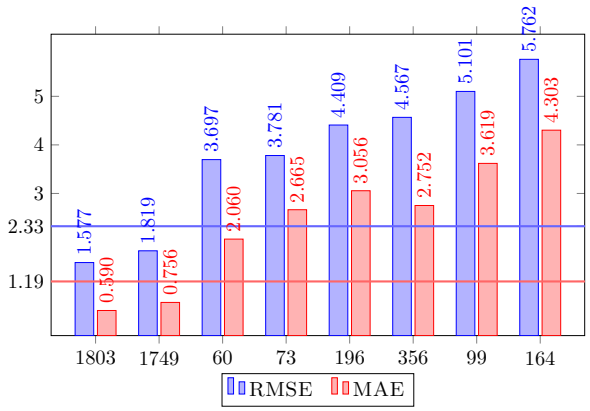


Figure 7.3 – Performances des auto-encodeurs profonds sur les différentes partitions (l'initialisation de la carte auto-organisatrice est basée sur les attributs de localisation)

Comme nous pouvons le constater, les meilleurs résultats, en termes de performances de prédiction, appartiennent aux clusters ayant le plus grand nombre de services (partition à 1803 et 1749 services). Leurs résultats sont bien inférieurs au résultat moyen (lignes bleu et rouge sur l'histogramme). Cependant, la troisième et quatrième positions reviennent aux clusters avec le plus petit nombre de services (60 et 73 respectivement). Pour les clusters restants, et malgré le fait qu'ils contiennent un nombre considérable de services, nous remarquons que leurs résultats sont moins performants par rapport à la moyenne pondérée de tous les résultats.

L'interprétation que nous pouvons donner pour ces résultats est fortement liée avec la technique d'initialisation de notre réseau de carte autoadaptative que nous avons adoptée. En effet, nous avons opté pour un réseau à 80×80 neurones pour effectuer la phase de partitionnement des différents services. Chaque ligne dans cette carte correspond à un pays de notre ensemble de données. Elle contient exactement 80 neurones [63]. La taille de chaque vecteur pour chaque neurone est égale au nombre d'utilisateurs disponibles dans notre ensemble de données.

Ainsi, chaque ligne correspond à un pays sélectionné ($country_i$) sur la carte. Elle représente la distance géodésique maximale (selon leurs valeurs de longitude/latitude) entre toutes les paires de services qui composent ce $country_i$. C'est-à-dire, nous avons essayé de récupérer les arguments, en termes de pays, de la fonction objectif suivante: $argmax(\mathcal{D}(s_1, s_2))$, $\forall s_1, s_2 \in Country_i$; où $\mathcal{D}$ désigne une fonction qui nous retourne la distance géodésique qu'existe entre deux zones géographies. Cela signifie que le premier neurone est initialisé avec le vecteur de s_1 divisé par la taille de tous les services qui composent le pays sélectionné. La même chose est opérée pour le dernier neurone avec le vecteur s_2. Les autres services sont distribués sur le reste des neurones en fonction de leurs distances géodésiques. À noter que, s'il y a plusieurs services avec les mêmes coordonnées géographiques, le neurone correspondant est initialisé avec le vecteur moyen de ces services. Enfin, les services qui ne disposent pas d'informations de localisation (c'est-à-dire latitude/longitude) sont répartis de manière

63. À noter que, dans le jeu de données WS-Dream, nous avons détecté 81 pays différents. Ce qui nous a contraint à fusionner deux pays dans la même ligne

aléatoire sur l'ensemble des neurones de leur ligne de pays correspondante.

À fin d'appuyer d'avantage l'intérêt de notre processus d'initialisation que nous avons adopté, nous avons appliqué le processus de partitionnement mais avec une initialisation aléatoire pour l'algorithme de carte autoadaptative. Comme nous pouvons le constater sur la figure 7.4, les résultats moyens en termes de RMSE et de MAE sont largement inférieurs aux résultats d'un partitionnement où l'initialisation s'était basée sur les caractéristiques spatiales des services. Nous remarquons, d'après les histogrammes de la figure 7.3 et la figure 7.4, que les performances, pour un partitionnement avec une initialisation basée sur le facteur de localisation des services Web, sont à 72% meilleures pour la métrique RMSE et de presque 67% pour la valeur MAE, par rapport à un partitionnement avec une inutilisation aléatoire.

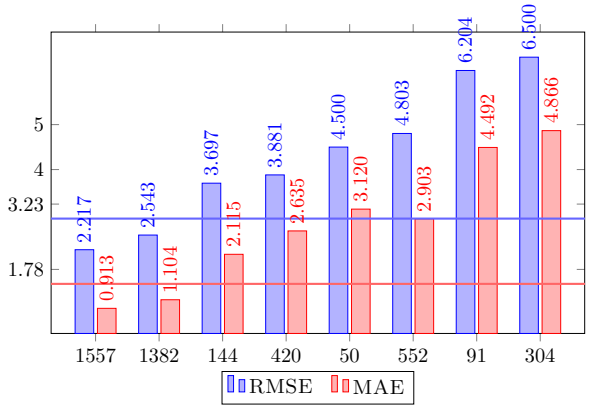


Figure 7.4 – Performances des auto-encodeurs profonds sur les différentes partitions (l'initialisation de la carte auto-organisatrice est basée sur un processus aléatoire)

Nombre de remarques et d'observations intéressantes, par rapport au choix de la technique d'initialisation de notre carte autoadaptative, ont été enregistrées. Nous les résumons comme suit:

— Nous observons, par rapport aux deux métriques RMSE et MAE, qu'environ 80% des services sont regroupés dans les trois premiers clusters pour un partitionnement avec une initialisation basée sur les caractéristiques de localisation (figure 7.3). Tandis que seulement 69% de l'ensemble des services pour une initialisation aléatoire (figure 7.4);

— Nous remarquons, aussi, qu'il existe une relation directe entre les résultats et la quantité de données d'entrée disponibles pour les différents clusters. Plus la densité des données d'entrée est élevée, meilleurs sont les résultats. La deuxième colonne du tableau 7.4 illustre cette relation proportionnelle;

Tableau 7.4 – Relation entre la taille des partitions et le % de parcimonie de données.

Taille du Cluster	% de sparsité	% de données manquantes [64]
1803	22,34%	13,81%
1749	23,10%	9,43%
60	19,30%	20,00%
73	24,44%	13,70%
196	28,06%	32,14%
356	26,63%	33,99%
99	22,53%	15,15%
164	26,69%	24,39%

64. En termes d'information de géolocalisation

— Nous constatons que les clusters avec des résultats moins performants sont composés par des services où leurs informations de localisation sont manquantes. En d'autres termes, la plupart des services avec des caractéristiques spatiales non disponibles sont regroupés dans les dernières clusters (en termes de performance des résultats). La troisième colonne du tableau 7.4 indique le taux d'information de localisation manquantes par rapport à la taille globale du cluster;

— Enfin, avec notre mécanisme d'initialisation adopté, nous avons assuré un comportement déterministe pour la phase du partitionnement.

7.3.3 Choix de modèles

Il a été prouvé expérimentalement en apprentissage profond, que le modèle classique des auto-encodeurs dégénère souvent en se déclinant en un réseau d'identité et qu'il ne parvient pas à apprendre les relations latentes qui existent entre les données (Glorot and Bengio, 2010). Pour réduire ce risque, nous utilisons une variante du modèle d'auto-encodeur profond en « bruitant » les entrées et en entraînant le modèle à « débruiter » les sorties finales. Pour améliorer la robustesse du modèle, nous appliquons un bruit de masquage (ou *masking noise*), qui est une technique de corruption de données, en définissant une fraction aléatoire de l'ensemble des données d'entrée à la valeur zéro[65]. Pour ce faire, nous explorons quatre taux de bruitage différents: 0% (c'est-à-dire sans bruit), 20%, 50% et 80%, appliqués aléatoirement, à l'ensemble des données. Notez que le même processus est appliqué pour chaque auto-encodeur différents, cela après application de la phase de partitionnement.

L'auto-encodeur profond – DAE

Afin d'extraire les facteurs latents de l'ensemble de données, nous utilisons un réseau de neurones d'encodage à quatre couches avec la configuration, en termes du nombre de couches ainsi que le nombre de neurones pour chacune de ces couches, que nous avons validée précédemment (voir 7.3.1), à savoir, 1024 pour la couche d'entrée, 512 et 256 pour les deux couches cachées intermédiaires et 128 neurones pour la couche des facteurs latents. A noter qu'une configuration symétrique similaire est utilisée pour le réseau de neurones de décodage pour reconstruire les données d'entrée à partir de la couche des facteurs latents (c'est-à-dire 128-256-512-1024).

Pour garantir des résultats de prédiction plus robustes et tenir compte de l'erreur de généralisation pour l'ensemble du jeu de données utilisé, nous menons nos expériences en appliquant le principe de la validation croisée à k-plis (ou k-fold cross-validation) (Refaeilzadeh et al., 2016). Cette technique consiste à diviser aléatoirement l'échantillon initial en données d'entraînement et en données test. En effet, les expériences sont réalisées à l'aide de cinq-plis (partition) avec des valeurs de densité différentes. Chaque partition représente un pourcentage donné (% de densité) de l'ensemble de données disponibles.

La densité représente le rapport entre les ensembles de données d'apprentissage à celles de la validation. A cette fin, nous considérons trois possibilités pour ce ratio de densité, à savoir, 20%, 50% et 80%. Enfin, les résultats de validation représentent la moyenne des résultats sur les cinq-plis, et qui sont calculées via l'application de l'équation 7.2.1 et l'équation 7.2.2.

La figure 7.5 illustre les performances de prédiction en termes des deux métriques que nous avons appliquées, à savoir, la RMSE et MAE. Ces performances représentent la moyenne des différents performances relatives aux cinq exécutions opérées sur trois densités de données différentes. Pour chaque valeur de densité donnée, nous appliquons quatre variations de bruit sur les données d'entrée. Notez que la précision de la prédiction de la qualité de service augmente continuellement lorsque la densité de la matrice augmente (en raison de l'augmentation du taux de bruit).

Cependant, dans la figure 7.6, nous constatons que l'apprentissage a convergé pour les différentes configurations après moins de 100 itérations. En outre, nous pouvons facilement observer qu'il y a

65. Il est important de mentionner que d'autres types de corruption peuvent être appliqués sur les données d'entrée, tels que, le *bruit gaussien* où un bruit gaussien isotrope est ajouté à un sous-ensemble de données, et *le bruit sel et poivre*, où une fraction de données est aléatoirement fixée à sa valeur minimale ou maximale possible. (Arora et al., 2012)

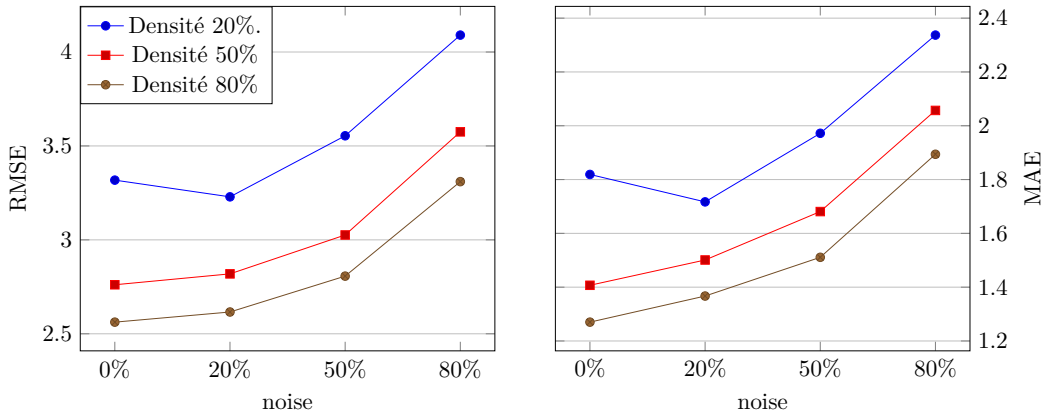


Figure 7.5 – Performance de la prédiction pour l'auto-encodeur profond en termes de RMSE et MAE (Variations moyennes des métriques RMSE/MAE)

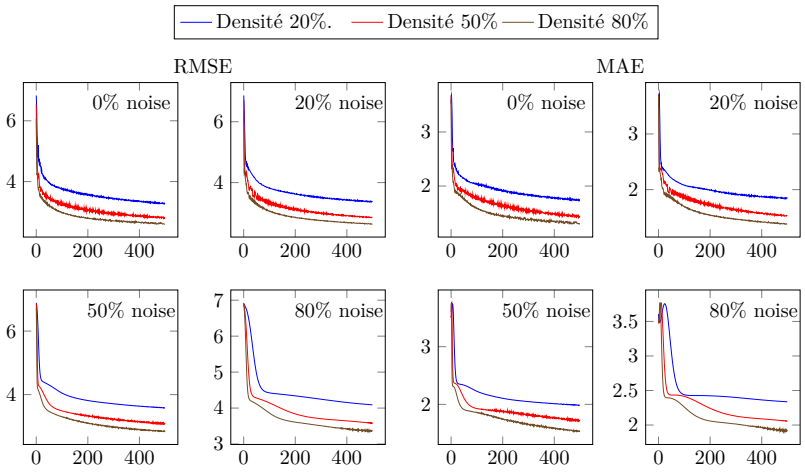


Figure 7.6 – Performances lors de la phase d'apprentissage pour l'auto-encodeur profond en termes de convergence des résultats d'apprentissage

une stabilité (pas d'augmentation après la convergence), et pas de dépassement (pas d'augmentation avant la convergence).

L'auto-encodeur profond partitionné – CDAE

Afin de garantir une meilleure précision dans la prédiction des valeurs de QoS, nous utilisons un auto-encodeur spécifique, et qui est relatif à chaque partition de services crée. En effet, les différents services de notre base de test sont partitionnés sur huit classes (clusters) homogènes par rapport à leurs caractéristiques de localisation.

Comme mentionné précédemment, nous utilisons la même architecture de notre auto-encodeur profond, avec la même configuration d'hyperparamètres (quatre couches pour l'encodage et le décodage, et la même fonction d'activation Leaky ReLU). À noter que la principale différence est que la taille des quatre couches qui composent chaque auto-encodeur dépend de la taille des entrées (nombre de services dans le cluster). De ce fait les tailles de chacune de ces couches sont mises à jour par la règle de trois sur la configuration initiale (1024-512-256-128) en fonction de la taille du cluster (nombre de services en entrée).

Pour ce faire, nous réalisons huit apprentissages différents pour chaque cluster. Nous appliquons le même processus précédent, appliqué sur l'auto-encodeur profond, et cela pour chaque partition selon diverses densités en termes de données d'entrée (20%, 50% et 80%). Pour chaque cas de densité, nous appliquons aussi, un ensemble de variations de bruit, en définissant plusieurs fractions aléatoires de l'ensemble des données d'entrée à la valeur zéro. Ces fractions sont de 0%, 20%, 50% et 80%. Par la

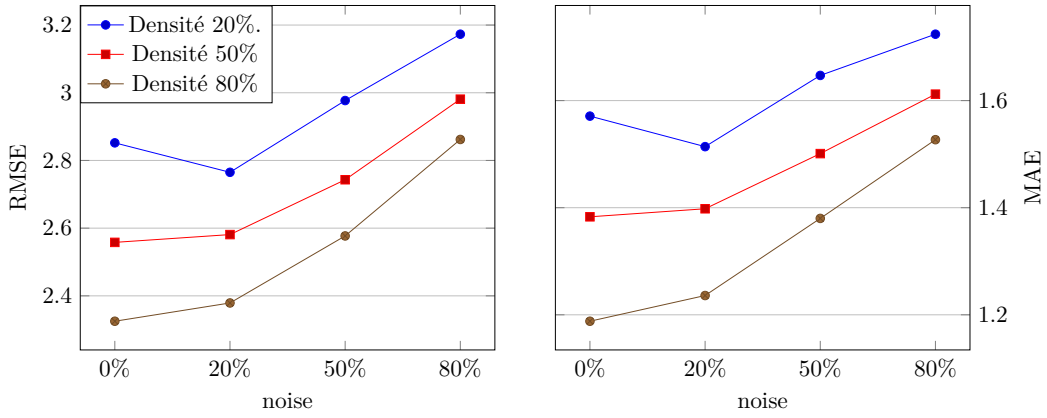


Figure 7.7 – Moyenne des performance de la prédiction pour les auto-encodeurs profonds en termes de RMSE et MAE (Variations moyennes des métriques RMSE/MAE)

suite, nous calculons la moyenne générale des différentes métriques que nous avons utilisées.

Comme le montre les résultats obtenues, illustrées par la figure 7.7, les valeurs moyennes des deux métriques RMSE et MAE, pour les différentes densités, augmentent continuellement lorsque le taux de données d'entrée corrompues augmente (sauf pour le cas où la densité vaut 20%).

Dans la Figure 7.8, nous constatons que la phase d'apprentissage converge un peu plus lentement que celui de l'auto-encodeur profond. À noter que cette convergence représente la moyenne de convergence pour chaque partition, où certains clusters convergent rapidement, tandis que d'autres convergent moins.

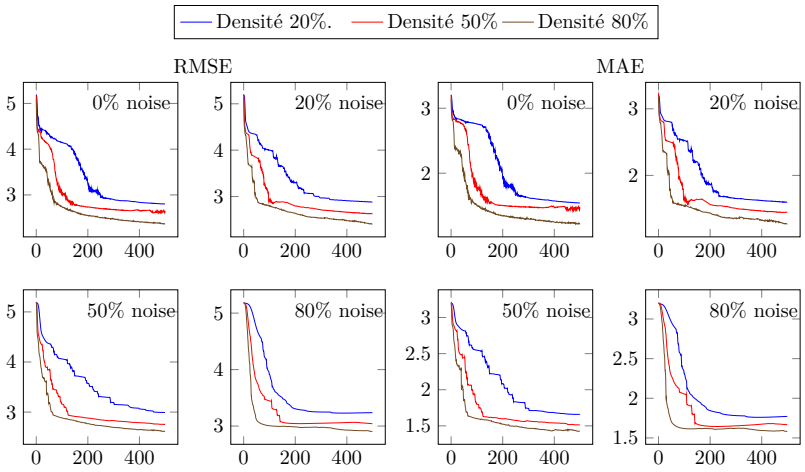


Figure 7.8 – Moyenne des performances lors de la phase d'apprentissage des auto-encodeurs profonds partitionnés en termes de convergence des résultats d'apprentissage

L'Adversarial auto-encodeur – ADAE

Dans cette partie, nous proposons une autre technique pour régulariser notre auto-encodeur. Nous combinons l'utilisation des auto-encodeurs avec les réseaux antagonistes génératifs (GAN) récemment proposés afin de faire correspondre le vecteur des facteurs latents de l'auto-encodeur avec la partie génératrice d'un réseau GAN. Dans ce cas, la partie décodage de notre auto-encodeur jouera le rôle d'un générateur pour le réseau antagoniste génératif.

Pour la mise en place de notre réseau GAN, nous apportons quelques modifications sur la structure originale du réseau proposée par (Goodfellow et al., 2016). En effet, afin de faire correspondre notre structure a celle de l'auto-encodeur profond, le générateur et le discriminateur du GAN sont tous deux constitués de quatre couches. Pour une meilleure différenciation entre les vecteurs générés et les vecteurs réels, nous appliquons, dans la phase d'apprentissage, une fonction de perte euclidienne, qui

mesure la distance spatiale du vecteur généré par rapport aux ensembles de données d'entraînement. Cette dernière est combinée avec RMSprop qui est une technique d'optimisation basée sur le gradient souvent utilisée dans les réseaux GAN (Mao et al., 2017).

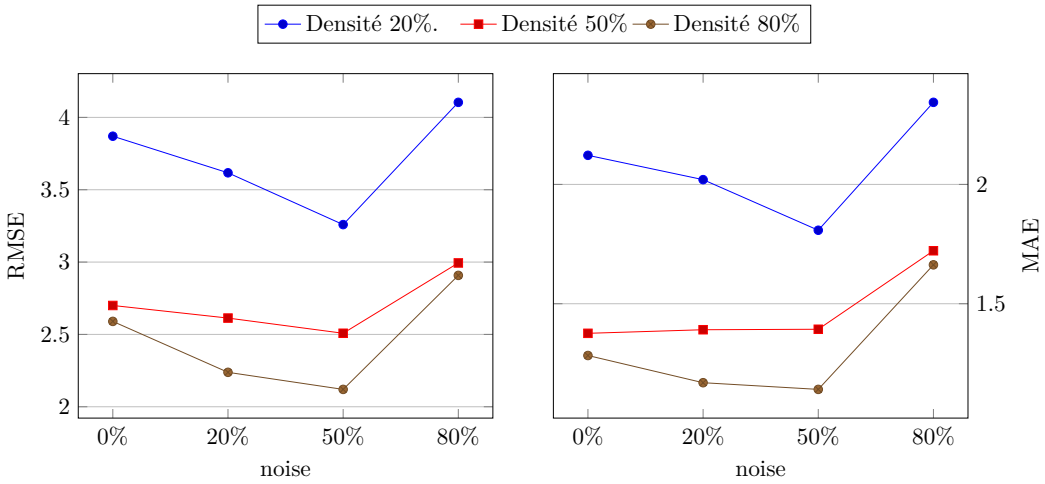


Figure 7.9 – Performance de la prédiction pour l'adversarial auto-encodeur en termes de RMSE et MAE (Variations moyennes des métriques RMSE/MAE)

Pour cette partie, nous appliquons nos différentes expériences via le principe de la validation croisée à k-plis; où nous divisons aléatoirement note échantillon initial en données d'entraînement et en données test. En effet, les mêmes ratios de densités seront appliqués, à savoir 20%, 50% et 80%. La figure 7.9 illustre les performances de prédiction en termes des deux métriques que nous avons appliquées, à savoir, la RMSE et la MAE. Ces performances représentent la moyenne des différents performances relatives aux cinq exécutions opérées sur trois densités de données différentes. Nous constatons que les résultats en termes de performances de prédiction sont meilleurs par rapport à celles de l'auto-encodeur profond. D'un autre côté, nous remarquons que les performances se sont améliorées après l'application de plusieurs taux de bruits sur les données en entrée de l'adversarial auto-encodeur. En effet, nous remarquons que le sommet des performances est atteint après application d'un taux de bruit aléatoire avoisinant les 50% sur l'ensemble des données d'apprentissage. Cette caractéristique est la garantie que notre modèle n'a pas simplement développé un mappage qui mémorise les données d'entraînement car nos entrées et sorties cibles ne sont plus les mêmes; mais plutôt notre adversarial auto-encodeur apprend un champ vectoriel pour mapper les données d'entrée vers une variété de dimension inférieure.

7.3.4 Résultats des expérimentations & comparaison des performances

Sur la base de ce qui a été présenté précédemment, nous avons utilisé une approche basée sur l'utilisation des réseaux de neurones pour la prédiction de la QoS. Le modèle de base que nous avons adopté était l'utilisation d'un auto-encodeur, nommé DAE, pour la prédiction des valeurs de QoS manquantes. Ce dernier était soigneusement paramétré pour traiter le problème de la parcimonie de données, tout en considérant l'ensemble de la base de test. Une amélioration des résultats des performances de notre approche était possible grâce à l'utilisation des cartes auto-organisatrices de Kohonen pour la création d'un ensemble de partions sur lesquelles le processus de prédiction se concrétise via l'application d'un auto-encodeur spécifique pour chaque partition. Les résultats de cette phase sont représentés par la moyenne pondérée des différentes performances relatives à chaque cluster (CDAE). Cette moyenne concerne les résultats de huit partitions de l'auto-codeur profond. Le troisième modèle (CDAE*) présente les résultats moyens des tests des trois meilleurs clusters (environ 80% de tous les services sont regroupés dans les trois meilleurs clusters). Le même principe est appliqué pour les différents modèles d'auto-encodeurs à base de réseau antagoniste génératif (ADAE, ACDAE et ACDAE*).

Pour évaluer les performances de notre approche proposée ainsi que ces différentes variantes, nous avons utilisé plusieurs méthodes de prédiction de la QoS, à base de FC, que nous avons épuisé à partir l'état de l'art présenté dans les deux chapitres précédents (Chapitre 5 et 6). Ces travaux de références sélectionnés, à des fins de comparaison, se veulent un « cocktail » entre les algorithmes traditionnels UPCC (Shao et al., 2007), IPCC (Sarwar et al., 2001), les algorithmes à base d'apprentissage profond LDCF (Zhang et al., 2019c) et NCF (He et al., 2017), et même avec les travaux autours de prédiction via les séries chronologiques, tel que, Lasso (Wang et al., 2016).

Il est important de mentionner qu'afin d'avoir des comparaisons plus exactes et plus sûres, en termes de RMSE et MAE, nous avons réexécuté localement la plupart des algorithmes relatifs à ces approches par rapport à notre stratégie de tests. Pour les premières méthodes (IPCC, UPCC, NCF et JCM), nous avons implémenté complètement leurs algorithmes. Tandis que pour la méthode du Lasso, nous avons considéré la médiane des différentes performances dans le cas où la densité de données vaut 50% (car elle est indisponible).

Tableau 7.5 – Comparaisons de performances de prédiction (sur le paramètre RT)

bruit	Densité=50%		Densité=80%		Densité=50%		Densité=80%	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
	UPCC				**IPCC**			
0%	3,034	1,470	3,032	1,467	2,951	1,396	2.925	1,372
	Lasso				**CAE**			
0%	2,872	1,021	2,572	0,893	3,825	1,892	2,803	1,250
	LDCF				**NCF**			
0%	**2.705**	**1,585**	**2,484**	**1,364**	2.745	1.622	2.527	1.400
	DAE				**CDAE**			
0%	**2,761**	**1,407**	**2,562**	**1,270**	**2,558**	**1,383**	**2,325**	**1,188**
20%	2,819	1,501	2,616	1,367	2,581	1,398	2,379	1,236
50%	3,026	1,681	2,807	1,511	2,743	1,501	2,577	1,380
80%	3,575	2,057	3,310	1,894	2,981	1,612	2,862	1,527
	ADAE				**ACDAE**			
0%	2,700	1,376	2.590	1,283	2,387	1,520	2.381	1.426
20%	2,613	**1,391**	2,238	1,169	2,371	1.640	2,172	1,371
50%	**2,508**	1,393	**2,120**	**1,141**	**2,198**	**1,124**	**1.914**	**0.920**
80%	2,994	1,722	2,908	1,663	2,725	1,638	2.779	1,470
	CDAE*				**ACDAE***			
0%	**1,762**	**0,771**	**1,700**	**0,748**	1,551	0,849	1,587	0,821
20%	1,806	0,802	1,731	0,771	1,630	0,955	1,674	0,900
50%	1,917	0,858	1,829	0.822	**1,499**	**0,757**	**1,557**	**0.619**
80%	2,128	0,945	2,071	0,930	2,001	0,895	1,917	0,927

Le Tableau 7.5 présente les performances en termes, de RMSE et MAE, pour les différentes méthodes de référence dans la prédiction de la QoS, ainsi que nos approches proposées et leurs variantes. Les expériences ont été effectuées sur la qualité du « temps de réponse » pour deux densités différentes de la base d'apprentissage. A partir de ces résultats nous constatons les observations suivantes:

— ***Country AE*–CAE**: nous remarquons que les performances de notre premier modèle, par rap-

port aux travaux de l'état-de-l'art, ne sont pas à la hauteur des attentes. En effet, mise à part une légère amélioration de $(RMSE = 7\%, MAE = 14\%)_{D=80\%}$ et de $(RMSE = 4\%, MAE = 8\%)_{D=80\%}$ par rapport, respectivement, aux méthodes UPCC et IPCC, notre premier modèle présente des taux d'amélioration négatifs par rapport à l'ensemble des autres méthodes. Ces résultats sont dus à la légèreté architecturale du modèle choisi. Nous rappelons que le partitionnement, dans cette approche, s'est effectué sur une localisation par pays, et que le modèle de factorisation matricielle proposé est basé sur l'apprentissage d'un auto-encodeur à une seule couche cachée.

— ***Deep AE*** **–DAE**: nous constatons que ce modèle dépasse largement les approches de FC à base du voisinage, ainsi que notre premier modèle pour les deux taux de densités; bien que le taux de bruit appliqué sur les données a atteint les 50%. Dans le cas où aucun bruit n'est appliqué, les taux d'amélioration, sur l'ensemble (IPCC, UPC et Country AE), varient de 6% à 27%, pour les RMSE, tandis qu'ils sont entres -1% à 25%, pour le MAE. En ce qui concerne la méthode Lasso, les performances sont très étroites pour le RMSE, $(D_{50\%} = 0.4\%, D_{80\%} = 3\%)$; tandis qu'elles sont négatives pour les MAE, $(D_{50\%} = -42\%, D_{80\%} = -37\%)$. Au-delà de ce taux de bruit, les performances commencent à se détériorer. Concernant la comparaison avec les méthodes à base d'apprentissage profond, le taux d'amélioration est de $(D_{50\%} = -2\%, D_{80\%} = -3\%)_{RMSE}$ et de $(D_{50\%} = 11\%, D_{80\%} = 23\%)_{MAE}$ par rapport à la méthode LDCF. Pour l'approche NCF les variations sont entre $(D_{50\%} \simeq 0\%, D_{80\%} = -1.38\%)_{RMSE}$ et de $(D_{50\%} = 13\%, D_{80\%} = 9\%)_{MAE}$. Nous remarquons que les performances, en termes de RMSE, sont négatives par rapport aux performances, en termes de MAE. L'interprétation la plus plausible qui justifie cette différence, est que notre modèle a généré des intervalles trop larges entre les valeurs réelles et prédites. Pour les taux de bruit supérieurs à 0% les performances vont se dégrader. Le Tableau 7.6 détaille les taux d'amélioration de nos méthodes par rapport à l'approche LDCF. En effet, le choix de comparaison s'est porté sur cette méthode car elle dépasse, en termes de performances, les autres méthodes que nous avons choisies.

Tableau 7.6 – Taux d'amélioration de nos méthodes par rapport à la méthode LDCF

Bruit	Dénsité=50%		Dénsité=80%		Dénsité=50%		Dénsité=80%	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
	DAE				CDAE			
0%	-2,07%	11,23%	-3,14%	6,89%	5,43%	12,74%	6,40%	12,90%
20%	-4,21%	5,30%	-5,31%	-0,22%	4,58%	11,80%	4,23%	9,38%
50%	-11,87%	-6,06%	-13,00%	-10,78%	-1,40%	5,30%	-3,74%	-1,17%
80%	-32,16	-29,78%	-33,25%	-38,86%	-10,20%	-1,70%	-15,22%	-11,95%
	ADAE				CADAE			
0%	0,18%	13,19%	-4,27%	5,94%	11,76%	4,10%	4,15%	-4,55%
20%	3,40%	12,24%	9,90%	14,30%	12,35%	-3,47%	12,56%	-0,51%
50%	7,28%	12,11%	14,65%	16,35%	18,74%	29,09%	22,95%	32,55%
80%	-10,68%	-8,64%	-17,07%	-21,92%	-0,74%	-3,34%	-11,88%	-7,77%
	CDAE*				CADAE*			
0%	34,86%	51,36%	31,56%	45,16%				
20%	33,23%	49,40%	30,31%	43,48%				
50%	29,13%	45,87%	26,37%	39,74%				
80%	21,33%	40,38%	16,63%	31,82%				

— ***Cluster AE*** **–CDAE**: les performances pour notre CDAE, par rapport à la méthode LDCF, sont largement supérieures pour presque tous les cas, sauf dans le cas où le taux de bruit est à 80%. Nous constatant que la meilleure configuration pour ce modèle est enregistrée dans le cas où le bruit vaut 0%.

— ***Adversarial AE*** **–ADAE**: l'adversarial deep auto-encodeur présente de meilleures performances par rapport aux différentes approches présentées précédemment. Nous remarquons que le meilleur modèle est enregistré dans le cas où le taux de bruit avoisine les 50%. Cela donnera plus de robustesse à notre modèle car nous avons évité le risque de l'auto-encodeur jouera le rôle d'une matrice identitaire.

— ***Adversarial Cluster AE*** **–ACDAE**: automatiquement, le modèle en question suivra la même tangente, en termes de performances, que le CDAE par rapport au DAE.

Nous rappelons que les performances pour les deux derniers modèles (CDAE* et CADAE*), sont calculées seulement sur la base des trois meilleures partitions, c'est-à-dire, l'apprentissage a considéré que les partitions les plus peuplées.

7.4 Problème du démarrage à froid

Pour traiter le problème du démarrage à froid, nous adoptons la stratégie décrite dans l'algorithme 7. Plus précisément, nous estimons les valeurs en termes de QoS initiales d'un nouveau service s ($\mathcal{V}(s) = \{qos_1, \cdots, qos_m\}$), comme suit:

Étape.1 Tout d'abord, nous utilisons les caractéristiques spatiales d'un service pour déterminer le cluster le plus représentatif (en termes du pays, du système autonome là où le service est logé, ou du fournisseur). Cette étape est effectuée en appliquant la première partie de l'algorithme 7 (de la ligne 1 à la ligne 13). Techniquement, nous vérifions si le fournisseur réel du nouveau service ($\mathcal{P}(s)$) appartient à l'ensemble initial de nos fournisseurs existants. Si c'est le cas, nous calculons le taux de représentativité du fournisseur actuel pour chaque cluster ($\alpha_{c,s}$, $\forall c \in C^S$). Si ce n'est pas le cas, nous testons si le AS du nouveau service ($\mathcal{A}(s)$) appartient à notre base de test. Si c'est le cas nous calculons le taux de représentativité de l'AS réel pour chaque cluster, et nous calculons la moyenne des valeurs de QoS initiales $\mathcal{V}(s)$ par rapport à tous les vecteurs représentatifs des différents clusters. Si ce n'est pas le cas, nous procédons de la même manière pour le pays du service. Dans le cas contraire, où aucune information spatiale n'est disponible, nous recherchons le fournisseur ayant le service le plus proche au service actuel, en termes de distance géographique, et nous calculons ses valeurs de QoS initiales comme mentionné précédemment.

Étape.2 Ensuite, nous utilisons les nouvelles valeurs de QoS (générées lors de la première étape) comme entrée de tous les auto-encodeurs formés et nous prenons la moyenne pondérée des résultats obtenus (ligne 14).

Les mêmes étapes s'appliqueront dans le cas où nous avons un nouvel utilisateur ($\mathcal{V}(u) = \{qos_1, \cdots, qos_n\}$).

À noter que, dans la phase d'apprentissage, nous utilisons environ 60% de l'ensemble de données entier comme base d'apprentissage pour adapter les paramètres du classificateur SOM. Dans la phase de test, nous utilisons environ 40% d'exemples pour évaluer le rendement de notre classificateur entièrement spécifié. Comme le montre le tableau 7.7, le RMSE, l'MAE et le temps d'essais, montre l'impact positif des caractéristiques contextuelles sur la résolution du problème de démarrage à froid. Pour cela, nous nous concentrons non seulement sur la vérification du cluster le plus représentatif en termes de caractéristiques géographiques (étape 1), mais nous utilisons aussi les auto-encodeurs déjà formés (le deep auto-encodeur et l'adversarial auto-encodeur), après avoir utilisé l'algorithme 4 et l'algorithme 6, afin de calculer la moyenne des résultats retournées.

66. Temps spécifié en seconde

Tableau 7.7 – Problème du démarrage à froid

Étape Cold-Start	RMSE	MAE	Temps[66]
Étape 1	3.877	1.951	1.201
Étape 2 (Clustered AE)	3.010	1.420	2.194
Étape 2 (Adversarial AE)	2.010	0.891	2.810

7.5 Menaces à la validité (Threats To Validity)

Les résultats que nous avons obtenus à travers cette thèse sont assez concluants. Cependant, il y a un certain nombre d'éléments à discuter et à prendre en considération pour la validité des différentes expériences. La validité d'une recherche est définie par (Wacheux, 1996) comme « la capacité des instruments à apprécier effectivement et réellement l'objet de la recherche pour lequel ils ont été créés ». Selon (Wohlin et al., 2012), il existe trois types de validité: la validité de construction (ou *construct validity*), la validité interne (ou *internal validity*), et la validité externe (ou *external validity*).

— **Validité interne**: La validité interne consiste à s'assurer de la pertinence et de la cohérence interne des résultats générés par une étude. Les auteurs dans (LeCompte and Goetz, 1982) indiquent que la validité interne est souvent considérée comme une force des recherches subjectives ou qualitatives, car elle devrait garantir une interrelation forte entre les observations empiriques et les concepts théoriques.

— **Validité externe**: La validité externe représente, selon (Feldt and Magazinius, 2010), le degré auquel les résultats d'une étude peuvent être généralisés à des individus, des situations ou des procédés au-delà de l'étude. Elle est considérée comme la principale limite de l'étude de cas qui, par définition, fait référence à l'étude d'un contexte particulier.

Pour notre cas de figure, seules les menaces de validité interne et externe seront développées. Par conséquent, nous ne traiterons pas de la validité du construit car elle a été prise en considération tout au long de notre processus de développement. En effet, la validité du construit, et par définition, vise à préciser et à délimiter les concepts étudiés afin de définir ce qu'il convient d'observer comme paramètres pour les modèles développés.

7.5.1 Validité interne

Les menaces de validité interne sont intrinsèquement liées à l'authenticité des différentes expériences. Afin de sélectionner les différents paramètres pour nos modèles, nous avons essayé d'être le plus objectif possible. Cependant, et malgré le fait que nous avons effectué plusieurs pré-entraînements pour sélectionner la meilleure configuration pour ces paramètres, rien n'empêche que d'autres valeurs différentes des hyperparamètres, pourraient conduire à des résultats différents. Afin d'atténuer ces menaces réelles, nous avons divisé l'ensemble du jeu de données en deux parties avec des proportions variables. La première partie est utilisée pour la phase d'apprentissage, tandis que la deuxième partie est réservée pour la phase des tests. D'autre part, cette menace est considérablement atténuée par le fait que nous avons utilisé le principe d'une validation croisée à k-plis. En ce qui concerne le modèle auto-encodeur profond clustérisé, et en plus des remarques précédentes, les menaces de validité interne sont atténuées puisque le nombre de neurones n'est pas fixe pour tous les clusters, mais il est proportionnel à la taille des différentes partitions.

7.5.2 Validité externe

Les menaces liées à la validité externe concernent le degré de généralisation de nos propres résultats et conclusions par rapport à d'autres contextes. Elles sont liées aux modèles de référence sélectionnés,

à l'ensemble de données QoS utilisé et à l'environnement de configuration expérimentale. Durant cette étude, nous avons réalisé nos différentes expériences sur un grand jeu de données, ce qui nous a permis d'entraîner correctement nos modèles de prédiction. À notre connaissance, il s'agit du seul grand jeu de données, qui s'intéresse à la qualité de service pour les services Web, de disponible publiquement. Bien que nous n'ayons utilisé qu'un seul jeu de données, nous sommes convaincus qu'avec le principe de la validation croisée à k-plis que nous avons adoptée, nous réduirons l'impact en cas d'une généralisation des résultats par rapport à d'autres jeux de données. D'autre part, le fait d'utiliser un auto-encodeur de débruitage en corrompant les données (en forçant aléatoirement certaines des valeurs de la matrice d'entrée à zéro), diminue considérablement les risques d'une validité externe.

7.6 Conclusion

Dans ce dernier chapitre nous avons présenté l'évaluation des différentes approches qui ont été proposées dans cette thèse. Pour la première partie, nous avons présenté les différents jeux de données disponibles dans le contexte des services Web en mettant l'accent sur celui que nous l'avons utilisé. Nous avons aussi détaillé dans cette partie les différentes métriques que nous avons adopté afin d'évaluer nos performances de prédiction. Dans la partie suivante, nous avons mené plusieurs expériences sur des mini batchs de test afin de sélectionner les valeurs optimales des différents hyperparamètres de nos modèles proposés. Ensuite, et afin d'évaluer les performances de l'approche proposée, nous les avons comparées à plusieurs approches de base.

Chapitre 8

Conclusion générale

L'orientation service est un paradigme architectural conçu pour renforcer la fiabilité, la disponibilité et la maintenabilité des systèmes à grande échelle. Cela grâce à de différentes caractéristiques, telles que, la modularité, la réutilisabilité et le couplage faible. La qualité de service (QoS) est un concept sur lequel s'appuie la nature modulaire d'une architecture orientée service. Souvent considérée comme un facteur primordiale dans le processus de mise en place de systèmes orientés services, la QoS permet d'améliorer la *confiance* du consommateur pour l'utilisation de ses systèmes dans leur ensemble. De ce fait, le besoin incessant de connaître les valeurs des différents facteurs en termes de QoS est devenu un champ de recherche à part entière.

8.1 Synthèse

La prédiction des valeurs de la QoS pour les services Web est considérée comme un problème difficile en théorie comme en pratique. Il s'agit, en fait, d'un domaine en cours de développement, comme en confirme la quasi-absence de systèmes de prédiction automatisés déployés dans le monde commercial. La plupart des techniques développées dans ce contexte, s'inspirent des principes définis dans le domaine de la recommandation et notamment du filtrage collaboratif. Cependant, la plupart de ces travaux existants souffrent de problèmes liés à la parcimonie des données, et au de démarrage à froid.

L'objectif principal de notre travail est de fournir une approche de prédiction de la QoS des services Web en se basant sur de différents modèles de réseaux de neurones définis en apprentissage profond. Pour assurer cet objectif, nous avons procédé comme suit:

1) Le but dans notre travail consiste en une amélioration des performances de prédiction par rapport aux différentes approches définies en état-de-l'art. Puisque ce dernier est engrangé de travaux de natures diverses, nous nous sommes intéressés seulement aux approches utilisant le FC comme base de solution. Ce choix a été pris par rapport à plusieurs considérations. En premier lieu, à la nature du sujet qui nous a été attribué où nous devons manipuler que des valeurs numériques, en termes d'invocation de services ainsi que de données relatives aux caractéristiques spatio-temporelles disponibles (il est important de préciser que les caractéristiques temporelles ne sont pas considérées dans cette thèse). Au deuxième lieu, au fait que nous a été recommandé d'épuiser des modèles à partir du domaine d'apprentissage profond pour réaliser nos objectifs. Finalement, c'était par rapport à la nature des problèmes à résoudre (prédiction de valeurs de QoS, démarrage à froid et la parcimonie de données). De ce fait, nous avons tracé un état-de-l'art, que nous le considérons assez détaillé sur les travaux de prédiction de qualité de service autours du FC, et qui s'étale sur deux chapitres (les Chapitres 5 et 6). Dans le premier, nous nous sommes intéressés à deux sous catégories de FC, à savoir, les modèles basés sur le voisinage ainsi qu'à ceux basés sur les séries temporelles. Pour ce dernier, nous avons constaté que les différentes approches utilisent soit un modèle statistique (tel que ARMIA), soit une technique d'apprentissage profond (tels que LSTM, GRU), soit une hybridation entre ces deux méthodes. Cependant, bien que la deuxième

catégorie de FC (à base de voisinage) n'utilise pratiquement pas de techniques d'apprentissage profond, mais nous avons jugé intéressant de l'étudier car elle est souvent combinée avec les méthodes de FC à base de modèles, tels que, les algorithmes de partitionnement et la factorisation matricielle. La deuxième partie de cet état-de-l'art, se focalise sur la prédiction de la QoS via l'utilisation d'un FC basé sur les modèles ainsi que celui basée sur les approches hybrides. Une synthèse détaillée est établie pour chacune de ces catégories. Cette première phase nous a permis de bien cerner notre objet de recherche.

2) Dans notre proposition de solutions, nous avons élaboré une approche de prédiction de QoS fondée sur un apprentissage profond combinée avec la factorisation matricielle, où cette dernière est basée, à la fois, sur une architecture des auto-encodeurs et une technique de partitionnement.
Dans notre première contribution (Smahi et al., 2018), nous avons exploré l'architecture des auto-encodeurs afin de les exploiter comme solution au problème de la factorisation matricielle. L'idée derrière l'utilisation des auto-encodeurs et leur faculté à compacter les vecteurs de grande taille en entrée en des vecteurs à taille réduite, connu sous le nom de vecteur des facteurs latents, dans la partie d'encodage. Ce vecteur réduit sera par la suite utilisé dans un réseau de décodage afin de reproduire le vecteur initial. Les valeurs de ce vecteur à facteurs latents peuvent ensuite être utilisées comme variables explicatives pour prédire les valeurs de QoS. Cela est possible par un produit scalaire entre les facteurs latents des utilisateurs avec celui des services.
La deuxième contribution (Smahi et al., 2021) représente une continuité de la première, où nous avons proposé deux améliorations majeures. La première, est l'utilisation d'un auto-encodeur profond avec plusieurs couches cachées avant et après la couche latente. L'idée par la suite, est d'appliquer cet auto-encodeur sur un ensemble d'information réduit afin d'éliminer le problème de parcimonie de données. La mise en œuvre de cette idée était possible via l'utilisation d'un réseau de neurones pour une classification non supervisée de données. Parmi les différents algorithmes de partitionnement qui existent, nous avons opté pour les réseaux de cartes auto-organisatrices de Kohonen. Les différentes caractéristiques de ce type de modèles, par rapport à la nature de notre problème, nous ont rendu notre choix plus facile. La caractéristique principale dans ce type de réseau est qu'il garantit l'aspect déterministe des résultats de classifications si nous lui appliquons les mêmes données d'initialisation. De ce fait, nous avons proposé un algorithme d'initialisation en utilisant les facteurs de localisation, telles que, l'adresse IP des fournisseurs d'accès à Internet, la ville, etc. Après la phase d'apprentissage, nous avons appliqué un autre algorithme d'apprentissage automatique pour la création finale des partitions. Le processus de prédiction sera le même que dans la première proposition.
La troisième proposition consistait en une combinaison entre le modèle des auto-encodeurs avec celui des réseaux antagonistes génératifs. L'idée de base de ce modèle, non supervisé, est inspirée de la notion de l'équilibre de Nash en théorie des jeux, où deux joueurs, le générateur et le discriminateur, s'affrontent afin d'obtenir un équilibre à la fin. Dans notre cas le générateur jouera le rôle d'un encodeur pour l'auto-encodeur. De ce fait la phase d'apprentissage se fera l'encodeur de l'auto-encodeur qui est en même temps le générateur du réseau GAN. L'apprentissage des autres parties, à savoir, le décodeur pour le premier modèle et le discriminateur pour le second se fait dans leurs propres modèles.
Dans ces différentes propositions, nous avons aussi abordé d'autres problèmes relatifs à la prédiction de la QoS, à savoir, le problème du démarrage à froid et de rareté de données. La solution pour le premier était possible grâce à l'utilisation des facteurs de localisation combinée avec les résultats de classification. Tandis que la solution du deuxième problème était via l'utilisation des résultats de l'algorithme de partitionnement, où au lieu de travailler sur toute la matrice d'invocation, le traitement s'effectue pour la matrice de chaque partition à part.

3) Dans le cadre de nos études expérimentales, nous avons essayé d'optimiser au maximum les performances de prédiction. Ces performances sont étroitement liées à nos différents choix des hyperparamètres des différents modèles que nous avons développés. Finalement, plusieurs expériences ont été menées, dans un processus de validation croisée, pour évaluer les performances de cette approche de prédiction. De plus, nous avons aussi exploré de nombreuses variantes de

cette méthode (tel que l'ajout de bruit à des pourcentages différents) et comparé leurs performances avec celles des méthodes de l'état-de-l'art. Les résultats expérimentaux ont montré que nos méthodes surpassent les méthodes existantes en termes de précision des valeurs de QoS.

Ainsi avec toutes ces étapes nous avons essayé d'apporter des solutions à l'ensemble des questions de recherches que nous avons présentées à l'introduction du manuscrit, à savoir, le choix du modèle issu du domaine d'apprentissage profond, le traitement des problèmes relatifs à la parcimonie de données et le démarrage à froid, et enfin le paramétrage des différents modèles proposés.

8.2 Perspectives

Dans le prolongement de ce travail, nous aimerions d'abord étudier l'utilisation d'architectures alternatives pour les modèles d'apprentissage profond, telles que les auto-encodeurs empilés ou les réseaux de neurones récurrents. Ces architectures peuvent donner des résultats de prédiction plus précis que les auto-encodeurs utilisés dans ce travail. Tout en mettant l'accent sur l'interprétation, en termes de représentativité, des différentes valeurs des vecteurs de facteurs latents générées. Nous essayerons dans nos futurs travaux d'incorporer d'autres caractéristiques que de localisation, à savoir, le temps, la confiance, etc. En fin, nous prévoyons de combiner le processus de prédiction de la QoS avec celui de la qualité d'expérience (QoE) des services Web. Nous avons l'intention d'utiliser plusieurs outils définis en text mining pour analyser, par exemple, les commentaires des utilisateurs afin d'estimer les scores en termes de QoE pour les services Web. Nous considérons que la combinaison de la QoS et de la prédiction QoE contribuera à améliorer les systèmes de prédiction pour les services Web.

Liste des publications

— **Revues internationales avec comité de lecture**
Smahi, M. I., Hadjila, F., Tibermacine, C., & Benamar, A. (2021). A deep learning approach for collaborative prediction of Web service QoS. Service Oriented Computing and Applications, 15(1), 5–20. https://doi.org/10.1007/s11761-020-00304-y

— **Conférences internationales avec comité de lecture**
Smahi, M. I., Hadjila, F., Tibermacine, C., Merzoug, M., & Benamar, A. (2018). An encoder-decoder architecture for the prediction of web service QoS. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11116 LNCS, 74–89. https://doi.org/10.1007/978-3-319-99819-0_6

Bibliographie

Abbasi, M. A., Tang, J., and Liu, H. (2014). Trust-aware recommender systems. *Machine Learning book on computational trust, Chapman & Hall/CRC Press.* *Cité page 35*

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., and Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938. *Cité page 44*

Adeleye, O., Yu, J., Yongchareon, S., Sheng, Q. Z., and Yang, L. H. (2019). A Fitness-Based Evolving Network for Web-APIs Discovery. In *Proceedings of the Australasian Computer Science Week Multiconference*, number May in ACSW 2019, pages 1–10, New York, NY, USA. Association for Computing Machinery. *Cité page 66*

Adomavicius, G., Sankaranarayanan, R., Sen, S., and Tuzhilin, A. (2005). Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems*, 23(1):103–145. *Cité page 24*

Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749. *Cité page 25*

Alain, G. and Bengio, Y. (2014). What regularized auto-encoders learn from the data-generating distribution. *The Journal of Machine Learning Research*, 15(1):3563–3593. *Cité page 60*

Alloghani, M., Al-Jumeily, D., Mustafina, J., Hussain, A., and Aljaaf, A. J. (2020). *A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science*, pages 3–21. Springer International Publishing, Cham. *Cité page 50*

Alonso, G., Casati, F., Kuno, H., and Machiraju, V. (2004). *Web Services: concepts, architectures and applications*. Springer Berlin Heidelberg, Berlin, Heidelberg. *Cité page 14*

Amin, A., Colman, A., and Grunske, L. (2012). An Approach to Forecasting QoS Attributes of Web Services Based on ARIMA and GARCH Models. In *2012 IEEE 19th International Conference on Web Services*, pages 74–81. IEEE. *2 citations pages 72 et 78*

Anithadevi, N. and Sundarambal, M. (2019). A design of intelligent qos aware web service recommendation system. *Cluster Computing*, 22(6):14231–14240. *Cité page 97*

Anwar, K., Siddiqui, J., and Sohail, S. S. (2020). Machine learning-based book recommender system: A survey and new perspectives. *International Journal of Intelligent Information and Database Systems*, 13(2-4):231–248. *Cité page 102*

Araban, S. and Sterling, L. (2004). Measuring quality of service for contract aware web-services. In *First Australian Workshop on Engineering Service-Oriented Systems*, pages 54–56. *Cité page 21*

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR. *Cité page 62*

Arora, S., Ge, R., Moitra, A., and Sachdeva, S. (2012). Provable ica with unknown gaussian noise, and implications for gaussian mixtures and autoencoders. *arXiv preprint arXiv:1206.5349*. *Cité page 122*

Aurrecoechea, C., Campbell, A., and Hauw, L. (1995). A survey of quality of service architectures. *MPG Group, University of Lancaster, Tech. Report MPG-95-18*. *Cité page 18*

Bao, J., Zheng, Y., Wilkie, D., and Mokbel, M. F. (2013). A survey on recommendations in location-based social networks. *ACM Transaction on Intelligent Systems and Technology*, pages 1–30. *2 citations pages 28 et 82*

Bell, M. (2008). *Service-oriented modeling: service analysis, design, and architecture*. John Wiley & Sons. *Cité page 1*

Ben Halima, R. (2009). *Conception, implantation et expérimentation d'une architecture en bus pour l'auto-réparation des applications distribuées à base de services web*. PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier. *2 citations pages 9 et 20*

Benesty, J., Chen, J., Huang, Y., and Cohen, I. (2009). *Pearson Correlation Coefficient*, pages 1–4. Springer Berlin Heidelberg, Berlin, Heidelberg. *Cité page 28*

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828. *Cité page 61*

Bhatnagar, V. (2017). *Collaborative Filtering Using Data Mining and Analysis*. Advances in Data Mining and Database Management. IGI Global. *4 citations pages 2, 23, 24, et 26*

Billsus, D., Pazzani, M. J., et al. (1998). Learning collaborative information filters. In *Icml*, volume 98, pages 46–54. *Cité page 89*

Bishop, C. M. (2006). *Pattern recognition and Machine learning*. Springer-Verlag. *Cité page 50*

Bloomberg, J. (2013). *The agile architecture revolution: how cloud computing, rest-based SOA, and mobile computing are changing enterprise IT*. John Wiley & Sons. *Cité page 15*

Bobadilla, J., Ortega, F., Gutiérrez, A., and Alonso, S. (2020). Classification-based Deep Neural Network Architecture for Collaborative Filtering Recommender Systems. *International Journal of Interactive Multimedia and Artificial Intelligence*, 6(1):68. *Cité page 27*

Bora, A. and Bezboruah, T. (2015). A comparative investigation on implementation of restful versus soap based web services. *International Journal of Database Theory and Application*, 8(3):297–312. *Cité page 17*

Bouguettaya, A., Singh, M., Huhns, M., Sheng, Q. Z., Dong, H., Yu, Q., Neiat, A. G., Mistry, S., Benatallah, B., Medjahed, B., Ouzzani, M., Casati, F., Liu, X., Wang, H., Georgakopoulos, D., Chen, L., Nepal, S., Malik, Z., Erradi, A., Wang, Y., Blake, B., Dustdar, S., Leymann, F., and Papazoglou, M. (2017). A Service Computing Manifesto: The next 10 Years. *Communications of the ACM*, 60(4):64–72. *3 citations pages 1, 3, et 66*

Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons. *Cité page 73*

Breese, J. S., Heckerman, D., and Kadie, C. (1998). Empirical Analysis of Predictive Algorithms for Collaborative Filtering. *arXiv preprint arXiv:1301.7363*, pages 43–52. *5 citations pages 28, 30, 38, 40, et 82*

Brescia, M., Cavuoti, S., D'Angelo, G., D'Abrusco, R., Deniskina, N., Garofalo, M., Laurino, O., Longo, G., Nocella, A., and Skordovski, B. (2009). The vo-neural project:. recent developments and some applications. *Memorie della Societa Astronomica Italiana*, 80:565. *Cité page 52*

Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370. *Cité page 25*

Canny, J. (2002). Collaborative filtering with privacy via factor analysis. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '02*, SIGIR '02, page 238, New York, New York, USA. ACM Press. *Cité page 27*

Cao, B., Liu, X. F., Rahman, M. M., Li, B., Liu, J., and Tang, M. (2020). Integrated content and network-based service clustering and web apis recommendation for mashup development. *IEEE Transactions on Services Computing*, 13(1):99–113. *Cité page 95*

Cao, J., Wu, Z., Wang, Y., and Zhuang, Y. (2013). Hybrid Collaborative Filtering algorithm for bidirectional Web service recommendation. *Knowledge and Information Systems*, 36(3):607–627. *Cité page 86*

Cardoso, J., Sheth, A., Miller, J., Arnold, J., and Kochut, K. (2004). Quality of service for workflows and web service processes. *Journal of Web Semantics*, 1(3):281–308. *Cité page 19*

Cavallo, B., Di Penta, M., and Canfora, G. (2010). An empirical comparison of methods to support QoS-aware service selection. In *Proceedings of the 2nd International Workshop on Principles of Engineering Service-Oriented Systems - PESOS '10*, page 64, New York, New York, USA. ACM Press. *2 citations pages 78 et 93*

Chattopadhyay, S., Adak, C., and Chowdhury, R. R. (2021). Fes: A fast efficient scalable qos prediction framework. *arXiv preprint arXiv:2103.07494*. *Cité page 97*

Chavda, K. F. (2004). Anatomy of a web service. *Journal of Computing Sciences in Colleges*, 19(3):124–134. *Cité page 11*

Chen, D., Gao, M., Liu, A., Chen, M., Zhang, Z., and Feng, Y. (2019a). A Recurrent Neural Network Based Approach for Web Service QoS Prediction. In *2019 2nd International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pages 350–357. IEEE. *Cité page 78*

Chen, H., Tang, F., Tino, P., and Yao, X. (2013). Model-based kernel for efficient time series analysis. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, number August in KDD '13, pages 392–400, New York, NY, USA. Association for Computing Machinery. *Cité page 75*

Chen, K., Mao, H., Shi, X., Xu, Y., and Liu, A. (2017a). Trust-Aware and Location-Based Collaborative Filtering for Web Service QoS Prediction. In *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 143–148. IEEE. *Cité page 98*

Chen, L., Xie, F., Zheng, Z., and Wu, Y. (2019b). Predicting Quality of Service via Leveraging Location Information. *Complexity*, 2019:1–16. *Cité page 97*

Chen, X., Liu, X., Huang, Z., and Sun, H. (2010). RegionKNN: A Scalable Hybrid Collaborative Filtering Algorithm for Personalized Web Service Recommendation. In *2010 IEEE International Conference on Web Services*, pages 9–16. IEEE. *Cité page 99*

Chen, X., Zheng, Z., Yu, Q., and Lyu, M. R. (2014). Web Service Recommendation via Exploiting Location and QoS Information. *IEEE Transactions on Parallel and Distributed Systems*, 25(7):1913–1924. *4 citations pages 90, 99, 100, et 101*

Chen, Z., Shen, L., and Li, F. (2017b). Exploiting Web service geographical neighborhood for collaborative QoS prediction. *Future Generation Computer Systems*, 68:248–259. *4 citations pages 81, 82, 98, et 103*

Chen, Z., Shen, L., and Li, F. (2019c). Your neighbors are misunderstood: On modeling accurate similarity driven by data range to collaborative web service QoS prediction. *Future Generation Computer Systems*, 95:404–419. *2 citations pages 81 et 97*

Chen, Z., Shen, L., Li, F., and You, D. (2017c). Your neighbors alleviate cold-start: On geographical neighborhood influence to collaborative web service QoS prediction. *Knowledge-Based Systems*, 138:188–201. *Cité page 98*

Chen, Z., Shen, L., Li, F., You, D., and Mapetu, J. P. B. (2020). Web service QoS prediction: when collaborative filtering meets data fluctuating in big-range. *World Wide Web*, 23(3):1715–1740. *3 citations pages 79, 80, et 85*

Chen, Z., Shen, L., You, D., and Li, F. (2016). A user dependent Web service QoS collaborative prediction approach using neighborhood regularized matrix factorization. In *2016 IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 316–321. IEEE. *Cité page 98*

Chen, Z., Shen, L., You, D., Li, F., and Ma, C. (2017d). Alleviating data sparsity in web service qos prediction by capturing region context influence. In Wang, S. and Zhou, A., editors, *Collaborate Computing: Networking, Applications and Worksharing*, pages 540–556, Cham. Springer International Publishing. *Cité page 98*

Chien, Y.-H. and George, E. I. (1999). A bayesian model for collaborative filtering. In *AISTATS*. *Cité page 39*

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*. *2 citations pages 57 et 75*

Clevert, D., Unterthiner, T., and Hochreiter, S. (2016). Fast and accurate deep network learning by exponential linear units (elus). In Bengio, Y. and LeCun, Y., editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. *Cité page 119*

Costa, A. d. R. and Demazeau, Y. (1996). Toward a formal model of multi-agent systems with dynamic organizations. In *Proceedings of the International Conference on Multi-Agent Systems, MIT Press, Kyoto, Japan*, volume 431. *Cité page 19*

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., and Bharath, A. A. (2018). Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65. *2 citations pages 61 et 62*

Cui, Z., Xu, X., Xue, F., Cai, X., Cao, Y., Zhang, W., and Chen, J. (2020). Personalized Recommendation System Based on Collaborative Filtering for IoT Scenarios. *IEEE Transactions on Services Computing*, 13(4):685–695. *Cité page 97*

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314. *Cité page 47*

Daigneau, R. (2012). *Service Design Patterns: fundamental design solutions for SOAP/WSDL and restful Web Services*. Addison-Wesley. *Cité page 13*

Deng, S., Huang, L., and Xu, G. (2014a). Social network-based service recommendation with trust enhancement. *Expert Systems with Applications*, 41(18):8075–8084. *Cité page 99*

Deng, S., Wu, H., Hu, D., and Leon Zhao, J. (2016). Service selection for composition with qos correlations. *IEEE Transactions on Services Computing*, 9(2):291–303. *Cité page 3*

Deng, S.-G., Huang, L.-T., Wu, J., and Wu, Z.-H. (2014b). Trust-Based Personalized Service Recommendation: A Network Perspective. *Journal of Computer Science and Technology*, 29(1):69–80. *Cité page 86*

Deshpande, M. and Karypis, G. (2004). Item-based top- N recommendation algorithms. *ACM Transactions on Information Systems*, 22(1):143–177. *2 citations pages 26 et 27*

Ding, S., Li, Y., Wu, D., Zhang, Y., and Yang, S. (2018). Time-aware cloud service recommendation using similarity-enhanced collaborative filtering and arima model. *Decision Support Systems*, 107:103–115. *3 citations pages 74, 78, et 98*

Dobson, G., Lock, R., and Sommerville, I. (2005). Qosont: a qos ontology for service-centric systems. In *31st EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 80–87. *2 citations pages 20 et 21*

Dziugaite, G. K., Roy, D. M., and Ghahramani, Z. (2015). Training generative neural networks via maximum mean discrepancy optimization. *arXiv preprint arXiv:1505.03906*. *Cité page 62*

Eirinaki, M., Gao, J., Varlamis, I., and Tserpes, K. (2018). Recommender Systems for Large-Scale Social Networks: A review of challenges and solutions. *Future Generation Computer Systems*, 78:413–418. *Cité page 35*

Ekstrand, M. D. (2011). Collaborative Filtering Recommender Systems. *Foundations and Trends® in Human–Computer Interaction*, 4(2):81–173. *Cité page 35*

Erl, T. (1900). *Service-oriented architecture: concepts, technology, and design*. Pearson Education India. *Cité page 10*

Erl, T. (2005). *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR, USA. *Cité page 1*

FanJiang, Y.-Y., Syu, Y., and Huang, W.-L. (2020). Time series qos forecasting for web services using multi-predictor-based genetic programming. *IEEE Transactions on Services Computing*, pages 1–1. *Cité page 93*

Fayon, D. (2010). Web 2.0 et au-delà. *Economica, 2e éd.* *Cité page 1*

Feldt, R. and Magazinius, A. (2010). Validity threats in empirical software engineering research-an initial survey. In *Seke*, pages 374–379. *Cité page 129*

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*. University of California, Irvine. *3 citations pages 14, 15, et 16*

Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, 29(2–3):131–163. *Cité page 38*

Ghafouri, S. H., Hashemi, S. M., and Hung, P. C. K. (2020). A Survey on Web Service QoS Prediction Methods. *IEEE Transactions on Services Computing*, pages 1–1. *4 citations pages 71, 86, 94, et 99*

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy. PMLR. *3 citations pages 53, 102, et 122*

Godse, M., Bellur, U., and Sonar, R. (2010). Automating QoS Based Service Selection. In *2010 IEEE International Conference on Web Services*, pages 534–541. IEEE, IEEE. *Cité page 78*

Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70. *2 citations pages 2 et 26*

Golovko, V. A. (2017). Deep learning: an overview and main paradigms. *Optical Memory and Neural Networks*, 26(1):1–17. *2 citations pages 48 et 55*

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. *7 citations pages 44, 55, 56, 57, 58, 102, et 124*

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27. *3 citations pages 4, 61, et 62*

Gouvert, O. (2019). *Factorisation bayésienne de matrices pour le filtrage collaboratif*. PhD thesis, Université de Toulouse. *Cité page 40*

Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649. *Cité page 56*

Guan, N., Tao, D., Luo, Z., and Yuan, B. (2012). Online nonnegative matrix factorization with robust stochastic approximation. *IEEE Transactions on Neural Networks and Learning Systems*, 23(7):1087–1099. *Cité page 90*

Hafid, A., von Bochmann, G., and Kerherve, B. (1996). A quality of service negotiation procedure for distributed multimedia presentational applications. In *Proceedings of 5th IEEE International Symposium on High Performance Distributed Computing*, pages 330–339. *Cité page 18*

Hanna, S. (2008). *Web services robustness testing*. PhD thesis, Durham University. *Cité page 12*

Hartigan, J. A. and Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108. *Cité page 104*

Hashimi, S. (2003). Service-oriented architecture explained. *O'Reilly ONDot Net. com, August*, 18. *Cité page 2*

Hasnain, M., Seung Ryul, J., Muhammad Fermi, P., and Imran, G. (2020). Performance Anomaly Detection in Web Services: an RNN-based Approach Using Dynamic Quality of Service Features. *Computers, Materials & Continua*, 64(2):729–752. *2 citations pages 77 et 78*

He, P., Zhu, J., Xu, J., and Lyu, M. R. (2014). A hierarchical matrix factorization approach for location-based web service qos prediction. In *2014 IEEE 8th International Symposium on Service Oriented System Engineering*, pages 290–295. *Cité page 98*

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. (2017). Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 173–182, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee. *2 citations pages 90 et 126*

Hecht-Nielsen, R. (1992). Theory of the backpropagation neural network. In Wechsler, H., editor, *Neural Networks for Perception*, pages 65–93. Academic Press. *Cité page 118*

Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, page 230–237, New York, NY, USA. Association for Computing Machinery. *Cité page 26*

Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53. *3 citations pages 41, 79, et 115*

Hill, W., Stead, L., Rosenstein, M., and Furnas, G. (1995). Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, volume 1 of *CHI '95*, pages 194–201. ACM Press/Addison-Wesley Publishing Co. *Cité page 26*

Hinton, G. E. (2006). Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507. *2 citations pages 4 et 57*

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507. *Cité page 118*

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780. *2 citations pages 57 et 75*

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558. *Cité page 45*

Hopfield, J. J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences of the United States of America*, 81(10 I):3088–3092. *Cité page 45*

Hu, Y., Peng, Q., and Hu, X. (2014). A Time-Aware and Data Sparsity Tolerant Approach for Web Service Recommendation. In *2014 IEEE International Conference on Web Services*, pages 33–40. IEEE. *Cité page 86*

Hu, Y., Peng, Q., Hu, X., and Yang, R. (2015). Web Service Recommendation Based on Time Series Forecasting and Collaborative Filtering. In *2015 IEEE International Conference on Web Services*, pages 233–240. IEEE. *3 citations pages 74, 78, et 98*

Huang, A. (2008). Similarity measures for text document clustering. In *New Zealand Computer Science Research Student Conference, NZCSRSC 2008 - Proceedings*, pages 49–56. *Cité page 29*

Huang, X. (2013). Usageqos: Estimating the qos of web services through online user communities. *ACM Trans. Web*, 8(1). *Cité page 86*

Huang, Z., Chen, H., and Zeng, D. (2004). Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):116–142. *Cité page 101*

Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154. *Cité page 55*

Huhns, M. N. and Singh, M. P. (2005). Service-oriented computing: Key concepts and principles. *IEEE Internet computing*, 9(1):75–81. *Cité page 10*

Hwang, S.-Y., Wang, H., Tang, J., and Srivastava, J. (2007). A probabilistic approach to modeling and estimating the qos of web-services-based workflows. *Information Sciences*, 177(23):5484–5503. Including: Mathematics of Uncertainty. *Cité page 19*

Jaeger, M., Rojec-Goldmann, G., and Muhl, G. (2004). Qos aggregation for web service composition using workflow patterns. In *Proceedings. Eighth IEEE International Enterprise Distributed Object Computing Conference, 2004. EDOC 2004.*, pages 149–159. *Cité page 19*

Jalili, M., Ahmadian, S., Izadi, M., Moradi, P., and Salehi, M. (2018). Evaluating collaborative filtering recommender algorithms: A survey. *IEEE Access*, 6:74003–74024. *Cité page 41*

Jannach, D., Zanker, M., Felfernig, A., and Friedrich, G. (2010). *Recommender systems: an introduction*. Cambridge University Press. *Cité page 95*

Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision*, pages 2146–2153. *Cité page 56*

Jatoth, C., Gangadharan, G., and Buyya, R. (2017). Computational Intelligence Based QoS-Aware Web Service Composition: A Systematic Literature Review. *IEEE Transactions on Services Computing*, 10(3):475–492. *Cité page 67*

Jiang, J., Li, W., Dong, A., Gou, Q., and Luo, X. (2020). A Fast Deep AutoEncoder for high-dimensional and sparse matrices in recommender systems. *Neurocomputing*, 412:381–391. *Cité page 102*

Jiang, Y., Liu, J., Tang, M., and Liu, X. (2011). An Effective Web Service Recommendation Method Based on Personalized Collaborative Filtering. In *2011 IEEE International Conference on Web Services*, pages 211–218. IEEE. *Cité page 71*

Jin, L.-j., Machiraju, V., and Sahai, A. (2002). Analysis on service level agreement of web services. *HP June*, 19:1–13. *Cité page 10*

Jin, Y., Wang, K., Zhang, Y., and Yan, Y. (2019). Neighborhood-aware web service quality prediction using deep learning. *EURASIP Journal on Wireless Communications and Networking*, 2019(1):222. *Cité page 97*

Kalaï, A., Zayani, C. A., Amous, I., Abdelghani, W., and Sèdes, F. (2018). Social collaborative service recommendation approach based on user's trust and domain-specific expertise. *Future Generation Computer Systems*, 80:355–367. *Cité page 85*

Kalchbrenner, N. and Blunsom, P. (2013). Recurrent continuous translation models. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1700–1709. *Cité page 56*

Kalman, B. and Kwasny, S. (1992). Why tanh: choosing a sigmoidal function. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 4, pages 578–581 vol.4. *Cité page 49*

Karim, R., Ding, C., and Miri, A. (2015). End-to-End QoS Prediction of Vertical Service Composition in the Cloud. In *2015 IEEE 8th International Conference on Cloud Computing*, pages 229–236. IEEE. *2 citations pages 2 et 86*

Karlik, B. and Olgac, A. V. (2011). Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4):111–122. *Cité page 49*

Keshavarzi, A., Toroghi Haghighat, A., and Bohlouli, M. (2020). Enhanced time-aware QoS prediction in multi-cloud: a hybrid k-medoids and lazy learning approach (QoPC). *Computing*, 102(4):923–949. *Cité page 93*

Keshavarzi, A., Toroghi Haghighat, A., and Bohlouli, M. (2021). Online QoS Prediction in the Cloud Environments Using Hybrid Time-Series Data Mining Approach. *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, 45(2):461–478. *Cité page 97*

Khojamli, H. and Razmara, J. (2021). Survey of similarity functions on neighborhood-based collaborative filtering. *Expert Systems with Applications*, 185(June):115482. *2 citations pages 29 et 79*

Kiang, M. Y., Kulkarni, U. R., and Tam, K. Y. (1995). Self-organizing map network as an interactive clustering tool—an application to group technology. *Decision Support Systems*, 15(4):351–374. *Cité page 63*

Kitts, B., Freed, D., and Vrieze, M. (2000). Cross-Sell: A Fast Promotion-Tunable Customer-Item Recommendation Method Based on Conditionally Independent Probabilities. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '00*, KDD '00, pages 437–446, New York, New York, USA. Association for Computing Machinery. *Cité page 38*

Kline, D. M. and Berardi, V. L. (2005). Revisiting squared-error and cross-entropy functions for training neural network classifiers. *Neural Computing & Applications*, 14(4):310–318. *Cité page 52*

Kohonen, T. (2012). *Self-organization and associative memory*, volume 8. Springer Science & Business Media. *2 citations pages 4 et 63*

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37. *3 citations pages 35, 37, et 88*

Kotsiantis, S. B. (2007). Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in EHealth, HCI, Information Retrieval and Pervasive Technologies*, page 3–24, NLD. IOS Press. *Cité page 38*

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90. *Cité page 55*

Kuang, L., Xia, Y., and Mao, Y. (2012). Personalized Services Recommendation Based on Context-Aware QoS Prediction. In *2012 IEEE 19th International Conference on Web Services*, pages 400–406. IEEE. *Cité page 99*

Kuang, L., Yu, L., Huang, L., Wang, Y., Ma, P., Li, C., and Zhu, Y. (2018). A Personalized QoS Prediction Approach for CPS Service Recommendation Based on Reputation and Location-Aware Collaborative Filtering. *Sensors*, 18(5):1556. *2 citations pages 85 et 88*

Kurbel, K. E. (2008). *The Making of Information Systems: Software Engineering and Management in a Globalized World*. Springer Science & Business Media. *Cité page 9*

LeCompte, M. D. and Goetz, J. P. (1982). Problems of reliability and validity in ethnographic research. *Review of educational research*, 52(1):31–60. *Cité page 129*

Lecue, F. (2010). Combining Collaborative Filtering and Semantic Content-Based Approaches to Recommend Web Services. In *2010 IEEE Fourth International Conference on Semantic Computing*, pages 200–205. IEEE. *Cité page 70*

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. *Cité page 55*

LeCun, Y. et al. (1989). Generalization and network design strategies. *Connectionism in perspective*, 19:143–155. *Cité page 55*

Lee, H., Ekanadham, C., and Ng, A. Y. (2007). Sparse deep belief net model for visual area v2. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, page 873–880, Red Hook, NY, USA. Curran Associates Inc. *Cité page 60*

Li, J., Wu, H., Chen, J., He, Q., and Hsu, C.-H. (2021). Topology-Aware Neural Model for Highly Accurate QoS Prediction. *IEEE Transactions on Parallel and Distributed Systems*, X(X):1–1. *Cité page 93*

Li, M., Hua, Z., Zhao, J., Zou, Y., and Xie, B. (2012). ARIMA Model-Based Web Services Trustworthiness Evaluation and Prediction. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7636 LNCS, pages 648–655. Springer Berlin Heidelberg. *Cité page 78*

Li, M., Lu, Q., and Zhang, M. (2020). A Two-Tier Service Filtering Model for Web Service QoS Prediction. *IEEE Access*, 8:221278–221287. *2 citations pages 95 et 97*

Li, S., Kawale, J., and Fu, Y. (2015). Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, page 811–820, New York, NY, USA. Association for Computing Machinery. *Cité page 102*

Li, S., Wen, J., Luo, F., and Ranzi, G. (2018). Time-Aware QoS Prediction for Cloud Service Recommendation Based on Matrix Factorization. *IEEE Access*, 6:77716–77724. *Cité page 98*

Li, S., Wen, J., and Wang, X. (2019). From Reputation Perspective: A Hybrid Matrix Factorization for QoS Prediction in Location-Aware Mobile Service Recommendation System. *Mobile Information Systems*, 2019:1–12. *Cité page 97*

Liang, T., Chen, M., Yin, Y., Zhou, L., and Ying, H. (2021). Recurrent neural network based collaborative filtering for qos prediction in iov. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–11. *2 citations pages 77 et 97*

Lika, B., Kolomvatsos, K., and Hadjiefthymiades, S. (2014). Facing the cold start problem in recommender systems. *Expert Systems with Applications*, 41(4, Part 2):2065–2073. *2 citations pages 3 et 110*

Linden, G., Smith, B., and York, J. (2003). Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80. *Cité page 40*

Liu, J. and Chen, Y. (2019). A personalized clustering-based and reliable trust-aware QoS prediction approach for cloud service recommendation in cloud manufacturing. *Knowledge-Based Systems*, 174:43–56. *Cité page 97*

Liu, J., Tang, M., Zheng, Z., Liu, X., and Lyu, S. (2016). Location-aware and personalized collaborative filtering for web service recommendation. *IEEE Transactions on Services Computing*, 9(5):686–699. *Cité page 85*

Liu, L., Lecue, F., and Mehandjiev, N. (2013). Semantic content-based recommendation of software services using context. *ACM Transactions on the Web*, 7(3):1–20. *Cité page 69*

Liu, L.-M. and Hudak, G. (1992). *Forecasting and Time series analysis usng the SCA statical system*, volume 142. Scientific Computing Associates. *Cité page 73*

Liu, Y., Ngu, A. H., and Zeng, L. Z. (2004). Qos computation and policing in dynamic web service selection. In *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters*, WWW Alt. '04, page 66–73, New York, NY, USA. Association for Computing Machinery. *Cité page 19*

Lo, W., Yin, J., Deng, S., Li, Y., and Wu, Z. (2012a). An extended matrix factorization approach for QoS prediction in service selection. *Proceedings - 2012 IEEE 9th International Conference on Services Computing, SCC 2012*, pages 162–169. *Cité page 99*

Lo, W., Yin, J., Deng, S., Li, Y., and Wu, Z. (2012b). Collaborative web service qos prediction with location-based regularization. In *2012 IEEE 19th International Conference on Web Services*, pages 464–471. *Cité page 94*

Lo, W., Yin, J., Li, Y., and Wu, Z. (2015). Efficient web service QoS prediction using local neighborhood matrix factorization. *Engineering Applications of Artificial Intelligence*, 38:14–23. *Cité page 98*

Lousame, F. P. and Sánchez, E. (2009). *A Taxonomy of Collaborative-Based Recommender Systems*, volume 229, pages 81–117. Springer Berlin Heidelberg, Berlin, Heidelberg. *Cité page 33*

Luger, G. F. (2005). *Artificial intelligence: structures and strategies for complex problem solving*. Pearson education. *Cité page 46*

Luo, X., Liu, J., Zhang, D., and Chang, X. (2016a). A large-scale web QoS prediction scheme for the Industrial Internet of Things based on a kernel machine learning algorithm. *Computer Networks*, 101:81–89. *Cité page 98*

Luo, X., Wu, H., Yuan, H., and Zhou, M. (2019a). Temporal pattern-aware qos prediction via biased non-negative latent factorization of tensors. *IEEE Transactions on Cybernetics*, 50(5):1798–1809. *Cité page 93*

Luo, X., Xia, Y., and Zhu, Q. (2012). Incremental Collaborative Filtering recommender based on Regularized Matrix Factorization. *Knowledge-Based Systems*, 27:271–280. *Cité page 35*

Luo, X., Zhou, M., Wang, Z., Xia, Y., and Zhu, Q. (2019b). An Effective Scheme for QoS Estimation via Alternating Direction Method-Based Matrix Factorization. *IEEE Transactions on Services Computing*, 12(4):503–518. *Cité page 93*

Luo, X., Zhou, M., Wang, Z., Xia, Y., and Zhu, Q. (2019c). An effective scheme for qos estimation via alternating direction method-based matrix factorization. *IEEE Transactions on Services Computing*, 12(4):503–518. *Cité page 3*

Luo, X., Zhou, M., Xia, Y., Zhu, Q., Ammari, A. C., and Alabdulwahab, A. (2016b). Generating highly accurate predictions for missing qos data via aggregating nonnegative latent factor models. *IEEE Transactions on Neural Networks and Learning Systems*, 27(3):524–537. *Cité page 98*

Ma, H., King, I., and Lyu, M. R. (2007). Effective missing data prediction for collaborative filtering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '07*, page 39, New York, New York, USA. ACM Press. *Cité page 82*

Ma, Y., Wang, S., Hung, P. C., Hsu, C.-H., Sun, Q., and Yang, F. (2016). A Highly Accurate Prediction Algorithm for Unknown Web Service QoS Values. *IEEE Transactions on Services Computing*, 9(4):511–523. *2 citations pages 85 et 115*

Ma, Y., Wang, S., Yang, F., and Chang, R. N. (2015). Predicting QoS Values via Multi-dimensional QoS Data for Web Service Recommendations. In *2015 IEEE International Conference on Web Services*, pages 249–256. IEEE. *Cité page 75*

Mani, A. (2002). Understanding quality of service for web services. *http://www-106. ibm. com/developerworks/webservices/*. *Cité page 19*

Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., and Paul Smolley, S. (2017). Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802. *Cité page 125*

Margolis, B. (2007). *SOA for the business developer: Concepts, BPEL, and SCA (Business Developers Series)*. MC Press, LLC. *Cité page 18*

Marinescu, D. C. (2017). *Cloud computing: theory and practice*. Morgan Kaufmann. *Cité page 15*

Markey, P. and Clynch, G. (2013). A performance analysis of ws-*(soap) and restful web services for implementing service and resource orientated architectures. *The IT&T*, page 93. *Cité page 17*

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133. *Cité page 45*

McGill, M., Koll, M., and T., N. (1979). *An Evaluation of Factors Affecting Document Ranking By Information Retrieval Systems*. School of Information Studies, Syracuse University, Syracuse, N.Y. *Cité page 28*

Menasce, D. (2002). Qos issues in web services. *IEEE Internet Computing*, 6(6):72–75. *Cité page 20*

Mezni, H. and Fayala, M. (2018). Time-aware service recommendation: Taxonomy, review, and challenges. *Software: Practice and Experience*, 48(11):2080–2108. *3 citations pages 71, 78, et 99*

Mikolov, T. and Zweig, G. (2012). Context dependent recurrent neural network language model. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 234–239. *Cité page 56*

Mingdong Tang, Yechun Jiang, Jianxun Liu, and Xiaoqing Liu (2012). Location-Aware Collaborative Filtering for QoS-Based Service Recommendation. In *2012 IEEE 19th International Conference on Web Services*, pages 202–209. IEEE. *2 citations pages 71 et 86*

Minsky, M. and Papert, S. A. (2017). *Perceptrons: An introduction to computational geometry*. MIT press. *2 citations pages 45 et 48*

Mohamed, K. and Wijesekera, D. (2012). Performance analysis of web services on mobile devices. *Procedia Computer Science*, 10:744–751. *Cité page 17*

Mulligan, G. and Gračanin, D. (2009). A comparison of soap and rest implementations of a service based interaction independence middleware framework. In *Proceedings of the 2009 Winter Simulation Conference (WSC)*, pages 1423–1432. IEEE. *Cité page 17*

Mumbaikar, S., Padiya, P., et al. (2013). Web services based on soap and rest principles. *International Journal of Scientific and Research Publications*, 3(5):1–4. *Cité page 18*

Neal, R. M. (1992). Connectionist learning of belief networks. *Artificial Intelligence*, 56(1):71–113. *Cité page 49*

Ng, A. (2011). Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19. *Cité page 60*

Ngaffo, N., El Ayeb, W., and Choukair, Z. (2021). A time-aware service recommendation based on implicit trust relationships and enhanced user similarities. *Journal of Ambient Intelligence and Humanized Computing*, 12(2):3017–3035. *Cité page 85*

Nilsson, N. J. (2011). *The Quest for Artificial Intelligence*. Cambridge University Press, Cambridge. *Cité page 43*

Ning, X., Desrosiers, C., and Karypis, G. (2015). A Comprehensive Survey of Neighborhood-Based Recommendation Methods. In *Recommender Systems Handbook*, pages 37–76. Springer US, Boston, MA. *2 citations pages 26 et 28*

Niu, Z., Zhong, G., and Yu, H. (2021). A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62. *Cité page 95*

O'Mahony, M. P., Hurley, N. J., and Silvestre, G. C. (2005). Recommender systems: Attack types and strategies. In *Proceedings of the National Conference on Artificial Intelligence*, volume 1, pages 334–339. AAAI Press. *2 citations pages 28 et 82*

Pan, W., Xiang, E. W., Liu, N. N., and Yang, Q. (2010). Transfer learning in collaborative filtering for sparsity reduction. In *Proceedings of the National Conference on Artificial Intelligence*, volume 1 of *AAAI'10*, pages 230–235. AAAI Press. *2 citations pages 3 et 35*

Panchal, G., Ganatra, A., Kosta, Y., and Panchal, D. (2011). Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers. *International Journal of Computer Theory and Engineering*, 3(2):332–337. *Cité page 118*

Pandharbale, P. B., Mohanty, S. N., and Jagadev, A. K. (2021). Recent web service recommendation methods: A review. *Materials Today: Proceedings*. *Cité page 2*

Papazoglou, M. (2008). *Web services: principles and technology.* Pearson Education. *4 citations pages 11, 13, 14, et 15*

Peerzade, S. S. (2017). Web service recommendation using PCC based collaborative filtering. In *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, pages 2920–2924. IEEE. *Cité page 79*

Pennock, D. M., Horvitz, E. J., Lawrence, S., and Giles, C. L. (2013). Collaborative Filtering by Personality Diagnosis: A Hybrid Memory- and Model-Based Approach. *arXiv preprint arXiv:1301.3885*, pages 473–480. *Cité page 27*

Platenius, M. C., von Detten, M., Becker, S., Schäfer, W., and Engels, G. (2013). A survey of fuzzy service matching approaches in the context of on-the-fly computing. In *Proceedings of the 16th International ACM Sigsoft Symposium on Component-Based Software Engineering*, CBSE '13, page 143–152, New York, NY, USA. Association for Computing Machinery. *Cité page 20*

Qi, L., Dou, W., and Zhang, X. (2017). An inverse collaborative filtering approach for cold-start problem in web service recommendation. In *Proceedings of the Australasian Computer Science Week Multiconference*, pages 1–9, New York, NY, USA. ACM. *2 citations pages 85 et 115*

Qi, L., Wang, R., Hu, C., Li, S., He, Q., and Xu, X. (2019). Time-aware distributed service recommendation with privacy-preservation. *Information Sciences*, 480:354–364. *Cité page 85*

Qiu, W., Zheng, Z., Wang, X., Yang, X., and Lyu, M. R. (2013). Reputation-Aware QoS Value Prediction of Web Services. In *2013 IEEE International Conference on Services Computing*, pages 41–48. IEEE. *Cité page 86*

Qu, Y., Fang, B., Zhang, W., Tang, R., Niu, M., Guo, H., Yu, Y., and He, X. (2018). Product-based neural networks for user response prediction over multi-field categorical data. *ACM Trans. Inf. Syst.*, 37(1). *Cité page 2*

Ramachandran, P., Zoph, B., and Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*. *Cité page 50*

Ran, S. (2003). A model for web services discovery with QoS. *ACM SIGecom Exchanges*, 4(1):1–10. *2 citations pages 19 et 67*

Ranzato, M. A., Boureau, Y.-L., and LeCun, Y. (2007). Sparse feature learning for deep belief networks. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, page 1185–1192, Red Hook, NY, USA. Curran Associates Inc. *Cité page 60*

Refaeilzadeh, P., Tang, L., and Liu, H. (2016). *Cross-Validation*, pages 1–7. Springer New York, New York, NY. *2 citations pages 107 et 122*

Ren, L. and Wang, W. (2018). An svm-based collaborative filtering approach for top-n web services recommendation. *Future Generation Computer Systems*, 78:531–543. *Cité page 98*

Rendle, S. (2010). Factorization machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000. *Cité page 95*

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work - CSCW '94*, pages 175–186, New York, New York, USA. *4 citations pages 26, 28, 31, et 82*

Resnick, P. and Varian, H. R. (1997). Recommender systems. *Communications of the ACM*, 40(3):56–58. *2 citations pages 2 et 23*

Ricci, F., Rokach, L., and Shapira, B. (2011). *Introduction to Recommender Systems Handbook*, pages 1–35. Springer US, Boston, MA. *2 citations pages 25 et 27*

Rich, E. (1979). User modeling via stereotypes. *Cognitive Science*, 3(4):329–354. *Cité page 24*

Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 833–840, Madison, WI, USA. Omnipress. *Cité page 61*

Rosen, M., Lublinsky, B., Smith, K. T., and Balcer, M. J. (2012). *Applied SOA: service-oriented architecture and design strategies*. John Wiley & Sons. *2 citations pages 9 et 10*

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386. *2 citations pages 45 et 46*

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536. *2 citations pages 52 et 56*

Ryu, D., Lee, K., and Baik, J. (2018). Location-Based Web Service QoS Prediction via Preference Propagation to Address Cold Start Problem. *IEEE Transactions on Services Computing*, 14(3):736–746. *Cité page 97*

Sahu, P., Raghavan, S., Chandrasekaran, K., and Usha, D. (2021). Time-Aware Online QoS Prediction Using LSTM and Non-negative Matrix Factorization. In Sheth, A., Sinhal, A., Shrivastava, A., and Pandey, A. K., editors, *Intelligent Systems*, pages 369–376. Springer Singapore, Singapore. *2 citations pages 75 et 97*

Salakhutdinov, R. and Mnih, A. (2008). Bayesian probabilistic matrix factorization using markov chain Monte Carlo. In *Proceedings of the 25th International Conference on Machine Learning*, pages 880–887, New York, New York, USA. ACM Press. *Cité page 40*

Salakhutdinov, R. and Mnih, A. (2009). Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems 20 - Proceedings of the 2007 Conference*, pages 1257–1264. *3 citations pages 35, 39, et 89*

Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, 3(3):210–229. *Cité page 44*

Saranya, K. G., Sudha Sadasivam, G., and Chandralekha, M. (2016). Performance comparison of different similarity measures for collaborative filtering technique. *Indian Journal of Science and Technology*, 9(29). *Cité page 85*

Sarwar, B., Karypis, G., Konstan, J., and Reidl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the tenth international conference on World Wide Web - WWW '01*, volume 1, pages 285–295, New York, New York, USA. ACM Press. *5 citations pages 32, 33, 40, 81, et 126*

Schafer, J. B., Frankowski, D., Herlocker, J., and Sen, S. (2007). *Collaborative Filtering Recommender Systems*, volume 4321, pages 291–324. Springer Berlin Heidelberg, Berlin, Heidelberg. *6 citations pages 28, 31, 33, 35, 40, et 82*

Schein, A. I., Popescul, A., Ungar, L. H., and Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, page 253–260, New York, NY, USA. Association for Computing Machinery. *2 citations pages 35 et 110*

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117. *Cité page 57*

Serrano, D., Bouchenak, S., Kouki, Y., de Oliveira Jr., F. A., Ledoux, T., Lejeune, J., Sopena, J., Arantes, L., and Sens, P. (2016). Sla guarantees for cloud services. *Future Generation Computer Systems*, 54:233–246. *Cité page 18*

Shani, G. and Gunawardana, A. (2011). *Evaluating Recommendation Systems*, pages 257–297. Springer US, Boston, MA. *Cité page 41*

Shao, L., Zhang, J., Wei, Y., Zhao, J., Xie, B., and Mei, H. (2007). Personalized QoS Prediction for Web Services via Collaborative Filtering. In *IEEE International Conference on Web Services (ICWS 2007)*, pages 439–446. *4 citations pages 2, 80, 86, et 126*

Shrivastava, P. and Sharma, D. (2021). Web based recommendation system using Multi-attribute collaborative filtering for user satisfaction. In *2021 International Conference on Innovative Practices in Technology and Management (ICIPTM)*, pages 180–185. IEEE. *2 citations pages 85 et 115*

Shum, S. B. et al. (2008). Cohere: Towards web 2.0 argumentation. *COMMA*, 8:97–108. *Cité page 1*

Singh, V. P., Pandey, M. K., Singh, P. S., and Karthikeyan, S. (2020). Neural net time series forecasting framework for time-aware web services recommendation. *Procedia Computer Science*, 171:1313–1322. Third International Conference on Computing and Network Communications (CoCoNet'19). *Cité page 93*

Smahi, M. I., Hadjila, F., Tibermacine, C., and Benamar, A. (2021). A deep learning approach for collaborative prediction of Web service QoS. *Service Oriented Computing and Applications*, 15(1):5–20. *3 citations pages 4, 105, et 132*

Smahi, M. I., Hadjila, F., Tibermacine, C., Merzoug, M., and Benamar, A. (2018). An encoder-decoder architecture for the prediction of web service QoS. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11116 LNCS, pages 74–89. *4 citations pages 4, 63, 105, et 132*

Smirnova, E. and Vasile, F. (2017). Contextual sequence modeling for recommendation with recurrent neural networks. In *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems*, DLRS 2017, page 2–9, New York, NY, USA. Association for Computing Machinery. *Cité page 3*

Smithamol, M. B. and Rajeswari, S. (2019). TMM: Trust Management Middleware for Cloud Service Selection by Prioritization. *Journal of Network and Systems Management*, 27(1):66–92. *Cité page 70*

Song, Y. (2020). Collaborative prediction of web service quality based on user preferences and services. *PloS one*, 15(12):e0242089. *Cité page 85*

Strub, F., Gaudel, R., and Mary, J. (2016). Hybrid recommender system based on autoencoders. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, DLRS 2016, page 11–16, New York, NY, USA. Association for Computing Machinery. *Cité page 3*

Su, K., Ma, L., Xiao, B., and Zhang, H. (2016). Web service QoS prediction by neighbor information combined non-negative matrix factorization. *Journal of Intelligent & Fuzzy Systems*, 30(6):3593–3604. *Cité page 98*

Su, K., Xiao, B., Liu, B., Zhang, H., and Zhang, Z. (2017). TAP: A personalized trust-aware QoS prediction approach for web service recommendation. *Knowledge-Based Systems*, 115:55–65. *3 citations pages 70, 95, et 98*

Su, X. and Khoshgoftaar, T. M. (2009). A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence*, 2009(Section 3):1–19. *2 citations pages 26 et 71*

Su, X., Zhang, M., Liang, Y., Cai, Z., Guo, L., and Ding, Z. (2021). A tensor-based approach for the qos evaluation in service-oriented environments. *IEEE Transactions on Network and Service Management*, 18(3):3843–3857. *Cité page 93*

Sumra, R. and Arulazi, D. (2003). Quality of service for web services-demystification, limitations, and best practices. *Retrieved February*, 10:2006. *Cité page 20*

Sun, H., Zheng, Z., Chen, J., and Lyu, M. R. (2013). Personalized Web Service Recommendation via Normal Recovery Collaborative Filtering. *IEEE Transactions on Services Computing*, 6(4):573–579. *2 citations pages 79 et 86*

Sun, Y., Wang, X., and Tang, X. (2015). Deeply learned face representations are sparse, selective, and robust. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2892–2900. *2 citations pages 49 et 119*

Sutskever, I., Martens, J., and Hinton, G. (2011). Generating text with recurrent neural networks. *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pages 1017–1024. *Cité page 75*

Syu, Y., Kuo, J.-Y., and Fanjiang, Y.-Y. (2017). Time series forecasting for dynamic quality of web services: An empirical study. *Journal of Systems and Software*, 134:279–303. *3 citations pages 73, 78, et 99*

Syu, Y. and Wang, C.-M. (2021). QoS Time Series Modeling and Forecasting for Web Services: A Comprehensive Survey. *IEEE Transactions on Network and Service Management*, 18(1):926–944. *2 citations pages 78 et 99*

Syu, Y., Wang, C. M., and Fanjiang, Y. Y. (2018). *A Survey of Time-Aware Dynamic QoS Forecasting Research, Its Future Challenges and Research Directions*, volume 10969 of *Lecture Notes in Computer Science*. Springer International Publishing, Cham. *2 citations pages 78 et 99*

Tang, M., Zhang, T., Liu, J., and Chen, J. (2015). Cloud service QoS prediction via exploiting collaborative filtering and location-based data smoothing. *Concurrency and Computation: Practice and Experience*, 27(18):5826–5839. *Cité page 86*

Tang, M., Zheng, Z., Kang, G., Liu, J., Yang, Y., and Zhang, T. (2016). Collaborative web service quality prediction via exploiting matrix factorization and network map. *IEEE Transactions on Network and Service Management*, 13(1):126–137. *2 citations pages 94 et 103*

Tao, Q., Chang, H.-y., Gu, C.-q., and Yi, Y. (2012). A novel prediction approach for trustworthy QoS of web services. *Expert Systems with Applications*, 39(3):3676–3681. *Cité page 99*

Thinh, L.-V. and Tu, T.-D. (2017). QoS prediction for web services based on user-trust propagation model. *New Review of Hypermedia and Multimedia*, 23(4):277–291. *Cité page 94*

Tian, G., Wang, J., He, K., Sun, C., and Tian, Y. (2017). Integrating implicit feedbacks for time-aware web service recommendations. *Information Systems Frontiers*, 19(1):75–89. *Cité page 94*

Valdiviezo-Diaz, P., Ortega, F., Cobos, E., and Lara-Cabrera, R. (2019). A Collaborative Filtering Approach Based on Naïve Bayes Classifier. *IEEE Access*, 7:108581–108592. *Cité page 39*

Van Gerven, M. and Bohte, S. (2017). Artificial neural networks as models of neural information processing. *Frontiers in Computational Neuroscience*, 11:114. *Cité page 50*

Velte, A. T., Velte, T. J., and Elsenpeter, R. (2010). Cloud computing. *A practical approach*, pages 135–140. *Cité page 18*

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning - ICML '08*, pages 1096–1103, New York, New York, USA. ACM Press. *3 citations pages 4, 57, et 103*

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A., and Bottou, L. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12):3371–3408. *2 citations pages 59 et 91*

Vinoski, S. (2007). Rest eye for the soa guy. *IEEE Internet Computing*, 11(1):82–84. *Cité page 15*

Viriyasitavat, W., Xu, L. D., and Martin, A. (2012). Swspec: The requirements specification language in service workflow environments. *IEEE Transactions on Industrial Informatics*, 8(3):631–638. *Cité page 18*

Vogel, A., Kerherve, B., von Bochmann, G., and Gecsei, J. (1995). Distributed multimedia and qos: a survey. *IEEE MultiMedia*, 2(2):10–19. *Cité page 18*

Wacheux, F. (1996). Méthodes qualitatives et recherches en gestion, éditions economica. *Gestion, Paris*. *Cité page 129*

Wagner, N., Michalewicz, Z., Khouja, M., and McGregor, R. R. (2007). Time Series Forecasting for Dynamic Environments: The DyFor Genetic Program Model. *IEEE Transactions on Evolutionary Computation*, 11(4):433–452. *2 citations pages 73 et 77*

Wang, F.-f., Chen, F.-z., and Li, M.-q. (2019a). A Collaborative Filtering Method for Cloud Service Recommendation via Exploring Usage History. In *Proceeding of the 24th International Conference on Industrial Engineering and Engineering Management 2018*, volume 1, pages 716–725. Springer Singapore, Singapore. *Cité page 85*

Wang, H., Wang, J., Wang, J., Zhao, M., Zhang, W., Zhang, F., Xie, X., and Guo, M. (2018). Graphgan: Graph representation learning with generative adversarial nets. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1). *Cité page 109*

Wang, J., Yu, L., Zhang, W., Gong, Y., Xu, Y., Wang, B., Zhang, P., and Zhang, D. (2017a). Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, page 515–524, New York, NY, USA. Association for Computing Machinery. *Cité page 109*

Wang, Q., Chen, M., Shang, M., and Luo, X. (2019b). A momentum-incorporated latent factorization of tensors model for temporal-aware QoS missing data prediction. *Neurocomputing*, 367:299–307. *Cité page 97*

Wang, S., Zheng, Z., Wu, Z., Lyu, M. R., and Yang, F. (2015). Reputation Measurement and Malicious Feedback Rating Prevention in Web Service Recommendation Systems. *IEEE Transactions on Services Computing*, 8(5):755–767. *Cité page 86*

Wang, X., Xu, Z., Xia, X., and Mao, C. (2017b). Computing User Similarity by Combining SimRank++ and Cosine Similarities to Improve Collaborative Filtering. In *2017 14th Web Information Systems and Applications Conference (WISA)*, volume 2018-Janua, pages 205–210. IEEE. *Cité page 79*

Wang, X., Zhu, J., Zheng, Z., Song, W., Shen, Y., and Lyu, M. R. (2016). A Spatial-Temporal QoS Prediction Approach for Time-aware Web Service Recommendation. *ACM Transactions on the Web*, 10(1):1–25. *4 citations pages 2, 74, 78, et 126*

Welch, G. and Bishop, G. (1995). An Introduction to the Kalman Filter. Technical report, USA. *Cité page 74*

Wenqiang Li, Hongji Xu, Mingyang Ji, Zhengzheng Xu, and Haiteng Fang (2016). A hierarchy weighting similarity measure to improve user-based collaborative filtering algorithm. In *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, pages 843–846. IEEE. *Cité page 85*

Werbos, P. (1974). Beyond regression:" new tools for prediction and analysis in the behavioral sciences. *Ph. D. dissertation, Harvard University.* *Cité page 52*

Werbos, P. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560. *Cité page 45*

White, G. (2020). *Monitoring & Predicting QoS in IoT Services Declaration of Authorship.* PhD thesis, Trinity College Dublin.School of Computer Science & Statistics. *Cité page 75*

White, G. and Clarke, S. (2020). Short-term qos forecasting at the edge for reliable service applications. *IEEE Transactions on Services Computing*, pages 1–1. *Cité page 91*

White, G., Palade, A., Cabrera, C., and Clarke, S. (2018a). IoTPredict: Collaborative QoS Prediction in IoT. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10. IEEE. *Cité page 85*

White, G., Palade, A., Cabrera, C., and Clarke, S. (2019). Autoencoders for qos prediction at the edge. In *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom*, pages 1–9. *3 citations pages 3, 91, et 93*

White, G., Palade, A., and Clarke, S. (2018b). Forecasting qos attributes using lstm networks. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. *Cité page 78*

Widrow, B. and Hoff, M. E. (1960). Adaptive switching circuits. Technical report, Stanford Univ Ca Stanford Electronics Labs. *Cité page 46*

Wiegers, K. and Beatty, J. (2013). *Software requirements.* Pearson Education. *Cité page 19*

Wittek, P., Gao, S., Lim, I., and Zhao, L. (2017). somoclu: An efficient parallel library for self-organizing maps. *Journal of Statistical Software, Articles*, 78(9):1–21. *Cité page 64*

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14*, pages 1–10, New York, New York, USA. ACM Press. *Cité page 68*

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering.* Springer Science & Business Media. *Cité page 129*

Wu, C., Qiu, W., Wang, X., Zheng, Z., and Yang, X. (2016). Time-Aware and Sparsity-Tolerant QoS Prediction Based on Collaborative Filtering. In *2016 IEEE International Conference on Web Services (ICWS)*, pages 637–640. IEEE. *Cité page 98*

Wu, C., Qiu, W., Zheng, Z., Wang, X., and Yang, X. (2015). QoS Prediction of Web Services Based on Two-Phase K-Means Clustering. In *2015 IEEE International Conference on Web Services*, pages 161–168. IEEE. *2 citations pages 80 et 98*

Wu, D. and Luo, X. (2020). Robust latent factor analysis for precise representation of high-dimensional and sparse data. *IEEE/CAA Journal of Automatica Sinica*, 8(4):796–805. *Cité page 93*

Wu, D., Luo, X., Shang, M., He, Y., Wang, G., and Wu, X. (2019a). A data-aware latent factor model for web service qos prediction. In Yang, Q., Zhou, Z.-H., Gong, Z., Zhang, M.-L., and Huang, S.-J., editors, *Advances in Knowledge Discovery and Data Mining*, pages 384–399, Cham. Springer International Publishing. *Cité page 97*

Wu, H., Yue, K., Hsu, C.-H., Zhao, Y., Zhang, B., and Zhang, G. (2017a). Deviation-based neighborhood model for context-aware QoS prediction of cloud and IoT services. *Future Generation Computer Systems*, 76:550–560. *2 citations pages 79 et 98*

Wu, H., Yue, K., Li, B., Zhang, B., and Hsu, C.-H. (2018a). Collaborative qos prediction with context-sensitive matrix factorization. *Future Generation Computer Systems*, 82:669–678. *2 citations pages 89 et 93*

Wu, H., Zhang, Z., Luo, J., Yue, K., and Hsu, C.-H. (2018b). Multiple attributes qos prediction via deep neural model with contexts*. *IEEE Transactions on Services Computing*, 14(4):1084–1096. *Cité page 93*

Wu, J., Chen, L., Feng, Y., Zheng, Z., Zhou, M. C., and Wu, Z. (2013). Predicting Quality of Service for Selection by Neighborhood-Based Collaborative Filtering. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(2):428–439. *Cité page 99*

Wu, X., Cheng, B., and Chen, J. (2017b). Collaborative Filtering Service Recommendation Based on a Novel Similarity Computation Method. *IEEE Transactions on Services Computing*, 10(3):352–365. *2 citations pages 82 et 85*

Wu, X., Fan, Y., Zhang, J., Lin, H., and Zhang, J. (2019b). Qf-rnn: Qi-matrix factorization based rnn for time-aware service recommendation. In *2019 IEEE International Conference on Services Computing (SCC)*, pages 202–209. *3 citations pages 75, 76, et 97*

Wu, Y., Xie, F., Chen, L., Chen, C., and Zheng, Z. (2017c). An embedding based factorization machine approach for web service qos prediction. In Maximilien, M., Vallecillo, A., Wang, J., and Oriol, M., editors, *Service-Oriented Computing*, pages 272–286, Cham. Springer International Publishing. *2 citations pages 3 et 98*

Xie, Q., Zhao, S., Zheng, Z., Zhu, J., and Lyu, M. R. (2016). Asymmetric correlation regularized matrix factorization for web service recommendation. In *2016 IEEE International Conference on Web Services (ICWS)*, pages 204–211. *Cité page 94*

Xiong, R., Wang, J., Li, Z., Li, B., and Hung, P. C. K. (2018a). Personalized LSTM Based Matrix Factorization for Online QoS Prediction. In *2018 IEEE International Conference on Web Services (ICWS)*, volume 63, pages 34–41. IEEE. *2 citations pages 76 et 98*

Xiong, R., Wang, J., Zhang, N., and Ma, Y. (2018b). Deep hybrid collaborative filtering for web service recommendation. *Expert Systems with Applications*, 110:191–205. *Cité page 93*

Xiong, W., Wu, Z., Li, B., and Gu, Q. (2017). A learning approach to qos prediction via multi-dimensional context. In *2017 IEEE International Conference on Web Services (ICWS)*, pages 164–171. *Cité page 98*

Xu, B., Wang, N., Chen, T., and Li, M. (2015). Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*. *Cité page 119*

Xu, J., Zheng, Z., and Lyu, M. R. (2016a). Web service personalized quality of service prediction via reputation-based matrix factorization. *IEEE Transactions on Reliability*, 65(1):28–37. *3 citations pages 71, 88, et 94*

Xu, X., Liu, Q., Zhang, X., Zhang, J., Qi, L., and Dou, W. (2019). A Blockchain-Powered Crowdsourcing Method With Privacy Preservation in Mobile Environment. *IEEE Transactions on Computational Social Systems*, 6(6):1407–1419. *Cité page 115*

Xu, Y., Yin, J., Deng, S., N. Xiong, N., and Huang, J. (2016b). Context-aware qos prediction for web service recommendation and selection. *Expert Systems with Applications*, 53:75–86. *2 citations pages 89 et 94*

Xu, Y., Yin, J., and Lo, W. (2013a). A Unified Framework of QoS-Based Web Service Recommendation with Neighborhood-Extended Matrix Factorization. In *2013 IEEE 6th International Conference on Service-Oriented Computing and Applications*, volume 1, pages 198–205. IEEE. *2 citations pages 88 et 99*

Xu, Y., Yin, J., Lo, W., and Wu, Z. (2013b). Personalized location-aware qos prediction for web services using probabilistic matrix factorization. In Lin, X., Manolopoulos, Y., Srivastava, D., and Huang, G., editors, *Web Information Systems Engineering – WISE 2013*, pages 229–242, Berlin, Heidelberg. Springer Berlin Heidelberg. *Cité page 94*

Yan, C., Zhang, Y., Zhong, W., Zhang, C., and Xin, B. (2022). A truncated SVD-based ARIMA model for multiple QoS prediction in mobile edge computing. *Tsinghua Science and Technology*, 27(2):315–324. *Cité page 78*

Yang, Y., Zheng, Z., Niu, X., Tang, M., Lu, Y., and Liao, X. (2018). A location-based factorization machine model for web service qos prediction. *IEEE Transactions on Services Computing*, 14(5):1264–1277. *Cité page 93*

Yao, L., Sheng, Q. Z., Ngu, A. H., Yu, J., and Segev, A. (2015). Unified Collaborative and Content-Based Web Service Recommendation. *IEEE Transactions on Services Computing*, 8(3):453–466. *2 citations pages 69 et 70*

Yao, L., Wang, X., Sheng, Q. Z., Benatallah, B., and Huang, C. (2018). Mashup recommendation by regularizing matrix factorization with api co-invocations. *IEEE Transactions on Services Computing*, 14(2):502–515. *2 citations pages 89 et 93*

Yin, J., Lo, W., Deng, S., Li, Y., Wu, Z., and Xiong, N. (2014). Colbar: A collaborative location-based regularization framework for QoS prediction. *Information Sciences*, 265:68–84. *Cité page 99*

Yin, Y., Cao, Z., Xu, Y., Gao, H., Li, R., and Mai, Z. (2020). QoS Prediction for Service Recommendation With Deep Features Learning in Mobile Edge Computing Environment. *IEEE Transactions on Cognitive Communications and Networking*, 6(4):1136–1145. *4 citations pages 3, 90, 91, et 97*

Yin, Y., Zhang, W., Xu, Y., Zhang, H., Mai, Z., and Yu, L. (2019). Qos prediction for mobile edge service recommendation with auto-encoder. *IEEE Access*, 7:62312–62324. *Cité page 97*

Youssef, A., Delpha, C., and Diallo, D. (2016). An optimal fault detection threshold for early detection using Kullback–Leibler Divergence for unknown distribution data. *Signal Processing*, 120:266–279. *2 citations pages 61 et 62*

Yu, C. and Huang, L. (2016). A Web service QoS prediction approach based on time- and location-aware collaborative filtering. *Service Oriented Computing and Applications*, 10(2):135–149. *2 citations pages 70 et 98*

Yu, D., Liu, Y., Xu, Y., and Yin, Y. (2014). Personalized QoS Prediction for Web Services Using Latent Factor Models. In *2014 IEEE International Conference on Services Computing*, pages 107–114. IEEE. *Cité page 99*

Yu, T., Zhang, Y., and Lin, K.-J. (2007). Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Trans. Web*, 1(1):6–es. *Cité page 2*

Yuan, W., Li, C., Guan, D., Han, G., and Khattak, A. M. (2018). Socialized healthcare service recommendation using deep learning. *Neural Computing and Applications*, 30(7):2071–2082. *Cité page 98*

Yuan, Y., Shang, M., and Luo, X. (2020). Temporal web service qos prediction via kalman filter-incorporated latent factor analysis. In Giacomo, G. D., Catalá, A., Dilkina, B., Milano, M., Barro, S., Bugarín, A., and Lang, J., editors, *Frontiers in Artificial Intelligence and Applications*, volume 325, pages 561–568. IOS Press. *Cité page 97*

Zeng, L., Lei, H., and Chang, H. (2007). Monitoring the qos for web services. In Krämer, B. J., Lin, K.-J., and Narasimhan, P., editors, *Service-Oriented Computing – ICSOC 2007*, pages 132–144, Berlin, Heidelberg. Springer Berlin Heidelberg. *Cité page 19*

Zhang, G., Liu, Y., and Jin, X. (2020). A survey of autoencoder-based recommender systems. *Frontiers of Computer Science*, 14(2):430–450. *Cité page 102*

Zhang, H., Zhang, X., and Cao, Z. (2018). Research on GRU-based credibility prediction of Web services. *MATEC Web of Conferences*, 232:01049. *Cité page 78*

Zhang, L.-J., Zhang, J., and Cai, H. (2007). *Services computing*. Springer Berlin Heidelberg. *Cité page 10*

Zhang, M., Liu, X., Zhang, R., and Sun, H. (2012). A web service recommendation approach based on QoS prediction using fuzzy clustering. *Proceedings - 2012 IEEE 9th International Conference on Services Computing, SCC 2012*, pages 138–145. *Cité page 99*

Zhang, P., Jin, H., Dong, H., Song, W., and Wang, L. (2019a). La-lmrbf: Online and long-term web service qos forecasting. *IEEE Transactions on Services Computing*, pages 1–1. *Cité page 97*

Zhang, P., Wang, L., Li, W., Leung, H., and Song, W. (2017). A Web Service QoS Forecasting Approach Based on Multivariate Time Series. In *2017 IEEE International Conference on Web Services (ICWS)*, pages 146–153. IEEE. *Cité page 78*

Zhang, Y., Pan, J., Qi, L., and He, Q. (2021a). Privacy-preserving quality prediction for edge-based IoT services. *Future Generation Computer Systems*, 114:336–348. *2 citations pages 85 et 115*

Zhang, Y., Wang, K., He, Q., Chen, F., Deng, S., Zheng, Z., and Yang, Y. (2021b). Covering-Based Web Service Quality Prediction via Neighborhood-Aware Matrix Factorization. *IEEE Transactions on Services Computing*, 14(5):1333–1344. *Cité page 97*

Zhang, Y., Yin, C., Lu, Z., Yan, D., Qiu, M., and Tang, Q. (2019b). Recurrent Tensor Factorization for time-aware service recommendation. *Applied Soft Computing*, 85:105762. *2 citations pages 75 et 97*

Zhang, Y., Yin, C., Wu, Q., He, Q., and Zhu, H. (2019c). Location-aware deep collaborative filtering for service recommendation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 51(6):3796–3807. *3 citations pages 91, 93, et 126*

Zhang, Y., Zheng, Z., and Lyu, M. R. (2011a). Exploring Latent Features for Memory-Based QoS Prediction in Cloud Computing. In *2011 IEEE 30th International Symposium on Reliable Distributed Systems*, pages 1–10. IEEE. *2 citations pages 2 et 99*

Zhang, Y., Zheng, Z., and Lyu, M. R. (2011b). WSPred: A time-aware personalized QoS prediction framework for Web services. In *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, pages 210–219. IEEE. *2 citations pages 94 et 115*

Zheng, Z. and Lyu, M. R. (2010). Collaborative reliability prediction of service-oriented systems. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - ICSE '10*, volume 1, page 35, New York, New York, USA. ACM Press. *Cité page 115*

Zheng, Z., Ma, H., Lyu, M. R., and King, I. (2009). WSRec: A Collaborative Filtering Based Web Service Recommender System. In *2009 IEEE International Conference on Web Services*, pages 437–444. IEEE. *2 citations pages 83 et 86*

Zheng, Z., Ma, H., Lyu, M. R., and King, I. (2011). QoS-Aware Web Service Recommendation by Collaborative Filtering. *IEEE Transactions on Services Computing*, 4(2):140–152. *Cité page 83*

Zheng, Z., Ma, H., Lyu, M. R., and King, I. (2013a). Collaborative Web Service QoS Prediction via Neighborhood Integrated Matrix Factorization. *IEEE Transactions on Services Computing*, 6(3):289–299. *Cité page 81*

Zheng, Z., Wu, X., Zhang, Y., Lyu, M. R., and Wang, J. (2013b). QoS Ranking Prediction for Cloud Services. *IEEE Transactions on Parallel and Distributed Systems*, 24(6):1213–1222. *Cité page 85*

Zheng, Z., Xiaoli, L., Tang, M., Xie, F., and Lyu, M. R. (2020). Web Service QoS Prediction via Collaborative Filtering: A Survey. *IEEE Transactions on Services Computing*, 1374(c):1–1. *4 citations pages 72, 86, 94, et 99*

Zheng, Z., Zhang, Y., and Lyu, M. R. (2010). Distributed QoS Evaluation for Real-World Web Services. In *2010 IEEE International Conference on Web Services*, pages 83–90. IEEE. *Cité page 115*

Zheng, Z., Zhang, Y., and Lyu, M. R. (2014). Investigating qos of real-world web services. *IEEE Transactions on Services Computing*, 7(1):32–39. *2 citations pages 2 et 116*

Zhou, J., Guo, X., and Yin, C. (2020). Recurrent factorization machine with self-attention for time-aware service recommendation. In *2020 6th International Conference on Big Data Computing and Communications (BIGCOM)*, pages 189–197. *Cité page 97*

Zhou, Q., Wu, H., Yue, K., and Hsu, C.-H. (2019). Spatio-temporal context-aware collaborative qos prediction. *Future Generation Computer Systems*, 100:46–57. *Cité page 93*

Zhu, J., He, P., Zheng, Z., and Lyu, M. R. (2017). Online qos prediction for runtime service adaptation via adaptive matrix factorization. *IEEE Transactions on Parallel and Distributed Systems*, 28(10):2911–2924. *Cité page 93*

Zhu, J., Kang, Y., Zheng, Z., and Lyu, M. R. (2012). A Clustering-Based QoS Prediction Approach for Web Service Recommendation. In *2012 IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, pages 93–98. IEEE. *Cité page 99*

Zhu, X., Jing, X.-Y., Wu, D., He, Z., Cao, J., Yue, D., and Wang, L. (2021). Similarity-Maintaining Privacy Preservation and Location-Aware Low-Rank Matrix Factorization for QoS Prediction Based Web Service Recommendation. *IEEE Transactions on Services Computing*, 14(3):889–902. *Cité page 97*

Zigoris, P. and Zhang, Y. (2006). Bayesian adaptive user profiling with explicit & implicit feedback. In *Proceedings of the 15th ACM international conference on Information and knowledge management - CIKM '06*, page 397, New York, New York, USA. ACM Press. *Cité page 39*

Zou, G., Jiang, M., Niu, S., Wu, H., Pang, S., and Gan, Y. (2018). QoS-Aware Web Service Recommendation with Reinforced Collaborative Filtering. In Pahl, C., Vukovic, M., Yin, J., and Yu, Q., editors, *Service-Oriented Computing*, volume 11236 of *Lecture Notes in Computer Science*, pages 430–445. Springer International Publishing, Cham. *Cité page 85*

Résumé

Dans cette thèse, nous proposons une approche de prédiction de QoS fondée sur le modèle d'apprentissage profond combinée avec de la factorisation matricielle, où cette dernière est basée, à la fois, sur une architecture des auto-encodeurs et une technique de classification. Trois variants du modèle d'auto-encodeur ont été utilisé. Le premier est composé d'une seule couche cachée représentant le vecteur des facteurs latents des utilisateurs et/ou services. Une deuxième architecture considère plusieurs couches cachées. Un troisième modèle consiste en une combinaison entre l'architecture d'auto-encodeur profond avec celle d'un réseau antagoniste génératif. D'autres problèmes sous-jacents à l'estimation des valeurs manquantes de QoS ont été traités dans ce travail. Le premier est relatif à la vulnérabilité des systèmes de prédiction face au problème de la rareté de données. Notre proposition pour surmonter ce dernier consiste en une utilisation d'un algorithme de partitionnement basé sur les cartes auto-organisatrices de Kohonen, où l'initialisation est faite à la base des attributs de localisation. Le second problème traité est le démarrage à froid, survenu lors de l'ajout de nouveaux utilisateurs/services au système de prédiction. Ce dernier est globalement maîtrisé en exploitant également les caractéristiques spatiales. Les différentes évaluations empiriques que nous avons menées, montrent que nos propositions de solutions sont en mesure de donner de meilleures performances en matière de prédiction des valeurs de QoS manquantes, et fournissent ainsi de meilleures orientations aux utilisateurs dans leurs choix de services préférés que les méthodes existantes. Les paramètres, en termes de QoS, sur lesquels nous nous sommes appuyés pour réaliser ces différentes expériences sont le temps de réponse et le débit. Cependant, les algorithmes de prédiction de QoS, proposés dans cette thèse, peuvent être appliqués aisément à d'autres facteurs de QoS.

Mots-clés: Prédiction de QoS, Service Web, Filtrage collaboratif, Apprentissage profond, Auto-encodeur, Réseau antagoniste génératif, Carte auto-organisatrice.

Abstract

In this thesis, we propose a deep learning-based approach wich combines a matrix factorization model based on a deep auto-encoder (DAE) and a clustering technique. Three variants of the auto-encoder design have been used. The first one is composed of a single hidden layer that represents the vector of latent factors of users and/or services. A second architecture considers several hidden layers. A third model consists of a combination of a deep auto-encoder model and a generative adversarial network. Other problems underlying the estimation of missing QoS values were addressed in this work. The first one is related to the vulnerability of prediction systems to the data sparsity problem. To deal with this issue our proposal consists of in using a clustering algorithm based on Kohonen's self-organising maps, where the initialization is done using location attributes. The second one that we have dealt with is the cold start problem, which occurs when adding new users/services to the prediction system. The latter one is globally managed by exploiting a spatial features as well. The conducted experiments show that our proposals can provide better performances in terms of QoS prediction, and consequently provide more guidance for users in their choice of preferred services than existing methods do. The QoS parameters on which we relied on to carry out our various experiments are response time and throughput. However, the proposed QoS prediction algorithms can be applied to other QoS factors.

Keywords: QoS Prediction, Web service, Collaborative Filtering, Deep learning, Autoencoder, Generative Adversarial Network, Self-Organizing Map.

ملخص

في هذه الأطروحة, نقترح نهجًا قائمًا على التعلم العميق يجمع بين نموذج عامل المصفوفة استنادًا إلى وحدة التشفير التلقائي العميقة وتقنية التجميع. من أجل ذلك تم الإستناد على ثلاثة تصاميم مختلفة من شبكات التشفير التلقائي. يتكون النموذج الأول من طبقة مخفية واحدة من العصبونات التي ترمز لمجموعة العوامل الكامنة الممثلة للمستخدمين و / أو الخدمات. بينما يتكون النموذج الثاني من عدة طبقات مخفية. النموذج الثالث هو عبارة عن دمج بين نموذج التشفير التلقائي العميق من جهة وشبكة خصومية توليدية من جهة أخرى. في هذا السياق تمت معالجة عدة صُعُوبات أخرى متعلقة بمشكلة التنبؤ بالقيم المفقودة لجودة الخدمة. الصعوبة الأولى تتعلق بقابلية أنظمة التنبؤ للتأثر بمشكلة قلة البيانات. و للتعامل مع هذه المشكلة, يتألف اقتراحنا في استخدام خوارزمية التجميع إعتمادا على خريطة شبكات كوهونين ذاتية التنظيم, حيث تتم تهيئتها باستخدام مجموعة من الميزات المكانية. الصعوبة الثانية التي تعاملنا معها هي مشكلة الإنطلاقة الفاترة, والتي تحدث عند إضافة مستخدمين و / أو خدمات جديدة إلى نظام التنبؤ. تتم إدارة هذه المشكلة بصفة عامة من خلال استغلال الميزات المكانية كذلك. تُظهر التجارب التي تم إجراؤها أن مقترحاتنا بأسنطاعتها تقديم أداء أفضل من حيث التنبؤ بجودة الخدمة, وبالتالي توفير المزيد من الإرشادات للمستخدمين في اختيارهم للخدمات المفضلة مقارنة بالطرق الحالية. يجدر الإشارة على أن الخصائص المُعَرِّفة لجودة الخدمة التي اعتمدنها من إجل إجراء تجاربنا المختلفة هي زمن الاستجابة وسرعة النقل. ومع ذلك, يمكن تطبيق خوارزميات توقع جودة الخدمة المقترحة على العوامل الأخرى لجودة الخدمة.

الكلمات الدالة: التنبؤ بجودة الخدمة, خدمة ويب, التصفية التعاونية, التعلم العميق, التشفير التلقائي, شبكة خصومية توليدية, خريطة الخصائص ذاتية التنظيم.