REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEURE ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ MATHEMATIQUES ET INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité : Réseaux et télécommunications

Par :

Draoui Fella
Chahbar Fatma

Sur le thème

Une approche basée sur le Deep Learning pour la Recommandation des services web

Soutenu publiquement le 15 / 07 / 2019 à Tiaret devant le jury composé de :

Mr MOSTEFAOUI Sid Ahmed	Grade	Université MCB	Président
Mr MEGHAZI Hadj Madani	Grade	Université MAA	Encadreur
Mr AID Lahcene	Grade	Université MCB	Examinateur

Remerciement

Avant tout nous remercions Dieu le tout puissant de nous avoir donné le courage et nous avoir guidé pour pouvoir mener à bien ce modeste travail.

A notre promoteur Mr MEGHAZI Hadj Madani

Malgré vos multiples préoccupations, vous avez bien voulu nous confier ce travail et le diriger. Nous avons eu le privilège de travailler parmi votre équipe et d'apprécier vos qualités et vos valeurs. Votre sérieux, votre compétence et votre sens du devoir nous ont énormément marqués. Veuillez trouver ici l'expression de notre respectueuse considération et notre profonde admiration pour toutes vos qualités humaines et professionnelles.

A Mr. MOSTEFAOUI Sid Ahmed

C'est pour nous un grand plaisir de vous compter parmi le jury de ce travail, nous tenons à vous témoigner notre profonde reconnaissance d'avoir aimablement accepter de présider le jury de ce mémoire. Veuillez croire à notre gratitude et à notre respectueuse considération.

A Mr. AID Lahcene

Nous sommes profondément reconnaissantes de l'honneur que vous nous faites en acceptant d'examiner notre modeste travail, Veuillez trouver dans ce travail l'expression de notre attention et le témoignage de notre profonde et sincères considération.

Un grand Merci aux enseignants ainsi que l'administration de la faculté informatique qui ont veillé sur notre formation et notre suivi durant tout le cursus d'étude, en particulier Mr. MERATI, d'avoir aimablement accepté de répondre à nos questions.

En fin nous adressons nos remerciements à tous ceux qui ont contribué par leurs conseils ou leurs encouragements à l'aboutissement de ce travail.

Dédicace

Je dédie ce mémoire à …

A mes parents

Aux plus belles créatures que dieu a créées sur terre, à cette source de tendresse, de Patience et de générosité. Aucune dédicace ne pourrait exprimer mon respect, ma considération et mes chaleureux sentiments envers mes Chers Parents, Grâce à leurs tendres encouragements et leurs grands sacrifices, ils ont pu créer le climat affectueux et propice à la poursuite de mes études. Je prie le bon lieu de les bénir, de veiller sur eux, en espérant qu'ils soient toujours fiers de moi.

A ma très chère Grande mère Hanifa

A mes très chers frères Aboubakr, Abdelhak

A mes très chères belles-sœurs Amina, Amel

A mes très chères Nièces Romaissa, Assia

En témoignage de l'attachement, de l'amour et de l'affectation que je porte pour vous

A tous les membres de ma famille

A ma chère amie Intissar

En témoignage de l'amitié qui nous uni et des souvenirs de tous les moments que nous avons passés ensemble, je te dédie ce travail et je te souhaite une vie pleine de santé et de bonheur.

A mes très chers amis Hamouda, Nounou, Madjid

A ma promotion de Master 2 Informatique 2018/2019

Fella

Dédicace

Je dédie ce mémoire à …

*Mon très cher père **CHAHBAR MOHAMED** que j'ai tant voulu qu'il soit présent parmi nous. Son absence m'a bouleversé et laissé une grande lacune que personne ne pourra la comblée.*

*Ma mère **CHARCHEB Messaouda** pour sa gentillesse, son savoir-faire, sa disponibilité, sa tendresse, ses encouragements et son soutien moral dans les moments les plus difficiles. Je vous dois mon éducation, ma réussite dans ma vie et mes études. Merci très chère mère je suis très reconnaissante.*

*Ma deuxième mère, ma très chère tante **CHARCHEB Kheira** que j'ai tant voulu qu'elle soit présente parmi nous.*

*Mes très chères sœurs **Hbiba** et **Aicha** pour l'entente, le soutien moral et la disponibilité pendant les moments les plus difficiles.*

*Tous les membres de ma grande famille pour leur encouragement et leur patience et particulièrement mes très chers frères AEK, **Ghalem** et **Tayeb** mes belles sœurs **Fatiha**, **Hayet** et **Rabia**, mes nièces et mes neveux, mes tantes et mes oncles.*

*Mes cousins et spécialement ma très chère **Zoulikha**.*

*Sincèrement et avec beaucoup d'émotion, je dédie ce travail à mes très chère amis **BRAHIMI AbdelMadjid Amine** et **DRAOUI Fella** pour leur gentillesse, leur patience, ses encouragements et la confiance qu'ils m'ont donnée pendant mes études.*

*Je ne saurais oublier tous mes amis (es) spécialement Sara **Chaib, Bousri Rahil, Charfaoui Younes** et **Fatma**.*

Tous mes collègues et tout le personnel de l'université de Tiaret.

Intissar

Résumé

Quand il s'agit des services web le problème le plus évident est toujours de trouver la meilleure approche pour permettre aux clients la découverte des services les plus pertinents pour une éventuelle requête. Dans cette perspective plusieurs approches ont été proposées, cependant, la majorité ignorent l'historique des interactions des services web. Le présent travail a pour objectif d'explorer une nouvelle piste qui prend en considération ces interactions passées et les exploiter afin de mettre en œuvre une solution issue du domaine du Deep Learning qui permet de recommander la consommation des services web pertinents.

Mots clés : Services Web, Services Web Sémantiques, Services Web Sociaux, Composition des Services Web, Deep Learning, Réseaux de Neurone Artificiels, Système de Recommandation.

Abstract

When it comes to web services the most obvious problem is always finding the best approach to allow customers to discover the most relevant services for a possible query. In this perspective several approaches have been proposed, however, the majority ignore the history of web services interactions. The present work aims to explore a new path that takes into account these past interactions and use them to implement a solution from the field of Deep Learning that can recommend the consumption of relevant web services.

Keywords: Web Services, Semantic Web Services, Social Web Services, Web Services Composition, Deep Learning, Artificial Neural Networks, Recommendation System.

ملخص

عندما يتعلق الأمر بخدمات الويب، فإن المشكلة الأكثر وضوحًا هي دائمًا إيجاد أفضل طريقة للسماح للعملاء باكتشاف أكثر الخدمات ذات الصلة للاستعلام المحتمل. في هذا المنظور، تم اقتراح عدة طرق، ومع ذلك، فإن الأغلبية تتجاهل تاريخ تفاعلات خدمات الويب. يهدف العمل الحالي إلى استكشاف مسار جديد يأخذ في الحسبان هذه التفاعلات السابقة ويستغلها لتنفيذ حل من مجال التعلم العميق يمكن أن يوصي باستهلاك خدمات الويب ذات الصلة .

الكلمات المفتاحية: خدمات الويب، خدمات الويب الدلالي، خدمات الويب الاجتماعية، تكوين خدمات الويب، التعلم العميق، الشبكات العصبية الاصطناعية، نظام التوصية.

Table des matières

Liste des Figures

Introduction générale

Plusieurs technologies innovantes ont été introduites ces dernières années, pour renforcer la communication, le partage des ressources et l'interopérabilité des systèmes. Ces tâches ont mené les experts du domaine à inventer et découvrir des nouvelles méthodes pour résoudre plusieurs problèmes informatiques. Parmi les technologies qui ont apparus au début du 21ème siècle et qui ont été adoptées par les entreprises, c'était la technologie des services web.

Les services web sont des briques de bases logicielles s'affranchissant de toute contrainte de compatibilité logicielle ou matérielles. Les services Web comportent de nombreux avantages, ils sont utilisables à distance via n'importe quel type de plate-forme, ils peuvent servir au développement d'applications distribuées et sont accessibles depuis n'importe quel type de clients. Les services Web appartiennent à des applications capables de collaborer entre elles de manière transparente pour l'utilisateur, leur mise en œuvre repose sur une architecture distribuée. Cette architecture orientée services (Service Oriented Architecture-SOA) est un style architectural fondé sur la description des services et de leurs interactions. Les services sont publiés dans des annuaires par des fournisseurs qui les hébergent. Ils sont accessibles via un réseau pour les utilisateurs qui les découvrent, les sélectionnent, les invoquent et les utilisent, mais ils se retrouvent souvent face au défi du choix de réaliser les meilleures compositions de services qui répondent le mieux à leurs besoins. Une solution serait d'assister les utilisateurs de manière intelligente dans leur quête de services à travers des suggestions pertinentes de ces derniers. Ce processus est connu sous le nom de *la recommandation de services Web*. Recommander du contenu personnalisé en termes de services aux utilisateurs offre plusieurs avantages :

- Le développement des applications modernes repose sur l'invocation d'une multitude d'APIs et de services Cloud. Les développeurs investissent un temps important dans la recherche et la sélection des services à composer ou à invoquer. Une recommandation pertinente de services aux utilisateurs leur permet d'économiser un temps précieux et de se consacrer intégralement aux tâches de développement.
- La recommandation permet d'améliorer la visibilité de services nouvellement déployés sur le réseau, les utilisateurs se verront recommander de nouveaux services dont ils ignoraient l'existence et pourtant pertinents à leurs requêtes. Les fournisseurs de services bénéficieront d'une meilleure visibilité et de plus d'opportunités sur le marché.
- Renforcer les relations business-to-business en permettant aux entreprises de s'engager dans des collaborations à court, moyen et long terme à travers la consommation mutuelle de services recommandés.

Bien que la tâche de recommander du contenu personnalisé de services offre autant d'avantages, elle reste néanmoins une tâche complexe et doit surmonter plusieurs challenges. Les systèmes de recommandations de nos jours et avec l'évolution exponentielle de la masse des données et leurs utilisations sur l'internet, se retrouvent face à de nombreuses difficultés :

- Les algorithmes de recommandation sont des algorithmes analytiques coûteux en temps de calcul et gourmands en terme de ressources hardware. La recommandation doit pouvoir réagir en temps réel aux derniers rebondissements du marché (déploiement de nouveaux services, mises à jour de fonctionnalités). Dans ce but la recommandation doit adopter les techniques de calcul parallèle.
- La précision et la pertinence de la recommandation repose sur l'analyse de larges historiques d'exemples et de plusieurs attributs pour chaque exemple. Un volume large de données à traiter ne doit pas influencer sur la réactivité (temps) ou la qualité (pertinence et précision) de la recommandation.
- Manque de modèles prédictives de qualité sur lesquels construire la recommandation. Les modèles linéaires, malheureusement, aboutissent à des faibles prédictions.

Dans ce sens, l'objectif de notre travail est de proposer une approche basée sur un système de recommandation des services web qui utilisent des données d'interaction sociales entre ces derniers, pour se faire nous avons structuré le présent manuscrit comme suit :

Le premier chapitre intitulé « ***les Services web*** » qui se compose de quatre parties, dans la première, nous présentons quelques définitions, caractéristique et fonctionnement des services web. Dans la deuxième partie, nous présentons les services web sémantiques en évoquant leur évolution du syntaxique au sémantique, ainsi que les standards utilisés dans la description sémantique des services web, Dans la troisième partie nous présentons quelques limites des services web, Dans la quatrième, nous donnons une définition des SWSoc et nous parlons de la valeur ajoutée d'un aspect social aux services Web.

Dans le deuxième chapitre intitulé « ***Les Techniques d'analyse de grandes masses de données*** » nous présentons les algorithmes et les différentes méthodes les plus utilisés pour l'analyse des données.

Dans le troisième chapitre intitulé « ***Conception et Implémentation*** » nous introduisons notre approche adoptée pour la recommandation des services web et les outils utilisés pour la mise en œuvre de notre propre solution, en expliquant toutes les étapes de l'implémentation et les résultats obtenus par l'implémentation de notre solution.

CHAPITRE I

SERVICES WEB

Sommaire

I.1 – Introduction

Au début de l'informatique, le dialogue entre machines nécessite une connaissance approfondie des protocoles réseau et parfois même du matériel réseau. La programmation orientée objet a permis le développement des architectures distribuées en fournissant des bibliothèques de haut-niveau pour faire dialoguer des objets répartis sur des machines différentes entre eux, ce qui a considérablement allégé le travail des programmeurs. Les objets distribués sur le réseau communiquent par messages en s'appuyant sur l'une des technologies suivantes :

- Common Object Request Broker Architecture (CORBA) : ce standard de l'Object Management Group permet de faire communiquer des objets écrits dans des langages différents (C++, Java, Smalltalk) et même d'encapsuler des programmes écrits dans des langages procéduraux pour les faire passer pour des objets.
- Remote Method Invocation (RMI) : cette technologie de Sun permet de faire communiquer très simplement des objets java distribués sur le réseau.
- Les services web.
- NET Remoting : cette méthode permet de créer facilement des applications largement distribuées, si les composants de l'application sont tous sur un même ordinateur ou répartis à travers le monde entier.
- Windows Communication Foundation (WCF) : est une technologie de développement d'applications basées sur une architecture orientée services (SOA)

Nous nous sommes focalisées sur les services web qui permettent l'interconnexion des applications à l'aide des protocoles d'internet. Les services Web sont des technologies émergentes et prometteuses pour le développement, le déploiement et l'intégration d'applications Internet. Un des avantages majeurs des services Web par rapport middlewares traditionnels (CORBA, DCOM et XML-RPC) est l'apport de l'interopérabilité et le couplage faible sur Internet par l'utilisation de technologies connues telles que XML et HTTP. Ces technologies permettent aux applications utilisant différents langages et plates-formes de s'interagir de manière familière.

I.2 – Les services web

Un service Web est un système logiciel identifié par un URI et conçu pour permettre la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués via internet. Il possède une interface définie et décrite dans un format pouvant être traité par une machine (WSDL). Cette définition peut être découverte par d'autres systèmes logiciels. D'autres systèmes peuvent alors interagir avec le service Web de la manière spécifiée par sa description à l'aide de messages SOAP, généralement acheminés via HTTP avec une sérialisation XML conjointement avec d'autres normes relatives au Web. Les services Web représentent donc la façon la plus efficace de partager des méthodes et des fonctionnalités. De plus, ils réduisent le temps de réalisation en permettant de tirer directement parti de services existants.

La figure I.1 schématise les facettes générales d'un service web, en premier lieu un service web est décrit par une interface XML nommée WSDL, il peut échanger des documents XML avec d'autres services à l'aide du protocole SOAP, il peut être recherché dans un annuaire tel que l'UDDI.

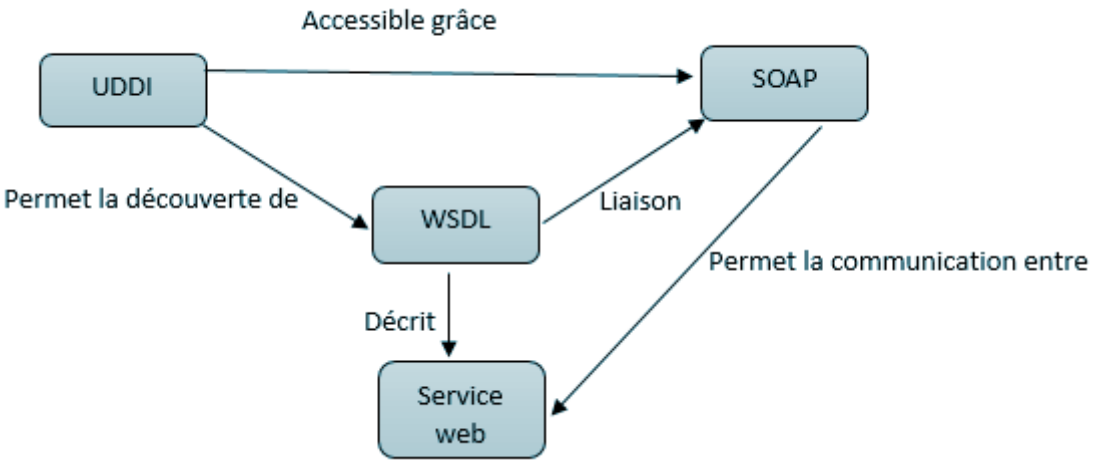


Figure I.1 – *Les différentes relations entre les spécifications des services web.*

I.2.1 – Les caractéristiques des services web

La technologie des services Web repose essentiellement sur une représentation standard des données (interfaces, messageries) au moyen du langage XML. Cette technologie est devenue la base de l'informatique distribuée sur Internet et offre beaucoup d'opportunités au développeur Web. Un service Web possède les caractéristiques suivantes :

- Il est accessible via le réseau.
- Il dispose d'une interface publique (ensemble d'opérations) décrite en XML.
- Ses descriptions (fonctionnalités, comment l'invoquer et où le trouver ?) sont stockées dans un annuaire.

- Il communique en utilisant des messages XML, ces messages sont transportés par des protocoles Internet (généralement HTTP, mais rien n'empêche d'utiliser d'autres protocoles de transfert tels : SMTP, FTP, BEEP...).
- L'intégration d'application en implémentant des services Web produit des systèmes faiblement couplés, le demandeur du service ne connaît pas forcément le fournisseur.

Ce dernier peut disparaître sans perturber l'application cliente qui trouvera un autre fournisseur en cherchant dans l'annuaire [1].

I.2.2 – Architecture d'un service Web

Les services Web reprennent la plupart des idées et des principes du Web (HTTP, XML) et les appliquent à des interactions entre machines. Comme pour le World Wide Web, les services Web communiquent via un ensemble de technologies fondamentales qui partagent une architecture commune. Ils ont été conçus pour être réalisés sur de nombreux systèmes développés et déployés de façon indépendante. Les technologies utilisées par les services Web sont HTTP, WSDL, REST, XML-RPC, SOAP et UDDI. Ces technologies seront détaillées tout au long du reste du chapitre. Le déroulement général du fonctionnement s'articule autour de trois principaux acteurs comme le montre la Figure I.2.

- **Fournisseur de service :** Développe les fonctionnalités du service, déploie ce dernier sur un serveur local/Web/Cloud. Puis permet l'invocation de son service par les clients en publiant l'interface WSDL sur un registre UDDI. Met en application le service Web et fournit une description de ce dernier en utilisant le langage WSDL disponible sur Internet.
- **Demandeur de service** : C'est le consommateur du service Web. Il émet une requête pour la recherche et l'invocation d'un service Web. Cette requête est une demande en XML (REST, XML-RPC, SOAP).
- **Annuaire service** : Le registre principal des services Web permet la découverte (récupération de la description et l'URL) et la publication des services Web par le biais de référentiel UDDI.

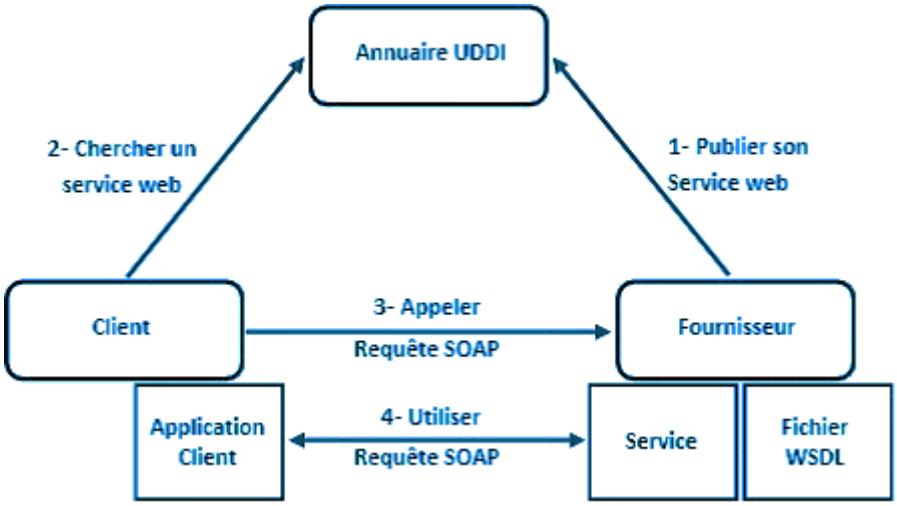


Figure I. 2 – *Architecture Des Services Web[2].*

Les interactions entre ces trois acteurs mentionnés précédemment se font initialement par le fournisseur qui diffuse les descriptions de ses services Web dans l'annuaire. Une fois que les services sont disponibles sur l'annuaire, le client (ou demandeur de service) s'adresse à ce dernier pour chercher un service particulier. Alors il récupère sa description et URL qu'il invoquera auprès de fournisseur de services.

I.2.3 – Les couches des Services Web

Les services Web respectent une structure en couches formée de quatre couches principales :

- **Couche transport :** Cette couche est responsable du transport des messages XML échangés entre les applications. Actuellement, cette couche inclut HTTP, SMTP, FTP et de nouveaux protocoles tels que BEEP.
- **Couche communication :** Cette couche est responsable du formatage des données échangées de sorte que les messages peuvent être compris à chaque extrémité. Actuellement, deux styles architecturaux totalement différents sont utilisés pour ces échanges de données. Nous avons d'un côté l'architecture orientée opérations distribuées (protocoles RPC) basée sur XML et qui comprend XML-RPC et SOAP et de l'autre côté une architecture orientée ressources Web, REST (Representational State Transfer) qui se base uniquement sur le bon usage des principes du Web (en particulier, le protocole HTTP).
- **Couche description de service :** Cette couche est responsable de la description de l'interface publique du service Web. Le langage utilisé pour décrire un service Web est WSDL qui est la notation standard basée sur XML pour construire la description de l'interface d'un service. Cette spécification définit une grammaire XML pour décrire les services Web comme des ensembles de points finaux de communication (ports) à travers lesquels on effectue l'échange de messages.

- **Couche découverte de service :** Cette couche est chargée de centraliser les services dans un registre commun et de simplifier les fonctionnalités de recherche et de publication des services Web. Actuellement, la découverte des services est assurée par un annuaire UDDI (Universal Description, Discovery, and Integration) [1].

I.2.3.1 – WSDL

WSDL (Web Services Description Language) est un langage de description standard. C'est l'interface présentée aux utilisateurs. Il indique comment utiliser le service Web et comment interagir avec lui. La notation utilisée par un fichier WSDL pour décrire les formats de messages est basé sur la norme du schéma XML, ce qui signifie que WSDL est à la fois neutre par rapport au langage de programmation et à la plateforme.

Il permet de décrire de façon précise les détails concernant le service Web tels que les protocoles, la localisation du service, les ports utilisés, les méthodes pouvant être invoquées, les opérations pouvant être effectuées, les formats des messages d'entrée et de sortie et les exceptions pouvant être envoyées.

Un document WSDL définit une suite de descriptions de composants (Figure I.3). Le fichier WSDL est principalement composé de six éléments, avec d'autres éléments optionnels :

- **Définitions** : élément racine du document, il donne le nom du service, déclare les espaces de noms utilisés et contient les éléments du service.
- **Types :** décrit tous les types de données utilisés entre le client et le prestataire. WSDL n'est pas exclusivement lié à un système spécifique de typage, mais utilise par défaut la spécification XML Schéma.
- **Message :** décrit un message de message unique, que ce soit requête seul ou un message de réponse seul. L'élément définit le nom du message et peut contenir (ou pas) des éléments part, qui font référence aux paramètres du message ou aux valeurs retournées par le message.
- **PortType :** combine plusieurs messages pour composer une opération. Chaque opération se réfère à un message en entrée et à des messages en sortie.
- **Binding :** d'écrit les spécifications concrètes concernant la manière dont le service sera implémenté : protocole de communication et format des données pour les opérations et messages définis par un type de port particulier. WSDL possède des extensions internes pour définir des services SOAP ; de fait, les informations spécifiques à SOAP se retrouvent dans cet élément.
- **Service** : défini les adresses permettant d'invoquer le service donné, ce qui sert à regrouper un ensemble de ports reliés. Plupart du temps, c'est une URL invoquant un service SOAP [2].

Figure I.3 – *Exemple de fichier WSDL [2].*

I.2.3.2 – XML

XML (e**X**tensible **M**arkup **L**anguage) est la langue maternelle des services web, c'est un standard promulgué par le W3C, l'organisme chargé de standardiser les évolutions du web. On retrouve dans l'XML une généralisation des idées contenues dans HTML. XML permet de définir des balises et de leur associer une interprétation. Dans HTML, on n'utilise les balises que pour décrire l'aspect graphique que doit revêtir la page dans le navigateur web. Dans XML, les balises permettent d'associer toutes sortes d'information au fil du texte.

XML a été conçu pour des documents arbitrairement complexes, tout en s'appuyant sur cinq grands principes simples et clairs :

- Lisibilité à la fois par les machines et par les utilisateurs.
- Définition sans ambigüité du contenu d'un document.
- Définition sans ambigüité de structure d'un document.
- Séparation entre documents et relations entre documents.
- Séparation entre structure du document et présentation du document [3].

I.2.3.3 – SOAP

SOAP (Simple object Access Protocol) est un protocole standard de communication. C'est l'épine dorsale du système d'interopérabilité. SOAP est décrit en XML et standardisé par le W3C. Il se présente comme une enveloppe pouvant être signée et pouvant contenir des données ou des pièces jointes (voir Figure I.4).
Il circule sur le protocole HTTP et permet d'effectuer des appels de méthodes à distance [1].

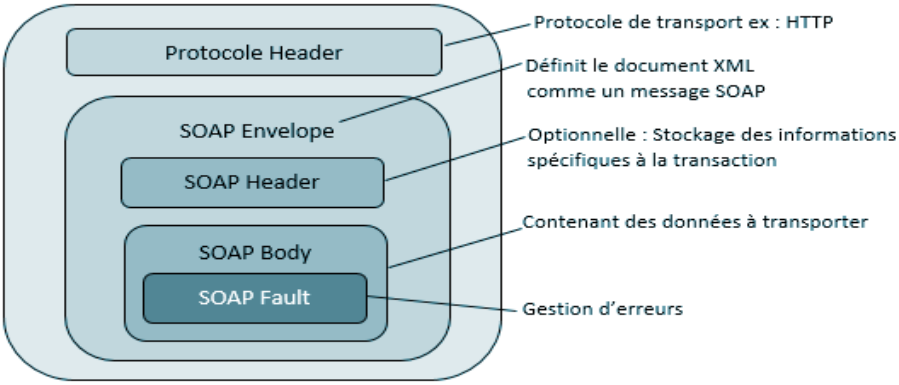


Figure I.4 – *La structure d'un message SOAP.*

Un message SOAP contient une enveloppe obligatoire, un en-tête facultatif et un corps obligatoire qui se présente sous la forme d'un document XML.

- **Envelope:** c'est lui qui contient le message et ses différents sous-blocs. Il s'agit du bloc racine XML. Il peut contenir un attribut encodingStyle dont la valeur est une URL vers un fichier de typage XML qui décrira les types applicables au message SOAP.

Exemple :

```
<?xml version="1.0" encoding="utf-8"?>
 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding">
        <soap:Header>
            <!-- en-tête -->
        </soap:Header>
        <soap:Body>
            <!-- corps -->
        </soap:Body>
</soap:Envelope>
```

• **Header:** c'est un bloc optionnel qui contient des informations d'en-têtes sur le message. S'il est présent, ce bloc doit toujours se trouver avant le bloc Body à l'intérieur du bloc Envelope.

Exemple :

```
<!--Élément USER : À destination du nœud RightManager-->
<soap:Header>
   <m:User xmlns:m="http://www.monsite.com/rights/"
    soap:actor="http://www.monsite.com/rights/RightsManager">
          Thunderseb
   </m:User>
    <!-- Élément Session : À destination du nœud final -->
   <m:Session xmlns:m="http://www.monsite.com/session/" soap:mustUnderstand="1">
          12AE3C
   </m:Session>
     <!-- Élément USER : À destination du prochain nœud -->
   <m:Lang xmlns:m="http://www.monsite.com/lang/"
   soap:actor="http://schemas.xmlsoap.org/soap/next" soap:mustUnderstand="0"> FR
   </m:Lang>
</soap:Header>
```

• **Body:** c'est le bloc qui contient le corps du message. Il doit absolument être présent de manière unique dans chaque message et être contenu dans le bloc Envelope. SOAP ne définit pas comment est structuré le contenu de ce bloc. Cependant, il définit le bloc Fault qui peut s'y trouver.

Exemple :

```
<soap:Body>
     <checkAccountBalance>
          <accountNumber xsi: type="xsd:int">1234567890</accountNumber>
     </checkAccountBalance>
</soap:Body>
```

• **Fault:** ce bloc est la seule structure définie par SOAP dans le bloc Body. Il sert à reporter des erreurs lors du traitement du message, ou lors de son transport. Il ne peut apparaître qu'une seule fois par message. Sa présence n'est pas obligatoire [1].

Exemple :

```
<soap:Body>
  <soap:Fault>
    <!-- Identifiant de l'erreur ? défini par SOAP -->
          <faultcode>soap:Server</faultcode>
    <!-- Description brève du message -->
    <faultstring>Impossible de router le message. </faultstring
    <!-- Composant qui a généré l'erreur (URL). -->
    <faultactor>http://www.monsite.com/messageDispatcher</faultactor>
    <!-- Message spécifique à l'application -->
    <detail>
      <m:error xmlns:m="http://www.monsite.com/errors"> E_NO_ROUTE </m:error>
    </detail>
  </soap:Fault>
</soap:Body>
```

Sur le web, la qualité de service et la latence sont variables et le réseau résulte plus de l'agrégation de nœuds et de sous-réseaux hétérogènes partageant la couche TCP/IP la plus profonde que d'un développement discipliné. Seul un protocole simple mais robuste peut alors s'accommoder, au niveau supérieur, de tant de variabilité : c'est le protocole HTTP.

I.2.3.4 – HTTP

HTTP (HyperText Transfer Protocol), désigne dans le langage informatique un protocole de communication entre un client et un serveur pour le World Wide Web. Il a été inventé dans les années 1990 par Tim Berners-Lee. Il établit une liaison entre un ordinateur (client) et un serveur Web. Le premier, via un navigateur Web, envoie une requête au second qui lui apporte une réponse presque instantanée. Autrement dit, le protocole de communication HTTP est ce qui permet à un internaute d'accéder à un contenu (une page Web, un fichier CSS, etc.), en commandant au serveur d'effectuer une action. Les requêtes et les réponses sont composées d'un entête et d'un corps de message exprimées en texte ASCII.

Sur un plan plus technique, le protocole HTTP est un protocole de la couche application, 7ieme couche du modèle OSI. Il utilise par défaut le port logiciel 80.

I.2.3.5 – UDDI

L'annuaire des services UDDI (Universal Description, Discovery and Integration) est un standard pour la publication et la découverte des informations sur les services Web. Il permet aux fournisseurs de présenter leurs services Web aux clients. La spécification UDDI vise à créer une plate-forme indépendante, un espace de travail (FrameWork) ouvert pour la description, la découverte et l'intégration des services des entreprises.

L'UDDI se concentre sur le processus de découverte de l'architecture orientée services (SOA) et utilise des technologies standards telles que XML, SOAP et WSDL qui permettent de simplifier la collaboration entre les partenaires dans le cadre des échanges commerciaux. L'accès au référentiel s'effectue de différentes manières.

- Les pages blanches comprennent la liste des entreprises ainsi que des informations associées à ces dernières (coordonnées, description de l'entreprise, identifiants...).
- Les pages jaunes recensent les services Web de chacune des entreprises sous le standard WSDL.
- Les pages vertes fournissent des informations techniques précises sur les services fournis.

Le scénario classique d'utilisation de UDDI est illustré (Figure I.5). L'entreprise B a publié le service Web S et l'entreprise A est client de ce service [1].

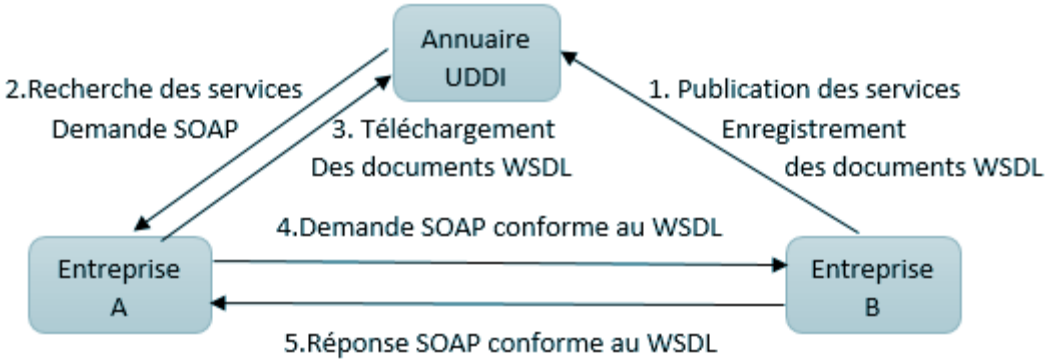


Figure I.5 – *Schéma générale de l'annuaire UDDI.*

Les services Web sont des applications auto-descriptives, modulaires et faiblement couplées qui fournissent un modèle simple de programmation et de déploiement d'applications, basé sur des normes et s'exécutant au travers de l'infrastructure Web. Tandis que la limitation majeure de la technologie des services Web est que la découverte et la composition des services nécessitent encore une intervention humaine accrue. Ceci constitue un handicap, surtout avec les montées en charge des services Web. Pour résoudre ce problème, la communauté du Web sémantique a proposé d'enrichir les services Web avec un contenu sémantique de leurs fonctionnalités afin de faciliter leur découverte et leur intégration. L'application du Web sémantique aux services Web a donné naissance aux services Web sémantiques (SWS). D'une manière automatisée, les utilisateurs et les agents logiciels devraient pouvoir découvrir, appeler, composer et surveiller l'exécution des services Web.

I.3 – Les services web sémantique

Un service web sémantique est le résultat de l'intégration de concepts sémantiques à la description du service web. Le concept des services web sémantiques se trouve à la convergence de deux domaines de recherche importants concernant les technologies de l'internet : le web sémantique et les services web (Figure I.6). La combinaison de ces deux tendances promet un Web entièrement mécanisée pour l'interaction de l'ordinateur, où les gens organisent leur collaboration et leurs relations d'affaires [4].

Figure I.6 – *L'évolution du web [5].*

Cette tâche de convergence est accomplie en rendant les services Web auto-exploitables par machines et de réaliser l'interopérabilité entre les applications via le Web en vue de rendre le Web plus dynamique.

La notion de services Web sémantiques est synonyme de l'automatisation des tâches d'utilisation de services tels que la découverte, la sélection, la composition et l'adoption de services appropriés.

I.3.1 – L'objectif des services web sémantiques

L'objectif des services web sémantique est de créer un web sémantique des services web dont les propriétés, les capacités, les interfaces et les effets sont décrits de manière non ambigüe et exploitable par des machines et ce en utilisant les couches techniques sans être conceptuellement dépendants [6].

La combinaison des technologies du Web sémantique à celles des Web services permettra de :

- Sélectionner des services web.
- Automatiser la découverte des services web, c'est-à-dire la localisation automatique des services web qui fournissent une fonctionnalité particulière et qui répondent aux propriétés demandées par l'utilisateur.
- Automatiser l'invocation d'un Web service, cela implique l'automatisation de l'exécution du service web par le programme d'utilisateur ou par un agent logiciel.

I.3.2 – L'architecture des services web sémantiques

Le groupe W3C travaille activement à l'élaboration d'une architecture étendue standard. Cette architecture est basée sur des services web, la pile est constituée de plusieurs couches, chaque couche s'appuyant sur un standard particulier. On retrouve, au-dessus de la couche de transport, les trois couches formant l'infrastructure de base décrite précédemment comme illustré dans la Figure I.7.

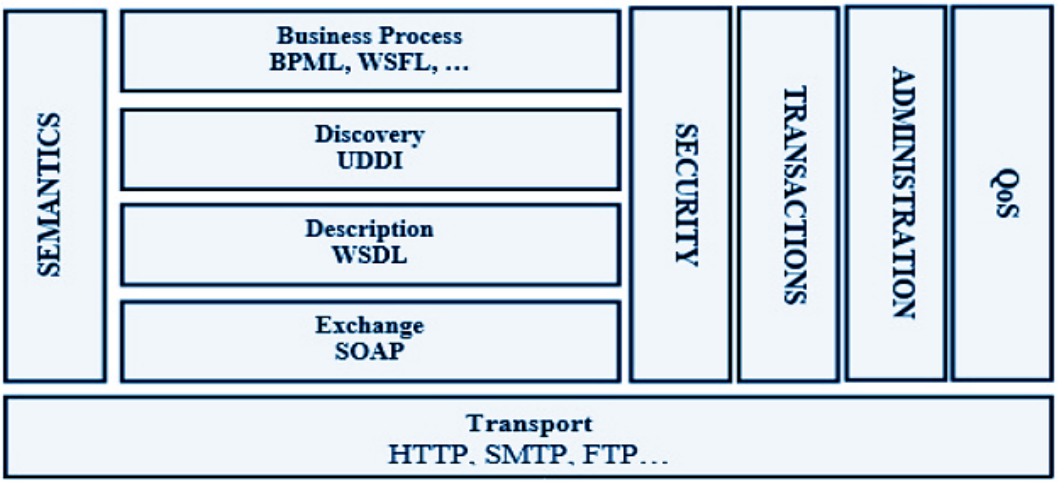


Figure I.7 – *Pile des services web sémantique [7].*

Ces couches s'appuient sur les standards émergeants SOAP, WSDL et UDDI. Une couche de business process est ajoutée, permettant de rendre les processus métiers accessibles à l'intérieur d'une entreprise et au-delà même de ces frontières. Les couches transversales : sécurité, transactions, administration et QoS, permettent de compléter la couche business process. La couche sémantique vient s'intégrer à chacune des quatre couches supérieures et permettant ainsi d'automatiser ces processus

I.3.3 – Descriptions sémantiques des services web

Un service Web est accessible à travers sa description WSDL mais cette dernière contient des détails techniques de communication avec le service. Par exemple les fonctions du service Web, les inputs de ces fonctions, les outputs des fonctions. Ces détails ne sont pas suffisant pour bien décrire la fonctionnalité d'un service. Alors les chercheurs ont rajouté une deuxième couche de description c'est la couche sémantique. Deux méthodes possibles pour la description sémantique de Web services [8] :

La première approche consiste à annoter les langages existants (WSDL) avec l'information sémantique. Le principal avantage de ce genre de solutions est la facilité pour les fournisseurs de services d'adapter leurs descriptions existantes aux annotations proposées. Exemple : SAWSDL.

La deuxième approche réécrit entièrement la description syntaxique du Web services en utilisant les technologies du Web sémantique Exemple : OWL-S.

I.3.3.1 – SAWSDL (Semantic Annotation for WSDL)

SAWSDL s'inscrit dans le cadre des approches à base d'annotations. C'est un langage recommandé par le W3C pour la description sémantique des services Web. Il est évolutif et compatible avec les standards de services Web existants, en particulier avec WSDL. En effet, il augmente l'expressivité du langage WSDL avec la sémantique en utilisant des concepts de l'ontologie. SAWSDL fournit un mécanisme permettant d'annoter sémantiquement les types de données et les opérations de WSDL. En particulier, il ajoute aussi de nouveaux éléments pour des spécifications spéciales. Pour ce faire, il définit un ensemble d'attributs pour l'extension de WSDL en proposant un mécanisme d'annotation indépendant du langage de représentation sémantique. Les annotations sémantiques sont réalisées par le biais des références à des modèles conceptuels comme les ontologies. Au lieu de spécifier un langage pour représenter les modèles sémantiques. SAWSDL prévoit des mécanismes par lesquels les concepts des modèles sémantiques peuvent être référencés à l'aide des annotations (voir Figure I.8).

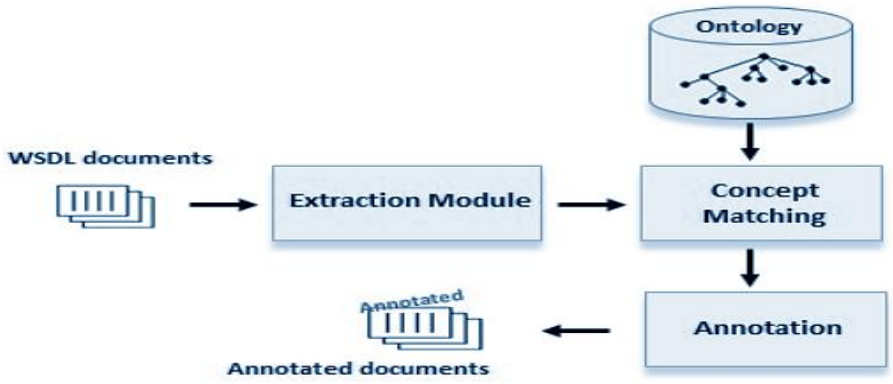


Figure I.8 – *Processus d'annotation sémantique.*

I.3.3.2 – OWL-S (Ontology Web Language for Services)

OWL-S est un sous ensemble du langage OWL dédié à la description sémantique de Web services. Il permet de décrire d'une façon non ambigüe les Web services de telle sorte qu'un agent logiciel puisse exploiter automatiquement ces informations. OWL-S permet la découverte automatique et la composition de Web services. Une description OWL-S se compose de trois éléments (voir Figure suivante) : le service Profile, le service Model et le service Grounding, qui décrivent respectivement que fait le service, comment le service fonctionne et comment accéder au service.

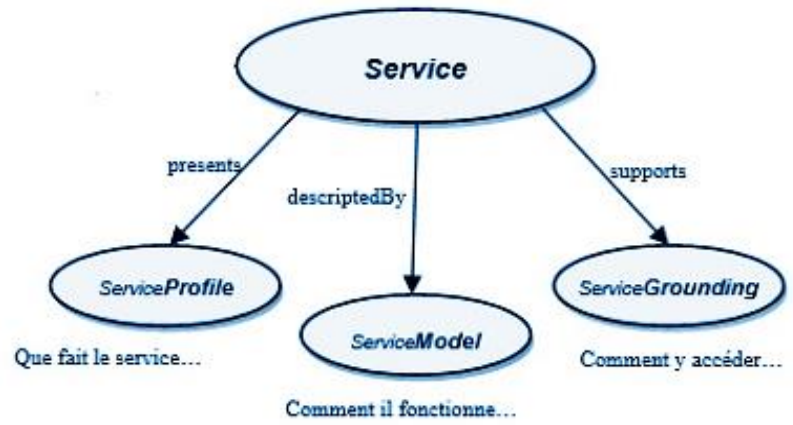


Figure I.9 – *Model conceptuel d'OWL-S [4].*

- **Service profile :** OWL-S fournit cette classe pour décrire un Web service. Cette classe Service profile spécifie trois informations :
  - Nom du service, contacts et description textuelle du service : le nom du service est utilisé comme identificateur du service, tandis que les informations contacts et la description textuelle sont destinées aux utilisateurs humains.
  - Description fonctionnelle du service : Elle spécifie ce que l'on peut attendre du service en termes d'entrées attendues et de résultats produits en sortie. Les transformations d'informations sont représentées par des Inputs et des Outputs. Le changement d'état du monde réel cause par l'exécution du service est représenté par préconditions et effets qui sont les préconditions et les post conditions de son exécution. Les Inputs et les Outputs font références à des classes d'OWL décrivant les types des instances à envoyer au service et aux réponses respectives attendues.
  - Propriétés additionnelles : Plusieurs propriétés sont utilisées pour définir un Web service. La première est la catégorie du Web service. La seconde est la qualité de Web service QoS. Enfin, le Web service peut fournir une liste de paramètres de fa con libre.
- **Service grounding :** La classe OWL-S Service Grounding définit les détails techniques permettant d'accéder au Web service. Les deux premières classes Service Profile et Service Model d'une description OWL-S s'attachent à abstraire la représentation d'un Web service. Service Grounding est la forme concrète d'une représentation abstraite, elle fournit les détails concrets d'accès au Web service, tels les protocoles, les Uris, les messages envoyés, etc [2].

- **Service model :** Les Web services peuvent être décrits en OWL-S en tant que processus grâce à la classe Process. La classe ainsi définie est une sous-classe de Service model. Pour décrire un processus, on spécifie ses entrées, sorties et ses états (préconditions et effets). Les transitions d'un état à un autre sont décrites par les préconditions et les effets de chaque processus. Il existe trois types de processus :
    - Les processus atomiques (AtomicProcess) : exécutable en une seule étape, un processus atomique ne peut pas être décomposé de fa con plus profond. Son exécution correspond à une unique avancée dans l'exécution du service, il est directement invoqué par l'utilisateur du service.
    - Les processus composites (CompositeProcess) : un processus composite est constitué par l'assemblage d'autres processus (eux-mêmes peuvent être composites ou non composite). Les processus composites associent des processus à l'aide de structures de contrôle permettant de décrire leur logique d'exécution.
    - Les processus simples (SimpleProcess) : Les processus simples ne sont pas invocables mais comme les processus atomiques, leurs exécutions s'exécutent en une seule étape. Les processus simples sont employés comme éléments d'abstraction. Un processus simple peut être employé pour fournir une vue d'un certain processus atomique, ou une représentation simplifiée d'un certain processus composite.

I.4 – Les limites des services web

L'état actuel de la SOA a limité l'utilisation généralisée des services Web (ou des services conformes aux principes de la SOA en général) car plusieurs problèmes importants n'ont toujours pas été résolus, notamment celui de la publicité des services pour une meilleure exposition immédiate et de la découverte des services en fonction des besoins des utilisateurs, comment faire confiance aux services quand ils sont trouvés et comment remplacer sans heurts les services quand ils échouent. Ces problèmes ont empêché les services de réaliser leur potentiel, car ils :

- Ne connaissent qu'eux-mêmes, pas leurs utilisateurs ni leurs pairs.
- Limitent considérablement l'intervention des utilisateurs et fonctionnent comme des boîtes noires.
- Ne prennent en compte que leurs propres détails fonctionnels et non fonctionnels internes lors de l'exécution et ignorent d'autres détails externes, tels que les interactions des utilisateurs passés.
- Ne peuvent pas déléguer leurs invocations.
- Ne coopèrent pas spontanément et volontairement les uns avec les autres, ni ne s'auto-organisent.
- Ne peuvent pas réconcilier les ontologies entre eux ou avec leurs utilisateurs.

Les incarnations actuelles des services Web sont impraticables et presque inutilisables, sauf dans des environnements d'entreprise soigneusement contrôlés. Le problème réside dans la sémantique utilisée pour caractériser les services : la sémantique dans les descriptions de service WSDL, ou dans les extensions proposées de WSDL, est inadéquate pour la découverte automatique. Il a été suggéré qu'une communauté d'utilisateurs puisse fournir des descriptions sémantiques via un effort similaire à celui de Wikipédia. Cependant, pour que des services appropriés et utiles soient découverts et utilisés, une sémantique suffisamment précise pour décrire les services doit être combinée à un logiciel suffisamment intelligent pour comprendre la description sémantique. En outre, une meilleure sémantique descriptive améliore uniquement la partie découverte du problème de service concret, pas la partie exécution à l'exécution.

Heureusement, la sémantique comportementale est disponible dans la manière dont les services sont utilisés et combinés, ainsi que dans leur comportement et leurs interactions. C'est l'aspect social des services web [9].

I.5 – Les services web sociaux

Nous considérons les SWSoc comme le résultat de la fusion de Social computing (l'informatique sociale) et de Service-Oriented Computing SOC (l'informatique orientée service). D'une part, le social computing est la facilitation informatique des études sociales et des dynamiques sociales humaines, ainsi que la conception et l'utilisation de technologies de l'information et de la communication tenant compte du contexte social [10]. L'informatique sociale concerne également les actions collectives, le partage de contenu et la diffusion d'informations en général. D'autre part, l'informatique orientée service SOC construit des applications sur les principes d'offre et de demande de service, de couplage faible et de flux d'organisation croisée [11].

La combinaison de l'informatique sociale et de l'informatique orientée services permet aux SWSoc de "savoir" avec qui ils ont travaillé dans le passé et avec lesquels ils aimeraient potentiellement travailler à l'avenir. Ces deux éléments horodatés constituent la « mémoire » des actions que les SWSoc peuvent accumuler dans le temps et appliquer à l'avenir [9]. Elles montrent également l'action collective d'un groupe de SWSoc partageant leurs expériences respectives en réponse à des demandes de développement de services composites complexes à valeur ajoutée.

Les SWSoc devraient prendre l'initiative de conseiller les utilisateurs sur la manière de développer et de réutiliser des services à valeur ajoutée.

I.5.1 – La valeur de l'ajout de réseaux sociaux aux services Web

Lorsque les entreprises découvrent et utilisent des services Web pour répondre à leurs besoins, elles sont incluses dans les compositions de services en fonction des fonctionnalités qu'elles offrent et de la qualité de service (QoS) qu'elles peuvent garantir, ce qui implique un besoin de contrats [9].

Cependant, lorsque les consommateurs engagent et composent des services, cela devient beaucoup plus informel et dynamique, un peu comme on télécharge les applications iPhone. Mais contrairement aux applications iPhone, qui sont monolithiques et fonctionnent indépendamment les unes des autres, les services Web sont destinés à être composés. Leur fonctionnalité et leur qualité de service sont interdépendantes des autres services.

De plus, ils s'exécutent à distance et avec un certain degré d'autonomie. Leur découverte et leur engagement ultérieur deviennent ainsi des activités sociales, un peu comme la collaboration et la concurrence soutenues dans les réseaux sociaux [9].

Les réseaux sociaux illustrent l'énorme popularité des applications Web 2.0, qui aident les utilisateurs à devenir proactifs. Familièrement, nous pouvons maintenant désigner les utilisateurs à la fois en tant que prosommateurs, fournisseurs et consommateurs. Les prosommateurs affichent les définitions sur les wikis, établissent des groupes d'intérêts et partagent des conseils et des astuces. Ces différentes opérations illustrent les principes « J'offre des services dont quelqu'un d'autre pourrait avoir besoin » et « J'ai besoin de services que quelqu'un d'autre pourrait offrir » sur lesquels repose la SOA [9].

Les offres et demandes de services démontrent parfaitement le comportement des gens dans la société d'aujourd'hui, en imposant une dimension sociale à la façon dont les services Web doivent être gérés en termes de description, de découverte, de liaison et de composition. Et si cette dimension sociale était le chaînon manquant ?

Cela pourrait constituer un ingrédient supplémentaire aux méthodes formelles prenant en charge les besoins SOA, à savoir la description, la découverte, la liaison et la composition de services. L'intégration d'éléments sociaux dans le fonctionnement des services Web signifie de nouveaux services Web sociaux (SWSoc) qui :

- Établiront et entretiendront des réseaux de contacts.
- Placeront les utilisateurs explicitement ou implicitement au cœur de leur cycle de vie permettant ainsi des fonctionnalités supplémentaires grâce à la collaboration.
- Compteront sur des contacts privilégiés en cas de besoin.
- Formeront avec d'autres pairs des groupes de collaboration solides et durables.
- Sauront avec qui s'associer pour minimiser la réconciliation d'ontologies [9].

I.5.2 – Services Web sociaux en action

Lorsque les services Web sont « socialisés », ils peuvent fournir des informations sur leur comportement et leur utilisation par le passé. Sur le plan architectural, les services déployés dans un nuage pourraient exploiter plus facilement leurs aspects sociaux [9]. L'établissement et la maintenance des réseaux sociaux des services Web peuvent se faire de trois manières:

- **Collaboration** : En combinant leurs fonctionnalités respectives, les SWSoc ont la capacité de travailler ensemble sur des requêtes utilisateur complexes. En conséquence, un SWSoc gère son propre réseau de collaborateurs afin de décider s'il aime collaborer avec des pairs sur la base d'expériences antérieures. Il peut également recommander des pairs.
- **Concurrence** : Les SWSoc se font concurrence quand ils offrent des fonctionnalités similaires. Leurs propriétés non fonctionnelles les différencient lorsque les exigences non fonctionnelles des utilisateurs doivent être satisfaites. Par conséquent, un SWSoc découvre son propre réseau de concurrents, ce qui lui permet d'essayer d'améliorer ses propriétés non fonctionnelles par rapport à d'autres pairs.
- **Substitution** : Même si les SWSoc se font concurrence, ils peuvent encore s'entraider quand ils échouent s'ils offrent des fonctionnalités similaires. Par conséquent, un SWSoc gère ses propres réseaux de substituts afin de pouvoir respecter ses contrats de niveau de service (service-level agreements SLA) en cas de défaillance potentielle. Il peut ensuite identifier ses propres meilleurs substituts en réponse aux besoins non fonctionnels des utilisateurs [9].

Ces trois manières de maintenir les réseaux sociaux peuvent être considérées indépendamment comme un réseau de comportements sociaux. Ils peuvent être le point de départ de la construction de plusieurs réseaux, en fonction des interactions entre les services Web tels que la délégation et la supervision [9].

I.5.3 – Systèmes de recommandation et plates-formes sociales

De nos jours, l'utilisation généralisée d'Internet dans le monde entier permet à beaucoup de personnes de se connecter. Cette explosion du Web 2.0 (blogs, wikis, sites de partage de contenu, réseaux sociaux, etc.) suscite un besoin croissant des systèmes de recommandation basés sur des méthodes d'exploration de réseaux sociaux et d'informations. Pour de tels systèmes, la structure sociale sous-jacente, également appelée réseau social ou communauté virtuelle, peut être exploitée [12].

La croissance substantielle du réseau social pose à la fois des défis et de nouvelles possibilités de recherche en RS[1]. La raison principale est le fait que le réseau social transforme les consommateurs d'informations en contributeurs actifs, leur permettant de partager leur statut, de commenter ou de noter le contenu Web. Trouver des contenus pertinents et intéressants au bon moment et dans le bon contexte est un défi pour les approches de recommandation existantes.

Parallèlement, la principale valeur ajoutée des plates-formes sociales est d'encourager l'interaction entre les utilisateurs. Chaque interaction peut être extraite et utilisée comme entrée pour le système de recommandation, car elle permet de mieux comprendre les intérêts des utilisateurs et les besoins en informations. En outre, la structure du réseau social sous-jacent dans une plate-forme sociale peut contribuer à générer des recommandations plus fiables (Par exemple, en tenant compte de la distance sociale dans le processus de recommandation, nous faisons généralement confiance à davantage de recommandations provenant de liens plus proches). Par conséquent, nous pouvons en conclure que le réseau social offre une énorme opportunité d'améliorer les systèmes de recommandation (Fig. I.10) [12].

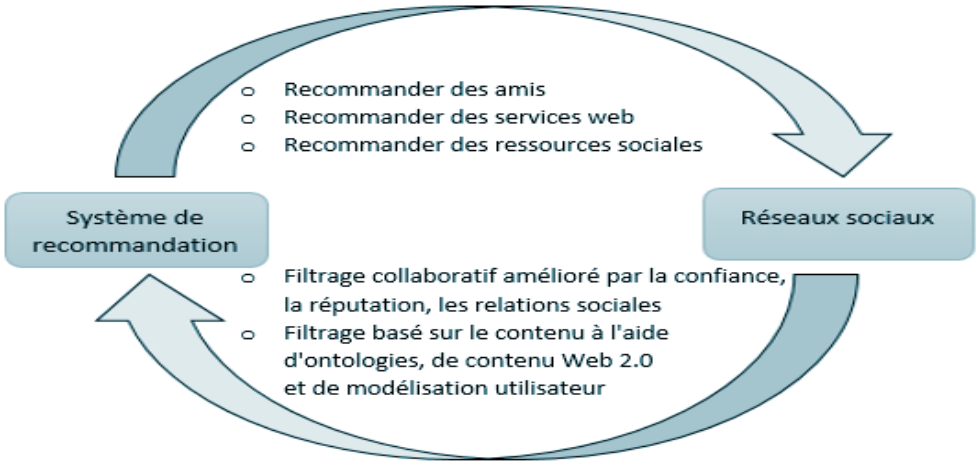


Figure I. 10 – *Contributions réciproques entre systèmes de recommandation et réseaux sociaux [12].*

D'autre part, les systèmes de recommandation peuvent clairement aider à améliorer la participation des utilisateurs aux systèmes sociaux, car ils peuvent recommander de nouveaux amis, service ou un contenu intéressant. Ainsi, l'utilisateur sera plus motivé pour continuer à participer à la plate-forme sociale, parce que plus les utilisateurs partagent de contenu, plus le système peut recommander des connexions pertinentes, avec un profil précis à son sujet.

En utilisant cette connexion entre les plates-formes sociales et les systèmes de recommandation, de nouveaux scénarios peuvent être définis pour des applications avancées, telles que la recommandation de personnes ou diverses recommandations de contenu (par exemple, des balises pour une annotation photo) [12].

[1] RS : Réseaux sociaux

I.6 – Conclusion

Étant donné les applications croissantes de service computing et du cloud computing, un grand nombre de services Web sont déployés sur Internet, ce qui déclenche la recherche de recommandation de service Web. Malgré la qualité de service, l'utilisation des commentaires des utilisateurs est devenue la tendance actuelle en matière de recommandation de service. En même temps, avec la prédominance des réseaux sociaux, les utilisateurs interagissent de manière active avec divers ordinateurs et utilisateurs, ce qui entraîne un volume énorme de données disponibles, telles que les informations de service, les évaluations de service utilisateur, les journaux d'interaction et les relations utilisateurs. Donc, La capture et l'analyse de ces donnes via les techniques d'IA va améliorer substantiellement la qualité de recommandation des services Web aux utilisateurs et aux entreprises.

Dans le chapitre suivant, nous allons introduire les techniques majeures qui permettent d'analyser de grands volumes de data et ainsi la prise de décisions pertinentes pour l'épanouissement business des entreprises.

CHAPITRE II

LES TECHNIQUES D'ANALYSES DE GRANDES MASSES DE DONNEES

Sommaire

II.1 Introduction

Avec l'avènement des réseaux sociaux et de l'Internet of Things (IoT), des milliards d'informations générées par jour entrant dans nos réseaux informatiques, sur le World Wide Web et sur divers dispositifs de stockage de données provenant des entreprises, de la société, de la médecine, des recherches scientifiques, des travaux de l'ingénierie et de presque tous les autres aspects de la vie quotidienne. Le moindre clic, le moins de téléchargement, un appel passé ou un SMS envoyé suffisent à collecter des pétaoctets de données, provoquant une explosion quantitative de données numériques qui a obligé les chercheurs à trouver de nouvelles méthodes de voir et d'analyser ces données. Il s'agit de découvrir de nouveaux ordres de grandeur pour la capture, la recherche, le partage, le stockage, l'analyse et la présentation de données. Ainsi est né le "Big Data". Il s'agit d'un concept permettant de stocker une quantité indescriptible d'informations sur une base numérique.

L'arrivée des technologies Big data dans le monde professionnel est perçus comme une bonne nouvelle pour les entreprises génératrices de grand volume de données sous forme de services Web contenant notamment des transactions de vente, des enregistrements boursiers, des descriptions de services, des promotions des ventes, des profils d'entreprise et des clients, ainsi que des commentaires des clients. La collection de ces informations non structurées, sans les traiter, revient à verser de l'eau dans la mer. Par conséquent, l'analyse de cette grande masse de données devient un besoin très important. C'est pourquoi des firmes comme Google ont décidé de transformer ces gigantesques bases d'informations en de vraies mines à trésors.

L'analyse du Big Data est le processus qui consiste à examiner des ensembles de données volumineux contenant des types de données hétérogènes pour découvrir des schémas cachés, des corrélations inconnues, les tendances du marché, les préférences des utilisateurs et d'autres informations exploitables. Les résultats analytiques peuvent apporter un gain d'efficacité du marketing, de nouvelles opportunités de recettes, un meilleur service clients, une amélioration de l'efficacité opérationnelle, des avantages concurrentiels sur les entreprises rivales et d'autres bénéfices métier.

L'analyse Big Data a pour principal objectif d'aider les entreprises à prendre des décisions plus avisées en permettant aux data-scientists, spécialistes en modélisation prédictive et autres professionnels du domaine à analyser de grands volumes de données transactionnelles, ainsi que d'autres formes de données encore inexploitées par les programmes conventionnels d'informatique décisionnelle (Business Intelligence) [13].

II.2 Les Techniques d'analyse de grandes masses de données

II.2.1 Définition

Le Big Data peut être très utile pour les entreprises. La plupart des organisations multinationales s'en remettent de nos jours à l'analyse de données pour aiguiller leurs décisions et ainsi stimuler leur croissance, augmenter leur chiffre d'affaires ou découvrir de nouvelles opportunités. Pour profiter des bienfaits de l'analyse de données, il convient de connaître les algorithmes à utiliser. Dans cette section nous allons présenter quelques techniques les plus utilisés pour l'analyse de grandes masses de données.

II.2.2 Apprentissage des Règles d'Associations

Cette méthode vise à identifier des ensembles cohérents dans un Dataset spécifique. Dans le domaine du e-commerce, cette méthode de datamining est appliquée afin de découvrir les corrélations entre différents produits dans des types de paniers. Par exemple : « si le produit *A* est acheté, il y aura un intérêt pour le produit *B* ». Cette technique permet donc d'effectuer de manière pertinente des recommandations de produits auprès des visiteurs d'un site. [14]

II.2.3 Analyse en composantes principales (ACP)

Pour l'analyse multi variée des données, l'analyse en composantes principales (ou ACP) est une méthode visant à modifier des variables liées (ayant des corrélations statistiques) en nouvelles variables synthétiques appelées « composantes principales ». L'analyse en composantes principales est utilisée pour tout type d'applications (ex : biologie computationnelle, finance quantitative, traitement d'image) et dans tous les domaines.

Elle est d'une part très utile à des fins de visualisation à titre d'exemples, on peut citer : la décorrélation entre variables, en vue de réduire le nombre de variables à mesurer, ou dans un espace 2 ou 3D, elle aide à identifier des sections ou groupes d'observations homogènes voire aberrantes. Mais aussi à des fins de modélisation par exemple : l'identification de facteurs non corrélés (en tant que combinaisons linéaires des variables de départ), utiles pour faire de la classification/prédiction telle que la régression linéaire, la régression logistique ou l'analyse discriminante [15].

II.2.4 Apprentissage Automatique

L'apprentissage automatique est un sous-domaine de l'intelligence artificielle (IA). En général, l'objectif de l'apprentissage automatique est de comprendre la structure des données et de les intégrer dans des modèles qui peuvent être compris et utilisés par tout le monde.
Bien que l'apprentissage automatique soit un domaine de l'informatique, il diffère des approches informatiques traditionnelles. En effet dans cette dernière, les algorithmes sont des ensembles d'instructions explicitement programmées utilisées par les ordinateurs pour calculer ou résoudre des problèmes.

Les algorithmes d'apprentissage automatique permettent aux ordinateurs de s'entraîner sur les entrées de données et utilisent l'analyse statistique pour produire des valeurs qui se situent dans une plage spécifique. Pour cette raison, l'apprentissage automatique facilite l'utilisation des ordinateurs dans la construction de modèles à partir de données d'échantillonnage afin d'automatiser les processus de prise de décision en fonction des données saisies.

Tout utilisateur des plus récentes technologies bénéficie de l'apprentissage automatique. La technologie de reconnaissance faciale par exemple permet aux plateformes de médias sociaux d'aider les utilisateurs à marquer et partager des photos d'amis. La technologie de reconnaissance optique des caractères (OCR) convertit les images du texte en caractères mobiles. Les moteurs de recommandation, alimentés par l'apprentissage automatique, suggèrent les films ou émissions de télévision à regarder en fonction des préférences de l'utilisateur. Les voitures autonomes qui utiliseront l'apprentissage automatique pour naviguer seront bientôt disponibles pour les consommateurs.

L'apprentissage automatique est un domaine en développement continu. Pour cette raison, on doit tenir compte de certaines considérations lorsque on travaille avec des technologies d'apprentissage automatique ou analysez l'impact des processus d'apprentissage automatique. La figure II.1 illustre les différentes catégories d'apprentissage automatique : l'apprentissage supervisé, non supervisé et par renforcement [16].

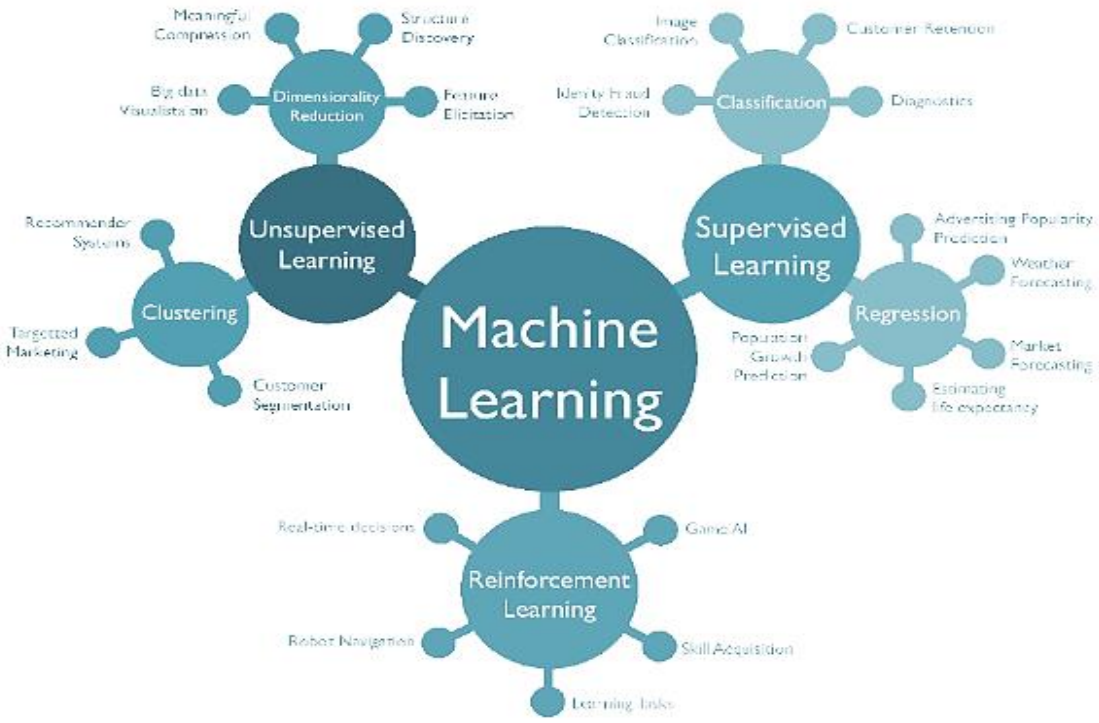


Figure II.1 – *Les différentes catégories d'apprentissage automatique* [17].

II.2.4.1 L'apprentissage Supervisé

Dans l'apprentissage supervisé, l'ordinateur est fourni avec des exemples d'entrées qui sont étiquetés avec les sorties souhaitées. Le but de cette méthode est que l'algorithme puisse « apprendre » en comparant sa sortie réelle avec les sorties « enseignées » pour trouver des erreurs et modifier le modèle en conséquence. L'apprentissage supervisé utilise donc des modèles pour prédire les valeurs d'étiquettes sur des données non étiquetées supplémentaires.

Par exemple, avec un apprentissage supervisé, un algorithme peut être alimenté avec des images de requins étiquetés Poisson des images d'océans étiquetés comme Océan. En étant formé sur ces données, l'algorithme d'apprentissage supervisé devrait être capable d'identifier plus tard des images de requin non marquées comme Poisson des images océaniques non étiquetées Océan. Un cas d'utilisation de l'apprentissage supervisé consiste à utiliser des données historiques pour prédire des événements futurs statistiquement probables. Il peut utiliser les informations historiques sur les marchés boursiers pour anticiper les fluctuations à venir ou être utilisé pour filtrer les courriers indésirables. Dans l'apprentissage supervisé, des photos étiquetées de chiens peuvent être utilisées comme données d'entrée pour classer les photos non marquées de chiens [16].

Les problèmes d'apprentissage supervisé sont catégorisés en problèmes de « régression » et de « classification ».

- **La Régression Linéaire Simple et Multiple :**

La régression est un ensemble de méthodes statistiques. Ce modèle vise à expliquer une variable aléatoire à l'aide de différentes variables non aléatoires. Le modèle de régression le plus connu est le modèle de régression linéaire, qui est l'algorithme le plus basique et l'un des plus utilisés dans le domaine de l'analyse de données et du Machine Learning [18]. La régression linéaire est une relation stochastique entre une ou plusieurs variables. Elle est appliquée dans plusieurs domaines, tels que la physique, la biologie, la chimie, l'économie…etc.

La régression linéaire explique une variable endogène par une seule variable exogène. A titre d'exemple, on peut citer : la relation entre la variable Prix et la variable Demande. Il s'agit de la régression linéaire simple. Dans un deuxième temps, la régression linéaire multiple représente la relation linéaire entre une variable endogène et plusieurs variables exogènes. Autrement dit, il s'agit de régresser linéairement une grandeur économique (variable à expliquer) sur plusieurs variables explicatives (variables exogènes). Par exemple, d'après la théorie économique, la demande d'un produit peut être expliquée par les grandeurs Prix, Revenu et Publicité [19].

- **Classification :**

La classification est un type d'apprentissage automatique supervisé dont L'objectif est principalement de définir des règles permettant de classer des objets dans des classes à partir de variables qualitatives ou quantitatives caractérisant ces objets. Les méthodes s'étendent souvent à des variables *Y* quantitatives. On dispose au départ d'un échantillon dit d'apprentissage dont le classement est connu. Cet échantillon est utilisé pour l'apprentissage des règles de classement. Il est nécessaire d'étudier la fiabilité de ces règles pour les comparer et les appliquer, évaluer les cas de sous apprentissage ou de sur apprentissage (complexité du modèle). On utilise souvent un deuxième échantillon indépendant, dit de validation ou de test.

La classification est appliquée souvent pour la reconnaissance de formes à titre d'exemples : la reconnaissance de chiffres manuscrits (codes postaux), la reconnaissance de visages, comme elle est très utile pour la catégorisation de textes

Par exemple : la classification d'e-mails, la classification de pages web. Les entrées de ces modèles sont des documents (texte ou html), tandis que la sortie sera une catégorie (thème, spam/non-spam) [20].

II.2.4.2 L'apprentissage non supervisé

Dans l'apprentissage non supervisé, les données sont non étiquetées, de sorte que l'algorithme d'apprentissage trouve tout seul des points communs parmi ses données d'entrée. Les données non étiquetées étant plus abondantes que les données étiquetées, les méthodes d'apprentissage automatique qui facilitent l'apprentissage non supervisé sont particulièrement utiles.

L'objectif de l'apprentissage non supervisé peut être aussi simple que de découvrir des modèles cachés dans un ensemble de données, mais il peut aussi avoir un objectif d'apprentissage des caractéristiques, qui permet à la machine intelligente de découvrir automatiquement les représentations nécessaires pour classer les données brutes.

L'apprentissage non supervisé est couramment utilisé pour les données transactionnelles. On peut avoir un grand ensemble de données sur les clients et leurs achats, mais en tant qu'être humain, on ne sera probablement pas en mesure de comprendre quels attributs similaires peuvent être tirés des profils de clients et de leurs types d'achats. Avec ces données introduites dans un algorithme d'apprentissage non supervisé, on peut déterminer que les femmes d'une certaine tranche d'âge qui achètent des savons non parfumés sont susceptibles d'être enceintes et donc une campagne de marketing liée à la grossesse et aux produits pour bébés pour augmenter leur nombre d'achats.

Sans une réponse « correcte », les méthodes d'apprentissage non supervisées peuvent examiner des données complexes, plus expansives et apparemment sans point commun, afin de les organiser de manière potentiellement significative. L'apprentissage non supervisé est souvent utilisé pour la détection d'anomalies, y compris pour les achats frauduleux de cartes de crédit et les systèmes de recommandation qui conseille sur les produits à acheter ensuite. Dans l'apprentissage non supervisé, les photos non marquées des chiens peuvent être utilisées comme données d'entrée pour l'algorithme afin de trouver des similitudes et de classer les photos de chiens ensemble [16].

Les algorithmes les plus connus dans l'apprentissage non-supervisé sont le « Clustering » et la « réduction de Dimensions »

- **Clustering**

L'analyse de cluster implique l'application d'un ou plusieurs algorithmes de clustering avec pour objectif de trouver les modèles ou les groupements cachés dans un Dataset. Les algorithmes de clustering permettent de former des groupements ou des clusters de manière à ce que les données d'un cluster possèdent une mesure de similarité plus élevée que les données de n'importe quel autre cluster.

L'analyse de cluster est utilisée en bio-informatique pour les analyses de séquences et les regroupements génétiques, en Data Mining pour l'extraction de séquences et de modèles, en imagerie médicale pour les segmentations d'images et en vision par ordinateur pour la reconnaissance d'objets [21].

- **Réduction de Dimensions**

La réduction de la dimension est un processus qui permet de réduire le volume d'informations à traiter et faciliter le processus de l'apprentissage. Elle tire avantage de la variance des variables plus précisément de la corrélation entre elles en éliminant la redondance de l'information contenue dans les variables. Nous pouvons classer toutes les techniques
Mathématiques de réduction des dimensions en deux grandes catégories :

- La sélection de variables : qui consiste à choisir des caractéristiques dans l'espace de mesure.
- L'extraction de traits : qui vise à sélectionner des caractéristiques dans un espace transformé (dans un espace de projection). [22]

La raison pour laquelle une telle opération est utile est que les données de plus petites dimensions peuvent être traitées plus rapidement. Cette opération est cruciale en apprentissage automatique par exemple, pour lutter contre le fléau de la dimension [23].

II.2.4.3 Apprentissage Par Renforcement

L'apprentissage par renforcement diffère fondamentalement des problèmes supervisés et non supervisés par ce côté interactif et itératif : l'algorithme essaie plusieurs solutions (on parle « d'exploration »), observe la réaction de l'environnement et adapte son comportement pour trouver la meilleure stratégie. Un des concepts clés de ce type de problèmes est l'équilibre entre ces phases d'exploration et d'exploitation. Cette méthode est particulièrement adaptée aux problèmes nécessitant un compromis entre la quête de récompenses à court terme et celle de récompenses à long terme. Parmi les exemples de problèmes traités de cette façon, on peut évoquer : apprendre à un robot à marcher en terrain difficile, à conduire (cas de la voiture autonome) ou à accomplir une tâche spécifique (comme jouer au jeu de go), piloter un agent à travers un labyrinthe. Les principales familles de problèmes d'apprentissage par renforcement sont les algorithmes de bandits, les problèmes de décisions (partiellement) markovien et les arbres de jeu [24].

Dans ce qui suit nous allons présenter Le Deep Learning ou apprentissage profond qui est un type d'intelligence artificielle dérivé du Machine Learning (apprentissage automatique) où la machine est capable d'apprendre par elle-même, contrairement à la programmation où elle se contente d'exécuter à la lettre des règles prédéterminées.

II.2.5 Deep Learning

II.2.5.1 Définition

L'apprentissage profond, ou « Deep Learning », est un aspect de l'intelligence artificielle (IA) qui permet aux ordinateurs d'apprendre de l'expérience et de comprendre le monde en termes de hiérarchie de concepts. Étant donné que l'ordinateur tire des connaissances de son expérience, un opérateur humain n'a pas besoin de spécifier formellement toutes les connaissances dont l'ordinateur a besoin. La hiérarchie des concepts permet à l'ordinateur d'apprendre des concepts complexes en les construisant à partir de concepts plus simples. Un graphe de ces hiérarchies aurait plusieurs couches de profondeur [26]. Le qualificatif « Deep » (profond) s'explique par le nombre de couches de traitement par lesquelles les données doivent passer [25].

Sous sa forme la plus simple, le Deep Learning peut être considéré comme un moyen d'automatiser l'analytique prédictive. Alors que les algorithmes traditionnels de l'apprentissage automatique sont linéaires, ceux du Deep Learning sont empilés dans une architecture d'une complexité et d'une abstraction croissantes [25]. La figure II.2 illustre la relation entre ces différentes disciplines de l'IA. Elle montre comment l'apprentissage profond est une sorte d'apprentissage automatique, qui est utilisé pour de nombreuses approches d'IA.

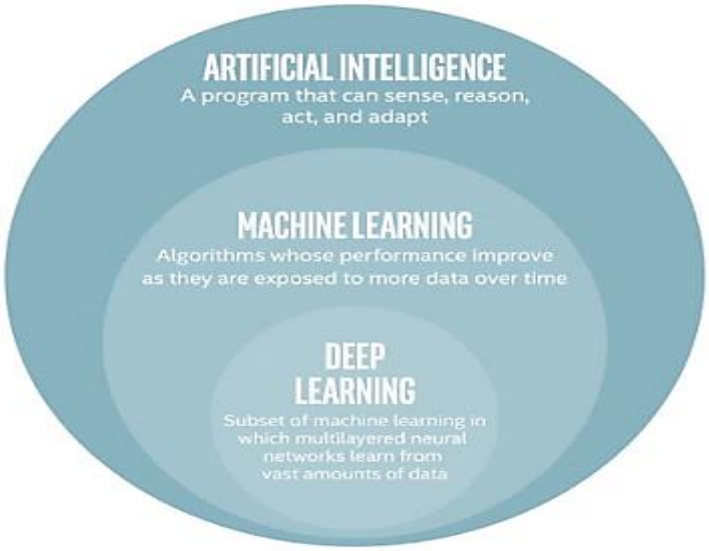


Figure II.2 – *La Relation Entre les Disciplines de L'IA [17].*

Parmi les algorithmes d'apprentissage automatique actuellement utilisés et développés, l'apprentissage profond absorbe le plus de données et a été capable de battre les humains dans certaines tâches cognitives. En raison de ces attributs, l'apprentissage profond est devenu l'approche avec un potentiel significatif dans le monde de l'intelligence artificielle. La reconnaissance faciale par ordinateur et la reconnaissance vocale ont toutes les deux permis de réaliser des progrès significatifs grâce à des approches d'apprentissage approfondies. IBM Watson[2] est un exemple bien connu d'un système qui exploite l'apprentissage profond [16].

Le principe de fonctionnement du Deep Learning est que chacun des algorithmes de l'architecture applique une transformation non linéaire aux données en entrée et utilise ce qu'il apprend pour créer un modèle statistique en sortie. Ces algorithmes peuvent être supervisés et servir à classer les données, ou non supervisés et à effectuer une analyse de modèle. Les itérations se poursuivent jusqu'à ce que la sortie ait atteint un niveau de précision acceptable.

Pour que le niveau de précision soit acceptable, les programmes de Deep Learning doivent avoir accès à des quantités phénoménales de données d'apprentissage et de puissance de traitement, deux conditions qui étaient difficiles à remplir pour les programmeurs avant l'avènement du Big Data, Cloud Computing et des réseaux sociaux.

Les programmes de Deep Learning peuvent élaborer des modèles prédictifs précis à partir de grandes quantités de données non structurées et sans étiquette car la plupart des données que les humains et les machines créent ne sont ni structurées, ni étiquetées. Les cas d'utilisation du Deep Learning sont tous les types d'applications d'analytique du Big Data, en particulier celles qui sont axées sur le traitement du langage naturel, la traduction, le diagnostic médical, les signaux boursiers, la sécurité des réseaux et l'identification des images [25]. Le Deep Learning est capable de reconnaître des visages, de synthétiser des textes ou encore de conduire une voiture autonome grâce à une architecture de réseaux de neurones artificiels (profonds), c'est-à-dire un ensemble de neurones (ce sont de petites calculatrices qui effectuent une opération mathématique) qui s'envoient des nombres en fonction de leurs liaisons [26], jusqu'à des neurones de sortie Pour illustrer ce point, nous allons présenter, dans ce qui suit, les réseaux de neurones artificiels, ainsi que quelques types et modèles de ces réseaux.

[2] IBM Watson : Watson est un supercalculateur IBM qui combine l'intelligence artificielle (AI) et un logiciel d'analyse sophistiqué pour des performances optimales en tant que répondeur "à questions". Le superordinateur porte le nom du fondateur d'IBM, Thomas J. Watson.

II.2.6 Les réseaux de neurones artificiels

II.2.6.1 Définition

En informatique, un réseau de neurones est un système de matériel et / ou de logiciel basé sur le fonctionnement de neurones dans le cerveau humain. Les réseaux de neurones - également appelés réseaux de neurones artificiels (RNA) - constituent une variété de technologies d'apprentissage en profondeur, qui relèvent également de l'intelligence artificielle.

Les applications commerciales de ces technologies sont généralement axées sur la résolution de problèmes complexes de traitement du signal ou de reconnaissance de formes. Les exemples d'applications commerciales importantes depuis 2000 comprennent la reconnaissance de l'écriture manuscrite pour le traitement des chèques, la transcription parole-texte, l'analyse des données d'exploration pétrolière, la prévision météorologique et la reconnaissance faciale [27].

II.2.6.2 Fonctionnement des réseaux de neurones artificiels

Un réseau de neurones implique généralement un grand nombre de processeurs fonctionnant en parallèle et disposés en niveaux. Le premier niveau reçoit les informations d'entrée brutes - analogues aux nerfs optiques dans le traitement visuel humain. Chaque niveau successif reçoit la sortie du niveau le précédent, plutôt que de l'entrée brute - de la même manière que les neurones plus éloignés du nerf optique reçoivent des signaux de ceux qui sont plus proches de lui. Le dernier niveau produit la sortie du système.

Chaque nœud de traitement a sa propre petite sphère de connaissances, y compris ce qu'il a vu et toutes les règles pour lesquelles il a été initialement programmé ou développé. Les niveaux sont fortement interconnectés, ce qui signifie que chaque nœud du niveau n sera connecté à de nombreux nœuds du niveau $n-1$ - ses entrées - et au niveau $n+1$, qui fournit une entrée pour ces nœuds. Il peut y avoir un ou plusieurs nœuds dans la couche en sortie, à partir desquels la réponse produite peut être lue [27].

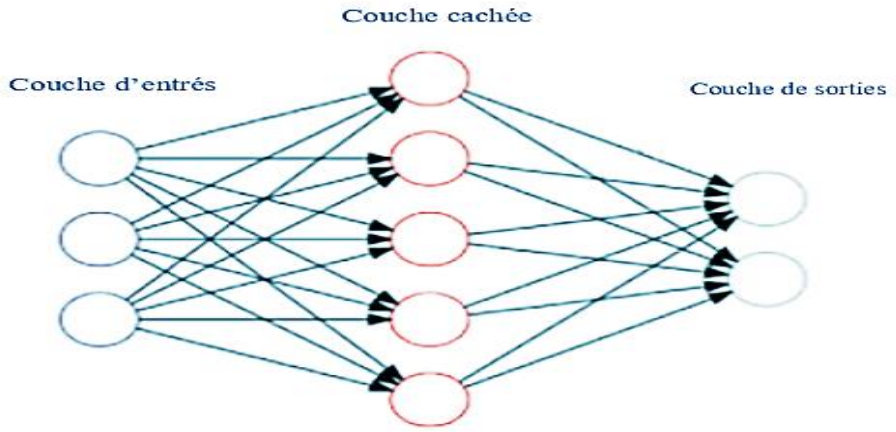


Figure II.3 – *Le schéma d'un RNA [27].*

Les réseaux de neurones fonctionnent en répartissant les valeurs des variables dans des automates (les neurones). Ces unités sont chargées de combiner entre elles leurs informations pour déterminer la valeur du paramètre de discrimination. C'est de la connexion de ces unités entre elles qu'émerge la capacité de discrimination du RNA. Chaque neurone reçoit des informations numériques en provenance de neurones voisins ; à chacune de ces valeurs est associée un poids représentatif de la force de la connexion. Chaque neurone effectue localement un calcul dont le résultat est transmis ensuite aux neurones avals [28].

Les réseaux de neurones se caractérisent par leur capacité d'adaptation, ce qui signifie qu'ils se modifient eux-mêmes au fur et à mesure qu'ils apprennent de la formation initiale et que les parcours ultérieurs fournissent davantage d'informations sur le monde. Le modèle d'apprentissage le plus fondamental est centré sur la pondération des flux d'entrée, c'est-à-dire la manière dont chaque nœud pondère l'importance des entrées de chacun de ses prédécesseurs. Les intrants qui contribuent à obtenir les bonnes réponses sont pondérés plus haut [27].

II.2.6.3 Les modèles des réseaux de neurones artificiels

II.2.6.3.1 Les réseaux Deep Feedforward

Les réseaux Deep Feedforward ou les perceptrons multicouches connus constituent le fondement de la plupart des modèles d'apprentissage en profondeur. Les réseaux tels que les réseaux CNN et RNN ne sont que quelques cas particuliers des réseaux Feedforward. Ces réseaux sont principalement utilisés pour des tâches d'apprentissage automatique supervisé pour lesquelles nous connaissons déjà la fonction cible, c'est-à-dire le résultat que nous souhaitons obtenir de notre réseau. Ils sont extrêmement importants pour la pratique de l'apprentissage automatique et constituent la base de nombreuses applications commerciales, telles que la vision par ordinateur et le traitement du langage naturel (PNL). Ont été fortement affectés par la présence de ces réseaux.

L'objectif principal d'un réseau à action directe est d'approcher une fonction f^*. Par exemple, une fonction de régression $y = f^*(x)$ mappe une entrée x sur une valeur y. Un réseau à anticipation définit une correspondance $y = f(x; \theta)$ et apprend la valeur des paramètres θ qui donnent la meilleure approximation de fonction.
La raison pour laquelle ces réseaux sont appelés Feedforward est que le flux d'informations se fait dans le sens aller, car x est utilisé pour calculer une fonction intermédiaire dans la couche masquée, laquelle sert à son tour à calculer y. En cela, si nous ajoutons le retour de la dernière couche masquée à la première couche masquée, cela représenterait un réseau de neurones récurrent.

Les couches entre la couche d'entrée et les couches de sortie sont appelées couches cachées, car les données d'apprentissage ne montrent pas la sortie souhaitée pour ces couches. Un réseau peut contenir un nombre quelconque de couches cachées avec un nombre quelconque d'unités cachées. Une unité ressemble en gros à un neurone qui prend en entrée les unités des couches précédentes et calcule sa propre valeur d'activation [29]. La figure suivante montre l'architecture d'un réseau Deep Feedforward avec trois couche cachées.

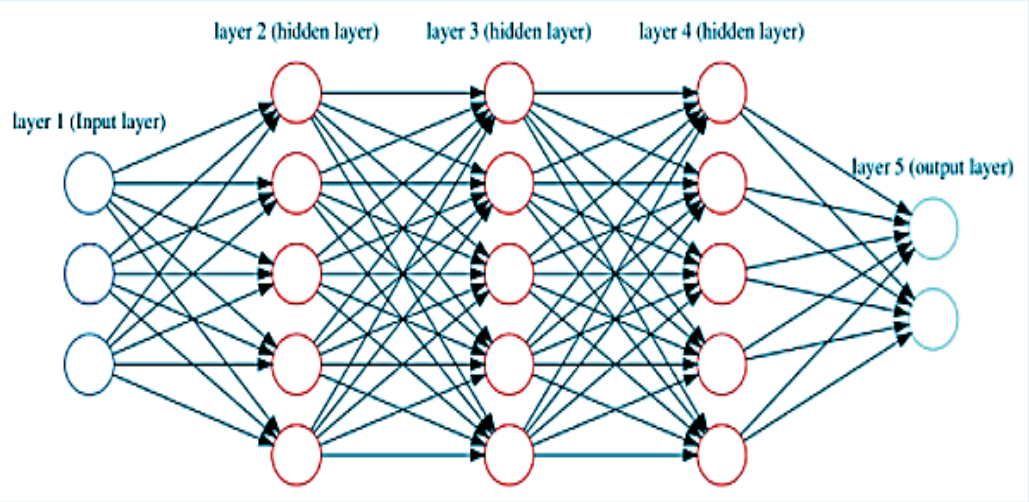


Figure II.4 – *Un réseau Deep Feedforward avec trois couches cachées [30].*

II.2.6.3.1 L'architecture du réseau Deep Feedforward

L'architecture d'un réseau fait référence à la structure du réseau, c'est-à-dire au nombre de couches cachées et au nombre d'unités cachées dans chaque couche. Selon le théorème d'approximation universelle, un réseau prévisionnel avec une couche de sortie linéaire et au moins une couche cachée avec une fonction d'activation « écrasante » peut approximer toute fonction mesurable de Borel d'un espace à dimension finie à un autre avec la quantité d'erreur non nulle voulue fournie que le réseau dispose d'un nombre suffisant d'unités cachées. Ce théorème indique simplement que, quelle que soit la fonction que nous essayons d'apprendre, il existe toujours un MLP (Multi-Layer Perceptron) capable de représenter la fonction.

Nous savons maintenant qu'il y aura toujours un système MLP capable de résoudre notre problème, mais il n'existe pas de méthode spécifiée pour déterminer cette architecture. Personne ne peut dire que si nous utilisons un nombre n de couches avec un nombre M d'unités cachées, nous pourrons résoudre le problème posé. Trouver cette configuration sans méthode de hit and trial reste un domaine de recherche actif et ne peut pour l'instant être que fait par hit et méthode d'essai.

Il est difficile de trouver la bonne architecture, car nous aurons peut-être à essayer de nombreuses configurations différentes, mais même si nous avons la bonne architecture MLP, il se peut que celle-ci ne puisse toujours pas représenter la fonction cible. Cela est dû à deux raisons : l'algorithme d'optimisation peut ne pas être en mesure de trouver les valeurs correctes des paramètres correspondant à la fonction souhaitée et l'autre raison est que les algorithmes d'apprentissage peuvent choisir une fonction incorrecte en raison d'un sur-ajustement.

II.2.6.3.1.1 Fonctions d'Erreurs

L'atout majeur dans le bon déroulement d'un modèle d'apprentissage et la capacité de diminuer le taux d'erreur entre le résultat cible et ce que le modèle a donné comme sortie. Pour mesurer cette quantité d'erreur, plusieurs fonctions ont été proposées. Ce qui suit une brève explication sur certaines de ces fonctions en utilisant le principe de la plausibilité maximum (Maximum Likelihood) :

- **Squared Error** Squared Error est donnée par l'expression:

$$\sum_{i=1}^{n} (y - \hat{y})^2$$

  Devrait être utilisé pour les problèmes de régression. La couche de sortie dans ce cas aura une seule unité.

- **Binary Cross Entropy** Binary Cross Entropy est donnée par l'expression :

$$-\sum_{i=1}^{n} y_i \log(f\ (x_i\ , \theta)\ ) + (1 - y_i) \log(1 - f\ (x_i\ , \theta))$$

  Où θ est l'ensemble des paramètres finaux de modèle Binary Cross Entropy est la fonction de perte recommandée pour la classification binaire. Cette fonction de perte devrait généralement être utilisée lorsque le réseau de neurones est conçu pour prédire la probabilité du résultat. Dans ces cas la couche de sortie comporte un seul neurone qui a une valeur entre 0 et 1.

- **Cross Entropy** Cross Entropy est donnée par l'expression :

$$-\sum_{i=1}^{n} y_i \log\big(f\ (x_i\ , \theta)\big)$$

  Elle est la fonction de perte recommandée pour la multi-classification. Cette fonction de perte devrait généralement être utilisée avec le réseau de neurones et est conçu pour prédire la probabilité des résultats de chacune des classes. Dans ce cas,
  La couche de sortie a des unités Softmax (une pour chaque classe).

II.2.6.3.1.2 Fonctions d'Activation

Une fonction d'activation est une fonction mathématique qui transforme le signal entrant dans un neurone. Elle permet d'avoir un meilleur calcul de gradient pour trouver les paramètres minimisant la fonction de perte sur les données. Les fonctions d'activation les plus courantes sont :

- **La Fonction Sigmoïde** La fonction transforme l'entrée x comme suit :

$$y = \frac{1}{1 + e^{-x}}$$

Généralement cette fonction est utilisée dans la couche de sortie d'un réseau de neurones si la classe est binaire de préférence. L'output de cette unité peut modéliser une distribution de Bernouli sur la sortie y conditionnée sur les input X [31].

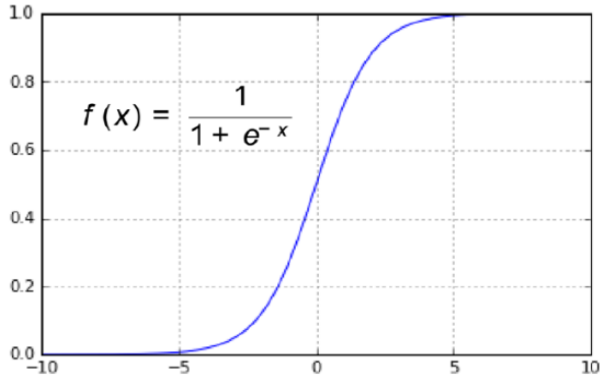


Figure II.5 – *Fonction Sigmoïde.*

- **Softmax** La couche Softmax est généralement utilisée comme couche de sortie pour les tâches de multi-classification en conjonction avec la fonction de perte Cross Entropy. La couche Softmax normalise les sorties de la précédente couche de sorte à des probabilités de somme égal à 1.
- **Rectified Linear Unit (ReLU)** ReLU est utilisée en conjonction avec une transformation linéaire transforme l'entrée x en :

$$f(x) = max(0;\ x)$$

Couramment utilisé comme une unité cachée ces derniers temps. Les résultats montrent que les unités ReLU conduisent à de grands et cohérents gradients, ce qui favorise l'apprentissage par gradient [19].

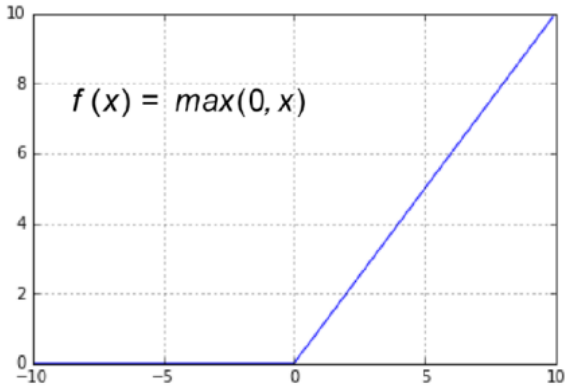


Figure II.6 – *La fonction ReLU.*

- **Tangente Hyperbolique** L'unité Tangente hyperbolique transforme l'input x comme suit : $f(x) = tanh(x)$. L'unité tangente hyperbolique est également couramment utilisée comme unité cachée [31].

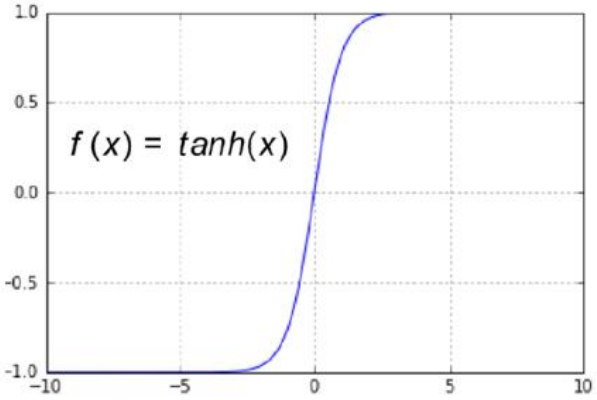


Figure II. 7 – *La fonction Tanh.*

II.2.6.3.1.3 La Descente de Gradient Stochastique

La décente de gradient stochastique (SGD) est une fonction qui sert à minimiser lors de l'apprentissage d'un modèle la fonction d'erreur $L(x)$. Par la mise à jour des paramètres associés au réseau. Il existe trois variantes pour la modification des paramètres :

- **Batch** : dans cette approche tout le Dataset est utilisé dans la phase de modification, ce qui est vraiment lourd en terme de calcule.
- **Stochastic Single Example** : dans cette approche un seul exemple est utilisé dans la phase de mise-à-jour choisi aléatoirement pour chaque itération. L'un des inconvénients de cette approche est que la direction mise à jour n'est pas aussi précise car la taille de la donnée d'apprentissage est vraiment petite [32].
- **Stochastic Mini-Batch** : Dans cette approche, un seul petit sous-ensemble d'exemples de Dataset est utilisé dans l'étape de mise à jour. Cette approche est la plus célèbre dans les travaux professionnels [33].

II.2.6.3.1.4 Challenges Pour Les SGD

L'implémentation de l'algorithme de la descente de gradient stochastique traditionnel dans le monde réel ne garantit pas une bonne convergence et il a fait face à des problèmes tels que

- **Minimum local :** Un défi majeur pour minimiser les fonctions d'erreur hautement non-convexes communes aux réseaux neurones est d'éviter d'être piégé dans leurs nombreux minimaux locaux sous-optimaux [34]. Un minimum local est une sorte de piège pour les approches classiques de descente de gradient qui empêche la procédure itérative de progresser vers la meilleure solution. Une meilleure solution consiste à trouver l'erreur minimale qu'on l'appelle le "Minimum Global".

- **Sélection du taux d'apprentissage** Le taux d'apprentissage est un paramètre qui contrôle la vitesse à laquelle les pondérations sont ajustées. Il a un grand impact sur la recherche de meilleures solutions au problème d'optimisation. Un taux d'apprentissage trop élevé peut faire rebondir la solution et empêcher la convergence vers l'erreur minimale. Par contre un taux d'apprentissage trop faible conduit à une convergence très lente [35]. Pour cela le problème de choisir un taux d'apprentissage idéal devient encore plus difficile. Pour remédier à ces problèmes et d'autres, plusieurs variantes de l'algorithme de la descente de gradient stochastique ont été proposés. Parmi ces travaux :

II.2.6.3.1.5 Algorithmes d'Optimisation de Descente de Gradient

Le choix de l'algorithme d'optimisation pour un modèle d'apprentissage profond (Deep Learning) peut faire la différence entre de bons résultats en minutes, heures et jours. Dans ce qui suit, nous allons présenter quelques algorithmes d'optimisation les plus célèbres qui sont utilisés pour traiter les problèmes susmentionnés.

- **Adagrad** met à l'échelle le taux d'apprentissage pour chaque paramètre en fonction de l'historique des gradients. Pour ce paramètre qui est essentiellement effectué en divisant le gradient actuel dans la règle de mise à jour par la somme des gradients précédents [36].
- **RMSprop** est un algorithme d'optimisation similaire à Adgard, où la seule différence entre eux est que le gradient actuel est calculé par la moyenne décroissante exponentiellement et non par la somme des gradients [37].
- **Adam** Adaptive Moment Estimation (Adam). Cet algorithme d'optimisation est une extension de la descente de gradient stochastique qui a récemment utilisé pour beaucoup d'applications d'apprentissage profond dans la vision par ordinateur et le traitement du langage naturel. Adam a été présenté par Diederik Kingma et Jimmy Ba [33]. Les auteurs décrivent l'algorithme Adam comme une combinaison des avantages des deux algorithmes précédents (Adagard et RMSprop). Adam est différent de la descente de gradient stochastique classique, Un algorithme d'optimisation de la descente de gradient stochastique maintient un seul taux d'apprentissage (Learning Rate) pour toutes les mises à jour des poids et ce taux ne change pas pendant l'apprentissage, par contre Adam calcule les taux d'apprentissage adaptatifs pour chaque paramètre. En plus il stocke une moyenne décroissante exponentiellement des gradients au carrées précédentes comme les algorithmes Adagard et RMSprop. Et également une autre moyenne décroissante exponentiellement des gradients passés comme l'algorithme de "Momentum" [38]. Pour ces avantages, Adam devient l'algorithme le plus optimisé et largement utilisé dans l'apprentissage de réseau de neurones. Comme indiqué dans le papier de Diederik et Jimmy [39].

II.2.6.3.1.6 Algorithme de Rétropropagation du Gradient (Back-propagation)

La Rétropropagation ou Back-propagation est une méthode utilisée pour calculer le gradient d'erreur qui est nécessaire pour modifier les poids d'un réseau de neurones.

L'algorithme de rétropropagation tente de minimiser l'erreur d'apprentissage calculée par la fonction d'erreur (Loss function) par modifier les poids de chaque neurone dans le réseau qui reçoivent initialement des valeurs aléatoires.

- D'abord le réseau de neurones reçoit une nouvelle observation $x = [x_1, .., x_n]$ et une classe cible y
- Le réseau de neurones applique l'algorithme Feedforward en propageant en avant (forward-propagation) dans les couches du réseau et calculer une nouvelle valeur pour chaque neurone. Par la formule : $x_{j=}\sigma\left(\sum w_i * x_i + b_j\right)$
- Lorsque la propagation vers l'avant est terminée, on obtient un nouveau résultat y^* à la sortie, on calcule donc l'erreur Δw entre la sortie y^* et la cible y en utilisant la fonction d'erreur (Loss function).
- Ensuite, l'algorithme de Back-propagation propage cette erreur vers l'arrière et modifie tous les poids des neurones couche par couche commençant par la sortie y^* pour que l'erreur d'apprentissage dans la prochaine propagation sera minimisée.

II.2.6.3.2 Autoencodeurs

Un *Autoencoder* est un réseau de : une couche d'entrée, une couche cachée (codage) et une couche de décodage. Le réseau est formé pour reconstruire ses entrées, ce qui oblige la couche cachée à essayer d'obtenir de bonnes représentations des entrées.

Un *Autoencoder* est un réseau de neurones formé pour tenter de copier son entrée dans sa sortie. Il comporte trois couches : une couche d'entrée, une couche de sortie et une couche cachée h en interne, qui décrit un code utilisé pour représenter l'entrée. Le réseau peut être considéré comme composé de deux parties : une fonction de codeur $h = f(x)$ et un décodeur qui produit une reconstruction $r = g(h)$.

Cette architecture est présentée dans la Figure II.3. Si un auto-codeur réussit simplement à apprendre à placer $g(f(x)) = x$ partout, cela n'est pas particulièrement utile. Au lieu de cela, les *Autoencodeurs* sont conçus pour être incapables d'apprendre à copier parfaitement. Ils sont généralement limités de manière à ne copier qu'approximativement et à ne copier que les entrées qui ressemblent aux données d'apprentissage. Parce que le modèle est obligé de hiérarchiser les aspects de l'entrée qui doivent être copiés, il apprend souvent des propriétés utiles des données [40].

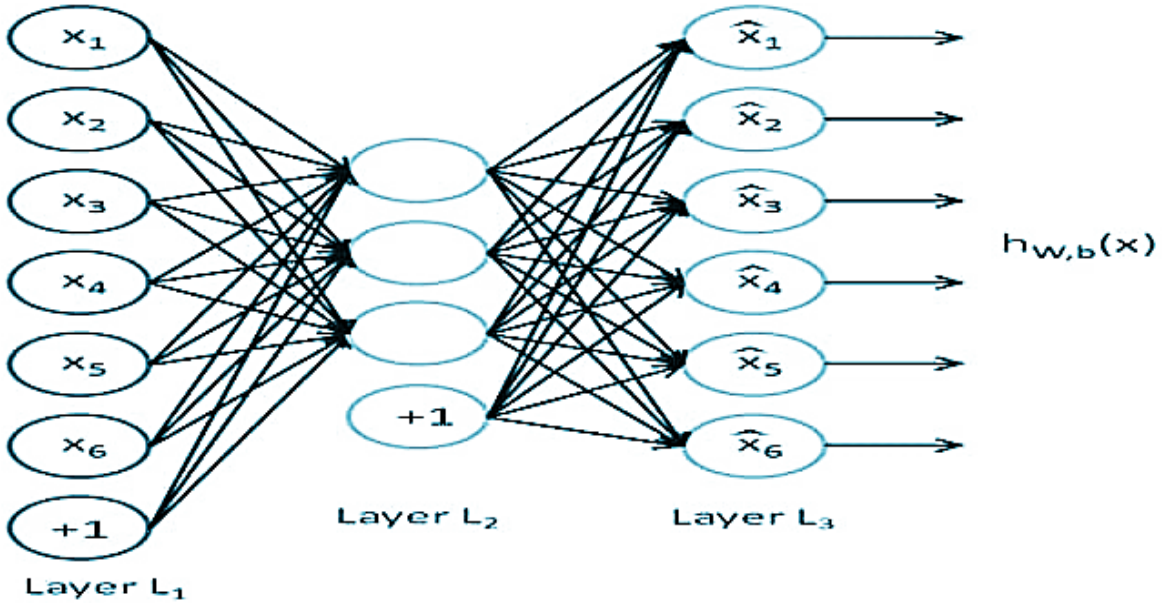


Figure II.8 – *Un Réseau de Neurones Autoencodeur [41].*

Traditionnellement, les auto-encodeurs étaient utilisés pour la réduction de la dimensionnalité ou l'apprentissage des fonctionnalités. Récemment, des connexions théoriques entre les auto-encodeurs et les modèles à variable latente ont amené les auto-encodeurs au premier plan de la modélisation générative. Les auto-encodeurs peuvent être considérés comme un cas particulier de réseaux à action directe et peuvent être formés à toutes les mêmes techniques, généralement la descente de gradient de mini-match suivant des gradients calculés par rétro-propagation. À la différence des réseaux à FeedForward généraux, les *Autoencodeurs* peuvent également être formés à la recirculation, un algorithme d'apprentissage basé sur la comparaison des activations du réseau sur l'entrée d'origine aux activations sur l'entrée reconstruite. La recirculation est considérée comme plus biologiquement plausible que la rétrodiffusion, mais elle est rarement utilisée pour des applications d'apprentissage automatique.

II.2.6.3.2.1 Undercomplete Autoencoders

Copier l'entrée sur la sortie peut sembler inutile, mais la sortie du décodeur ne nous intéresse généralement pas. Au lieu de cela, nous espérons que la formation de l'*Autoencodeur* à la tâche de copie des entrées permettra à h d'acquérir des propriétés utiles. Une façon d'obtenir des fonctionnalités utiles de l'auto-codeur consiste à contraindre h à avoir une dimension plus petite que x. Un *Autoencodeur* dont la dimension de code est inférieure à la dimension d'entrée est appelé sous-complétion. L'apprentissage d'une représentation incomplète oblige l'*Autoencodeur* à capturer les principales caractéristiques des données d'apprentissage.

Le processus d'apprentissage est simplement décrit comme minimisant une fonction de perte $L(x, g\ (f\ (x)))$, où L est une fonction de perte pénalisante $g\ (f\ (x))$ car elle est différente de x telle que l'erreur quadratique moyenne.

Lorsque le décodeur est linéaire et que L correspond à l'erreur quadratique moyenne, un *Autoencodeur* incomplet apprend à couvrir le même sous-espace que PCA. Dans ce cas, un encodeur automatique formé pour effectuer le travail de copie a appris le sous-espace principal des données d'apprentissage en tant qu'effet secondaire.

Les *Autoencodeurs* dotés de fonctions de codeur non linéaires f et de décodeurs non linéaires g peuvent ainsi apprendre une généralisation non linéaire plus puissante de PCA.

En cas de capacité excessive du codeur et du décodeur, l'*Autoencodeur* peut apprendre à effectuer la copie sans extraire d'informations utiles sur la distribution des données. Théoriquement, on pourrait imaginer qu'un *Autoencodeur* avec un code unidimensionnel mais un encodeur non linéaire très puissant pourrait apprendre à représenter chaque exemple d'apprentissage x (i) avec le code i. Le décodeur pourrait apprendre à mapper ces indices entiers aux valeurs d'exemples d'apprentissage spécifiques. Ce scénario spécifique ne se produit pas dans la pratique, mais il montre clairement qu'un auto-codeur formé à la tâche de copie peut ne rien apprendre d'utile sur le Dataset si la capacité de l'auto-codeur devient trop importante [40].

II.2.6.3.2.2 Les Autoencodeurs Régularisés (Regularized Autoencoders)

Les *Undercomplete Autoencoders*, dont la dimension de code est inférieure à la dimension d'entrée, peuvent connaître les principales caractéristiques de la distribution de données. Nous avons vu que ces auto-encodeurs n'apprennent rien d'utile si l'encodeur et le décodeur ont trop de capacité.

Un problème similaire se produit si le code masqué est autorisé à avoir une dimension égale à l'entrée et dans le cas de sur-complétion dans lequel le code masqué a une dimension supérieure à l'entrée. Dans ces cas, même un codeur linéaire et un décodeur linéaire peuvent apprendre à copier l'entrée sur la sortie sans rien apprendre d'utile sur la distribution des données.

Idéalement, on pourrait former n'importe quelle architecture d'*Autoencodeur* avec succès, en choisissant la dimension du code et la capacité du codeur et du décodeur en fonction de la complexité de la distribution à modéliser. Les auto-encodeurs régularisés permettent de le faire. Plutôt que de limiter la capacité du modèle en gardant le codeur et le décodeur peu profonds et la taille du code réduite, les *Autoencodeurs* régularisés utilisent une fonction de perte qui encourage le modèle à avoir d'autres propriétés que la possibilité de copier son entrée dans sa sortie. Ces autres propriétés incluent la faible densité de la représentation, la petite taille de la dérivée de la représentation et la robustesse au bruit ou aux entrées manquantes. Un *Autoecodeur* régularisé peut être non linéaire et excessivement complet, tout en apprenant quelque chose d'utile sur la distribution des données, même si la capacité du modèle est suffisamment grande pour apprendre une fonction d'identité triviale.

En outre les méthodes décrites ici, qui sont le plus naturellement interprétées comme des auto-encodeurs régularisés, presque tous les modèles génératifs à variables latentes et équipés d'une procédure d'inférence (pour le calcul des représentations latentes données en entrée) peuvent être considérés comme une forme particulière d'auto-encodeur. Naturellement, ces modèles apprennent les encodages d'entrée de grande capacité et surcomplets et ne nécessitent pas de régularisation pour que ces encodages soient utiles. Leurs codages sont naturellement utiles car les modèles ont été formés pour maximiser approximativement la probabilité des données d'apprentissage plutôt que pour copier l'entrée dans la sortie. Dans ce qui suit nous allons présenter les techniques principales utilisés dans la régularisation des *Autoencoders* [30].

- **Les Autoencodeurs épars (Sparse Autoencoders)**

Un *Autoencoder* épars (Sparse) est un *Autoencoder* qui représente de façon éparse les entrées. Cet éparpillement permet de forcer l'*Autoencoder* d'extraire des caractéristiques utiles sur les données.

Un *Autoencodeur* épars est simplement un *Autoencodeur* dont le critère d'apprentissage implique une pénalité de parcimonie $\Omega\,(h)$ sur la couche de code h, en plus de l'erreur de reconstruction :

$$L\,(x, g\,(f\,(x)))\,+\,\Omega\,(h),$$

où $g\,(h)$ est la sortie du décodeur et typiquement nous avons $h\,=\,f\,(x)$, la sortie du codeur.

Les auto-encodeurs épars sont généralement utilisés pour apprendre les fonctionnalités d'une autre tâche, telle que la classification. Un *Autoencodeur* régularisé pour être épar doit répondre à des caractéristiques statistiques uniques du Dataset sur lequel il a été formé, plutôt que d'agir simplement comme une fonction d'identité. De cette manière, une formation à la tâche de copie avec une pénalité de parcimonie peut générer un modèle ayant acquis des fonctionnalités utiles en tant que sous-produit.

- **Les Autoencodeurs Débruiteurs (Denoising Autoencoders)**

Plutôt que d'ajouter une pénalité Ω à la fonction de coût, on peut obtenir un *Autoencodeur* qui apprend quelque chose d'utile en modifiant le terme d'erreur de reconstruction de la fonction de coût. Traditionnellement, les *Autoencodeurs* minimisent certaines fonctions :

$$L(x, g(f(x))),$$

Où L est une fonction de perte pénalisant $g\,(f\,(x))$ parce qu'elle est différente de x, telle que la norme de leur différence. Cela les incite à apprendre à n'être qu'une simple fonction d'identité, s'ils en ont la capacité. Un *Autoencoder* (DAE) avec réduction de bruit minimise

$$L\left(x, g\left(f\left(x^{*}\right)\right)\right),$$

où x* est une copie de x qui a été corrompue par une forme de bruit. Les *Autoencodeurs* en débruitage doivent donc annuler cette corruption plutôt que de simplement copier leur entrée.

Le débruitage des forces d'entraînement f et g pour apprendre implicitement la structure de p données (x). Le codage automatique des *Autoencodeurs* fournit donc un autre exemple de la façon dont des propriétés utiles peuvent émerger en tant que sous-produit de la minimisation des erreurs de reconstruction. Ils constituent également un exemple de la façon dont des modèles surcomplétés et à haute capacité peuvent être utilisés comme *Autoencodeurs*, à condition de veiller à ne pas les apprendre à utiliser la fonction d'identité [40].

II.2.6.3.3 Les réseau de neurones récurrents (RNN)

II.2.6.3.3.1 Définition

Un réseau de neurones récurrents (RNN) est un type de réseau de neurones artificiel couramment utilisé dans la reconnaissance de la parole et le traitement du langage naturel. Les RNN sont conçus pour reconnaître les caractéristiques séquentielles d'une donnée et ses modèles d'utilisation pour prédire le prochain scénario probable. Les RNN sont utilisés dans l'apprentissage profond et dans le développement de modèles simulant l'activité des neurones dans le cerveau humain. Ils sont particulièrement puissants dans les cas d'utilisation dans lesquels le contexte est essentiel pour prédire un résultat et se distinguent des autres types de réseaux neuronaux artificiels car ils utilisent des boucles de rétroaction pour traiter une séquence de données qui informe le résultat final, qui peut également être une séquence de données.

Les cas d'utilisation de RNN ont tendance à être liés à des modèles de langage dans lesquels la connaissance de la lettre suivante d'un mot ou du mot suivant d'une phrase dépend des données qui le précèdent. L'écriture par RNN est une forme de créativité informatique. Cette simulation de la créativité humaine est rendue possible par la compréhension par l'intelligence artificielle de la grammaire et de la sémantique apprises lors de son apprentissage.

II.2.6.3.3.2 Déroulement d'un RNN

Un réseau de neurones récurrent se compose en séquence de valeur d'entrée X qui produisent une séquence de valeur en sortie Y tout en passant par une couche cachée H.
Le déroulement général d'un réseau de neurones récurent se résume dans les deux étapes suivantes :

1. Le réseau calcule l'état caché de la première entité de la séquence notée $h(t)$.

$$h^{(t)} = tangh(Ux^{(t)} + Wh^{(t-1)} + b)$$

Le calcule de $h(t)$ utilise la valeur d'entrée $x(t)$ et l'état caché précédente $h(t-1)$. Ce qui explique la différence du RNN par rapport aux autres réseaux feedforward.
Les poids associés aux valeurs d'entrée et celle de l'état caché précédent sont notés dans l'équation précédente par U et W respectivement. Et il y a aussi le biais noté par b. La fonction d'activation $tanh$ est utilisée pour le calcul de l'état caché.

2. Le réseau ensuite terminera par calculer la valeur de sortie notée $O(t)$.
$$O^{(t)} = softmax(V\, h^{(t)} + c)$$
3. La valeur de sortie est calculée en utilisant l'état caché $h(t)$. V est le poids associé à la valeur de l'état caché $h(t)$ et c est le biais. La fonction d'activation Softmax est utilisée pour retourner les probabilités associées aux valeurs de sortie. Alors l'apprentissage consiste à trouver les valeurs des paramètres V, U, W, c et b qui minimise le taux d'erreur L pour les valeurs de sortie. En utilisant comme un réseau de neurones classique la méthode de Backpropagation avec la décente du gradient stochastique par exemple [15].

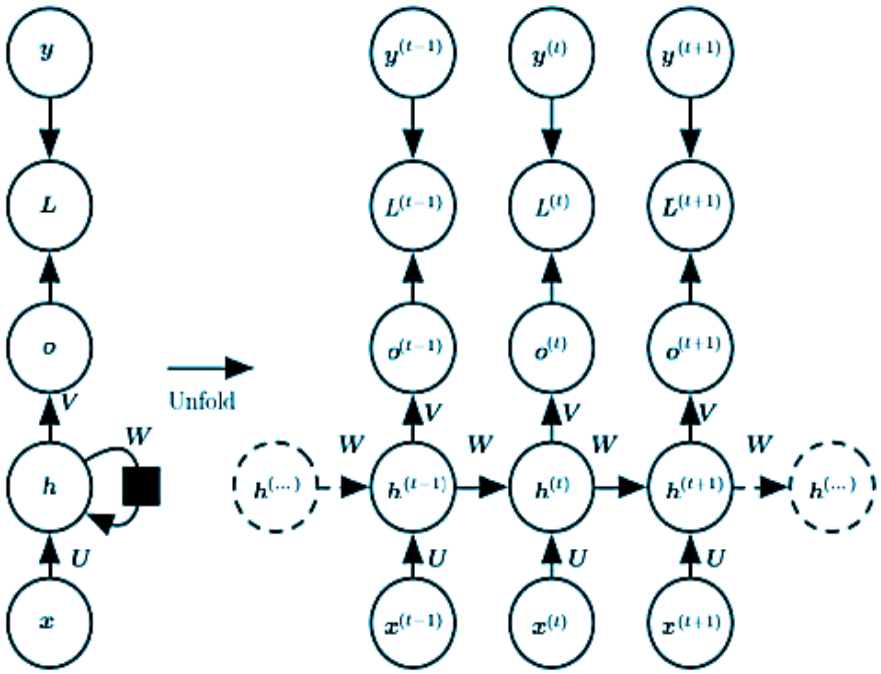


Figure II. 9 – *Déroulement des réseaux de neurones récurrents (RNN) [27].*

II.2.6.3.3.3 Réseaux forward, backward et bidirectionnels

Nous avons étudié le comportement de tous les réseaux décrits jusqu'ici en trois versions : Forward, Backward et bidirectionnel. La version Forward est celle que nous venons de détailler. La version Backward est équivalente, à la seule différence près qu'elle traite les séquences dans le sens inverse par rapport à la version Forward, c'est-à-dire de la fin vers le début. La version Backward permet donc de prédire l'étiquette à la position t d'une séquence, étant donnés les mêmes mots que la version Forward et les N étiquettes futures, ou les N états futurs de la couche cachée, ou les N étiquettes futures associées aux N états futurs de la couche cachée (dans notre seconde variante).

Les RNNs bidirectionnelle utilise en même temps l'information passée et future mémorisées par les deux réseaux Forward et Backward, quel que soit le type de réseau. Le fonctionnement de cette troisième version est donc forcément différent de celui des autres :

- D'abord le réseau Backward est utilisé pour prédire les étiquettes et/ou les états futur(e)s de la couche cachée, selon le type de réseau.
- Les étiquettes et/ou les états de la couche cachée sont initialisés avec des valeurs "fillers" pour la phase Forward.
- Un modèle global parcourt la séquence en avant en utilisant les étiquettes et/ou les états passés et futurs de la couche cachée.

Les RNN bidirectionnels sont basés sur l'idée que la sortie au temps t peut dépendre non seulement des éléments précédents de la séquence, mais également des éléments futurs. Par exemple, pour prédire un mot manquant dans une séquence, on souhaite examiner à la fois les contextes gauche et droit. Les RNN bidirectionnels sont assez simples. Ils ne sont que deux RNN empilés les uns sur les autres. La sortie est ensuite calculée en fonction de l'état caché des deux RNN.

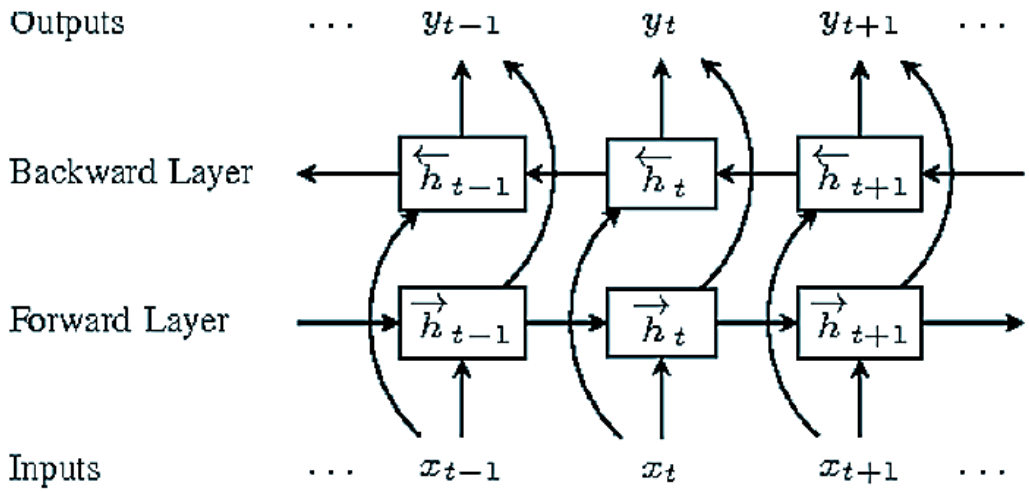


Figure II. 10 *– Architecture d'un RNN Bidirectionnel [44].*

II.2.6.3.3.4 Explosion et Disparition de Gradient

Un inconvénient pour les RNN standard est le problème du gradient en voie de disparition ou d'explosion, dans lequel la performance du réseau neuronal en souffre car il ne peut pas être entraîné correctement. Cela se produit avec les réseaux neuronaux profondément en couches, qui sont utilisés pour traiter des données complexes. Lors du déroulement d'un RNN, la valeur du gradient peut écourter jusqu'à ce qu'elle soit proche de zéro (Disparition). De même elle peut augmenter à une valeur très élevée (Explosion).

Parmi les techniques utilisées pour résoudre le problème d'explosion du gradient on trouve le Gradient Clipping qui consiste à définir une valeur de seuil. Si le gradient dépasse cette valeur lors de la phase d'apprentissage. Sa valeur sera mise à jour à une nouvelle valeur spécifique donnée.

Par contre la disparition du gradient est plus difficile à détecter. Pour cela une extension du modèle réseaux de neurones récurrent nommée Long Short-Term Memory est proposée afin de surpasser le risque de la disparition du gradient et autres limitations [28]. Les RNN sont construits avec des unités LSTM catégorisent les données dans des cellules mémoire à court et à long terme. Cela permet aux RNN de déterminer ce qui est important et doit être mémorisé et retransmis dans le réseau et quelles données peuvent être oubliées.

II.2.6.3.4 Long Short-Term Memory (LSTM)

II.2.6.3.4.1 Définition

Les réseaux de mémoire à court terme (LSTM) sont une extension des réseaux de neurones récurrents, ce qui étend leur mémoire. Par conséquent, il est bien adapté pour tirer parti d'expériences importantes qui ont très peu de temps entre les deux. Le réseau de neurones récurrent utilise des blocs de mémoire à court terme de longue durée pour fournir un contexte à la manière dont le programme reçoit les entrées et crée les sorties [45]. Le bloc de mémoire long terme est une unité complexe avec divers composants tels que des entrées pondérées, des fonctions d'activation, des entrées de blocs précédents et des sorties éventuelles.

L'unité est appelée bloc de mémoire à court terme car le programme utilise une structure basée sur des processus de mémoire à court terme pour créer une mémoire à plus long terme. Ces systèmes sont souvent utilisés, par exemple, dans le traitement du langage naturel. Le réseau de neurones récurrent utilise les blocs de mémoire à court terme pour prendre un mot ou un phonème particulier et l'évaluer dans le contexte des autres chaînes, où la mémoire peut être utile pour trier et catégoriser ces types d'entrées. En général, le LSTM est un concept accepté et commun dans les réseaux de neurones récurrents pionniers [46].

Les unités d'un LSTM sont utilisées comme unités de construction pour les couches d'un RNN, ce qui s'appelle alors souvent un réseau LSTM. Les LSTM permettent aux RNN de se souvenir de leurs entrées sur une longue période. En effet, les LSTM stockent leurs informations dans une mémoire, ce qui ressemble beaucoup à la mémoire d'un ordinateur car il peut lire, écrire et supprimer des informations de sa mémoire. Cette mémoire peut être vue comme une cellule fermée, où porte signifie que la cellule décide de stocker ou de supprimer des informations (par exemple, si elle ouvre ou non les portes), en fonction de l'importance qu'elle attribue à l'information. L'attribution d'importance se fait par le biais de pondérations, également apprises par l'algorithme. Cela signifie simplement qu'elle apprend avec le temps quelles informations sont importantes et lesquelles ne le sont pas.

Dans un LSTM, on a trois portes : une entrée, une porte cachée et une porte de sortie. Ces portes déterminent s'il faut ou non laisser une nouvelle entrée (entrée), supprimer l'information car elle n'a pas d'importance (oublier la porte) ou la laisser affecter la sortie au pas de temps actuel (porte de sortie).

Les portes d'un LSTM sont analogiques, sous la forme de sigmoïdes, ce qui signifie qu'elles vont de 0 à 1 [45]. Noté que chaque porte prend en entrée la valeur xt d'entrée et la sortie de l'état précédente $h(t-1)$. La figure II.8 illustre la cellule LSTM et son architecture.

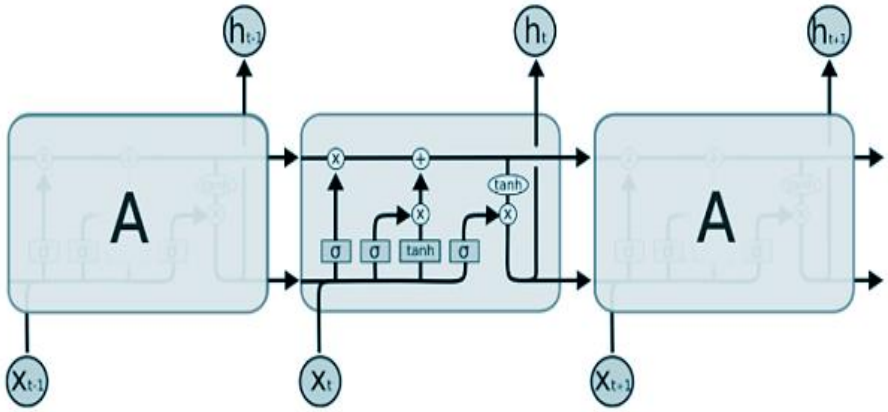
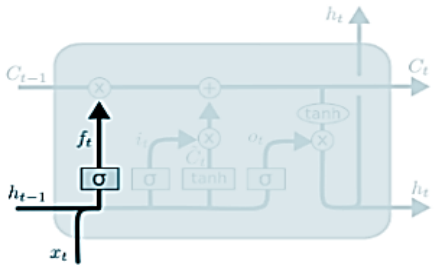


Figure II. 11 – *Le Module de Répétition Dans un LSTM Contient Quatre Couches Interactives [47].*

Au début, l'unité LSTM doit estimer l'importance des informations reçues par la cellule précédente $c(t-1)$. Cette chose est déterminée par La porte d'oubli f (forget gate), illustré dans la Figure II.13. L'idée est de sommer l'entrée $x(t)$, la sortie précédente $h(t-1)$ et le biais passant le résultat dans une fonction d'activation sigmoïde. Si la valeur obtenue est plus proche de 1, alors la cellule de mémoire sauvegarde l'information. Sinon si la valeur obtenue est plus proche de 0 alors l'information est considérée impertinente et la mémoire est formatée.

$$. \ f_t = \sigma_g(W_{fx_t} + U_f \, h_{t-1} + b_f)$$



$$f_t = \sigma\,(W_f \cdot [h_{t-1}, x_t] \; + \; b_f)$$

Figure II. 12 – *La Porte d'Oubli (Forget Gate) Dans la Cellule LSTM [47].*

La porte d'écriture i (Write gate) nous permet de déterminer les informations qu'on veut écrire dans la cellule de mémoire. Exactement comme la porte d'oubli, le calcul de la porte d'écriture utilise l'entrée xt et l'output précédent $h(t-1)$. Avec des poids différents associés à l'entrée X et l'output h. La Figure II.14 Suivantes montre l'architecture de cette porte.

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$

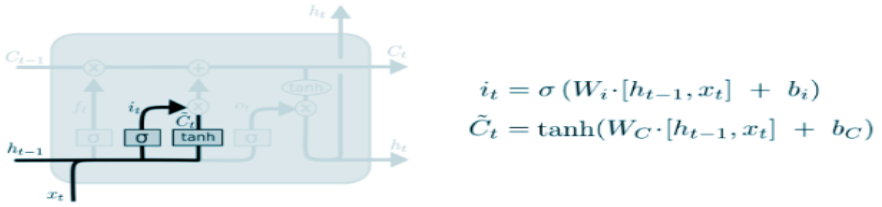


Figure II.13 – *La Porte d'Écriture (Write Gate) Dans la Cellule LSTM [47].*

Une fois la porte d'écriture et d'oubli sont calculés, alors on peut notamment créer notre cellule de mémoire ct. Le calcule de ct utilise notamment la cellule précédente $c(t-1)$, l'entrée $x(t)$ et l'output précédent $h(t)$. La figure II.15 explique la sortie de la porte d'écriture.

$$c_t = f_t * c_{t-1} + i_t * \sigma_g(W_c x_t + U_c h_{t-1} + b_c)$$

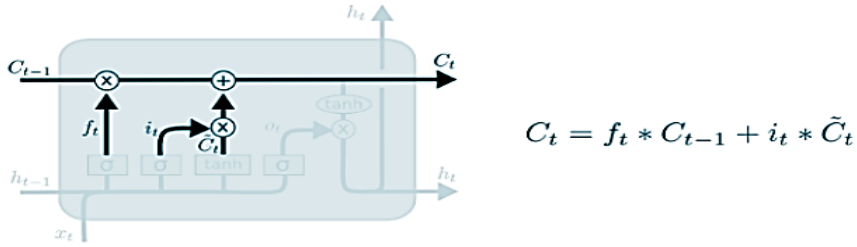


Figure II.14 – *La Sortie de la Porte d'Écriture [47].*

Finalement, pour chaque étape l'unité LSTM fourni une valeur de sortie, ce travail se fait à travers la porte d'output $o(t)$.comme illustre la Figure II.16 ci-dessous :

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$

La sortie $h(t)$ est ainsi calculée en multipliant la valeur de la porte d'output et la tangente hyperbolique de la cellule mémoire. $h_t = o_t * \sigma_h(c_t)$

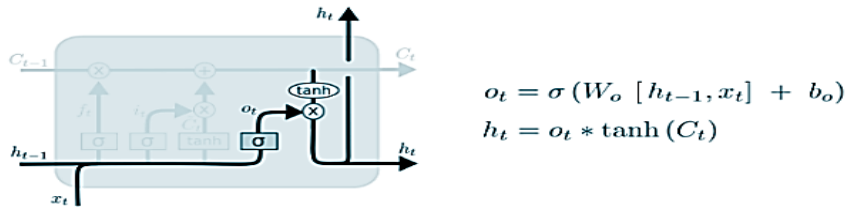


Figure II.15 – *La Porte de Sortie (Output Gate) d'une Cellule LSTM [47].*

II.3 Conclusion

L'arrivée des technologies Big Data dans le monde professionnel est perçue comme une bonne nouvelle pour les entreprises génératrices de grand volume de données qui avaient du mal à les traités. L'analyse de grande masse de données offre plusieurs techniques pour le traitement du Big Data. Nous nous sommes particulièrement focalisés sur l'apprentissage automatique, la technique révolutionnaire Deep Learning, les réseaux de neurones ainsi que les différents modèles des réseaux de neurones. Dans le chapitre suivant, nous introduisons l'étude conceptuelle et les étapes d'implémentation de notre approche de recommandation de services Web proposée et les différentes étapes de réalisation de notre solution proposée.

CHAPITRE III

CONCEPTION ET IMPLEMENTATION

Sommaire

III.1 Introduction

L'essor du Web dans ces dernières années et l'évolution massive du domaine e-commerce ont notamment contribué à la mise en place des systèmes d'aide à la recherche et l'exploitation des ressources. On parle des systèmes de recommandation.

Un système de recommandation a pour objectif de fournir à un utilisateur des ressources pertinentes en fonction de ses préférences. Ce dernier voit ainsi réduit son temps de recherche mais reçoit également des suggestions de services auxquels il n'aurait pas spontanément prêté attention et qui répondent pertinemment à son besoin. La recommandation peut être comparée à un bibliothécaire qui va pouvoir, en fonction des goûts d'un de ses clients et sa connaissance des différents ouvrages et les items de manière individuelle, proposer une liste d'ouvrages à ce dernier qui répond le mieux à ses intérêts. [48].

Les différents acteurs du Web qui développent ces outils de recommandations sont donc de plus en plus nombreux aujourd'hui. Parmi les leaders de ce domaine, le site web de vente en ligne Amazon.com qui vend pratiquement toutes les catégories de produits tels que les livres, les CDs, les logiciels, l'électronique ..., les utilisateurs de ce site reçoivent des suggestions personnalisées régulièrement dans leurs comptes comme la proposition des articles similaires, produits complémentaires et d'autres. On cite aussi Netflix.com qui propose une recommandation personnalisée des films aux utilisateurs après l'analyse des évaluations utilisateurs de ces films. Par ailleurs, il existe également des recommandations non personnalisées qui se basent sur l'analyse de l'historique de l'utilisation des différents produits en faisant abstraction des données des utilisateurs.

Ces dernières années ont témoigné du décuplement du nombre de services cloud et APIs déployées sur la toile. De ce fait, les utilisateurs doivent passer en revue un large espace de services avant de pouvoir sélectionner ceux qui répondent le mieux à leurs besoins. La tâche de sélection prend une dimension complexe lorsqu'il s'agit de découvrir des services interopérables qui peuvent être composés ou peuvent se substituer mutuellement. Plusieurs travaux de recherches ont introduit des systèmes de recommandation qui permettent aux utilisateurs de découvrir les services les plus pertinents à leurs besoins. Ces systèmes de recommandation se basent essentiellement sur les données des services offertes par leurs fournisseurs mais négligent l'aspect interactif des services Web. En effet, les services Web sont engagés dans plusieurs scénarios de collaboration et de compétition et offrent ainsi un historique d'interaction dont l'analyse par les techniques de l'IA va améliorer substantiellement la qualité de recommandation des services aux clients. Dans ce qui suit, nous proposons un nouveau modèle basé sur le Deep Learning pour la classification des services web en analysant leurs données diverses, telles que les descriptions de services et l'historique de leurs compositions.

III.2 Acquisition des données

Dans notre approche proposée consistant à recommander des services Web, nous avons utilisé des services Web réels issus du répertoire *programmableWeb.com*. Nous avons développé un outil grâce à la bibliothèque **NLTK** de **Python**, qui parcourt automatiquement tous les services (*Mashups* et atomic) publiés dans l'annuaire et obtient tous les attributs de service. Le répertoire indique si un service est un *Mashup* ou non. Dans le cas d'un service atomique, il répertorie tous les attributs du service Web et dans le cas d'un *Mashup*, il répertorie les services qui lui sont associés ainsi que certains autres attributs disponibles sur le répertoire.

À la fin du processus d'extraction des données, nous aurons deux (02) fichiers, « **API_Data** » qui contient *10630* services Web atomiques avec les attributs illustrés dans la Figure-III.1 et « **Mashup_Data** » qui contient *4211 Mashup* et quelques attributs comme l'illustre la Figure-III.2.

API Endpoint	http://api.linkedin.com/v1/
API Portal / Home Page	https://developer.linkedin.com/docs
Primary Category	Social
Secondary Categories	Enterprise
API Provider	LinkedIn
SSL Support	Yes
Twitter URL	https://twitter.com/LinkedIn
Support Email Address	atrachtenberg@linkedin.com
Developer Support URL	https://developer.linkedin.com/support
Interactive Console URL	https://developer.linkedin.com/rest-console
Authentication Model	OAuth 2
Version	1
Terms Of Service URL	https://developer.linkedin.com/documents/linkedin-apis-terms-use
Is the API Design/Description Non-Proprietary ?	No
Scope	Single purpose API
Device Specific	No
Docs Home Page URL	https://developer.linkedin.com/docs/rest-api
Architectural Style	REST
Supported Request Formats	JSON, URI Query String/CRUD, XML
Supported Response Formats	JSON, XML
Is This an Unofficial API?	No
Is This a Hypermedia API?	Yes
Restricted Access (Requires Provider Approval)	Yes

Figure III.1 – *Les attributs du Service Web LinkedIn.*

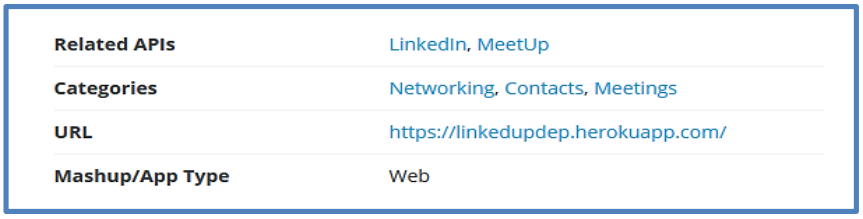


Figure III.2– *Les attributs du Mashup LinkedUp.*

III.3 Nettoyage et préparation des données

III.3.1 Nettoyage des données des services web

Le fichier « *API Data* » contient de nombreux attributs, nous avons gardé seulement les attributs les plus discriminants (c'est-à-dire les attributs qui peuvent être considérés comme des variables catégorielles) : « **Name, Info, Categories, Primary Category, Authentication, Device Specific, Hypermedia, Request Format, Response Format, Non proprietary, SSL Support, Restricted access, Scope, Style ».** En éliminant tous les autres attributs qui ont des valeurs uniques, pour qu'ils n'influencent pas l'apprentissage.

Après le nettoyage vertical du fichier « *API Data* », nous allons faire un nettoyage horizontal, l'algorithme suivant expliquera ce processus.

Algorithme 1: *Association des Mashups Ids aux services*

```
Services = listOfServices()
Nb _ Ser = Length(Services)          # La taille globale de la liste des services
Nb _ Mush = Length(Mushups)          # La taille globale de la liste des Mashups
R_APIs=Mushups(Related_APIs)         # l'ensemble de services web qui constituent  les Mashups
for each integer i in Nb _ Ser do
    service = Services(i)
    Name = getName(service)
    for each integer j in Nb _ Mush do
        if  Name in R_APIs[j] do
            IDMush_SW[i] = j
for each integer i in Nb _ Ser do
    if IDMush_SW[i] is empty do
        IDMush_SW[i]= -1
```

Description de l'algorithme 1

Au début, nous listons tous les noms des API, ainsi que leur numéro et les noms de tous les *Mashups*, un *Mashup* est une composition de services Web, nous pouvons trouver les API qui ont été composés pour ce *Mushup* dans l'attribut « **Related API** », ce que fait cet algorithme, c'est qu'il prend un nom d'un service web et le cherche dans l'attribut « **Related API** » de tous les *Mashups*. Une fois trouvé, il ajoute l'id de ce *Mashup* au service web concerné dans une nouvelle colonne intitulée « **Mashup ID** », s'il n'a trouvé le service web dans aucun des *Mashups*, il définit la valeur à « *-1* », ce qui signifie que ce service web n'a participé à aucun *Mashup.*

Lorsque l'algorithme a terminé le parcours de tous les services Web, nous avons détecté que certains services n'avaient participé à aucun des *Mashups*, afin de disposer de données réelles pouvant être utiles pour l'entrainement de notre modèle d'apprentissage, nous devons éliminer ces services en supprimant toutes les lignes dont la valeur de l'attribut « **Mashup ID** » est mise à « *-1* » du fichier « **API Data** ». À la fin de ce processus de nettoyage, nous obtiendrons un nouveau fichier appelé « **Cleaned API Data** » qui est en réalité une matrice ou toutes les valeurs sont comprise entre 0 et 1, où nous trouvons les 648 services web ayant réellement participé à au moins un des *Mashups.*

III.3.1 Nettoyage des données des Mashups

Nous disposons seulement de 648 services web dans le fichier nettoyé « **Cleaned API Data** » et c'est trop peu pour les classer dans 4211 *Mashups*, pour cette raison nous avons dû éliminer certains *Mashups* afin de rapproché les dimensions des deux fichiers, l'algorithme suivant expliquera cette phase.

Algorithme 2 : *Nettoyage des Mashups*

```
Mushups = listOfMushups()
Nb_Mush = Length(Mushups)          # La taille globale de la liste des Mashups
for each integer i in Nb_Mush do
    Mushup= Mushups(i)
    R_APIs=Mushup(Related_APIs)
    if length( R_APIs ) ==1 do
        delete Mushup from listOfMushups()
```

Description de l'algorithme 2

Au début de cette étape, nous allons lister tous les Mashups qui figure dans le fichier « ***Mashup Data*** », ensuite nous allons extraire la valeur de l'attribut « **Related APIs** » pour chaque *Mashups* qui est un vecteur qui contient les noms des services qui le compose, selon la taille de ce vecteur on décide de garder ou de supprimer le *Mashup* concerné, si on trouve qu'il est constitué d'un seul service au plus, on le supprime, sinon on le garde.

A la fin nous allons sauvegarder la liste des Mashups comprenant au moins de deux services dans un nouveau fichier appelé « ***Cleaned Mashup Data*** » contenant 1461 *Mashups.*

III.3.2 Nettoyage des descriptions des services web

L'attribut « ***description*** » désigne la description des services web offert par les fournisseurs, par intuition, les mots qui apparaissent le plus souvent devraient peser d'avantage dans l'analyse de données textuelles, mais ce n'est pas toujours le cas. Des mots tels que « the », « for » et « you » appelés mots vides (**Stop Words**), apparaissent le plus souvent dans un corpus de texte, mais ont très peu de signification. Au lieu de cela, les mots rares sont ceux qui permettent réellement de distinguer les données et ont plus de poids. Par conséquent, nous devons supprimer tous les mots vides afin de ne conserver que les mots qui ont une signification, ont du sens et ont plus d'importance dans l'analyse de la description. La Figure III.3 et La Figure III.4 montrent un exemple de description du service web **LinkedIn** avant et après le nettoyage respectivement.

```
LinkedIn is the world's largest business social networking hub.
Launched in 2003, LinkedIn has millions of users and is
implemented in over 200 countries. One purpose of the site is
to allow registered users to maintain a list of contact details
of people with whom they have some level of relationship, called
Connections. Users can invite anyone (whether a site user or
not) to become a connection. \n\LinkedIn actually provides 2
APIs:\n-The JavaScript API is a rich client library enabling
you to build dynamic applications in the web browser. Use OAuth
2 to easily authorize users via the "Sign in with LinkedIn"
button, access LinkedIn data with native objects, and interact
with Plugins.\n\n-The REST API provides a simple, consistent
representation of people, companies, jobs, and the interactions
and relationships between them. Our query language lets you
read data in XML and JSON at the granularity and aggregation
that you choose. Use OAuth 1.0a to authorize users and begin
making REST API calls using any programming language.
```

Figure III.3 – *Description du service Web LinkedIn avant le nettoyage.*

```
linkedin worlds largest business social networking hub launched
2003 linkedin millions users implemented 200 countries one
purpose site allow registered users maintain list contact
details people level relationship called connections users
invite anyone whether site user become connection linkedin
actually provides 2 apis javascript api rich client library
enabling build dynamic applications web browser use oauth 2
easily authorize users via sign linkedin button access linkedin
data native objects interact plugins rest api provides simple
consistent representation people companies jobs interactions
relationships query language let read data xml json granularity
aggregation choose use oauth 1.0a authorize users begin making
rest api calls using programming language
```

Figure III. 4 – *Description du service Web LinkedIn après le nettoyage.*

III.4 Représentation des données

Avant de pouvoir introduire un Dataset dans le modèle choisi pour l'apprentissage profond, il est important d'effectuer un prétraitement afin que les données se comportent bien pour notre modèle. Les algorithmes d'apprentissage attendent des données d'entrée numériques. Nous avons donc représenté nos données catégorielles des services web et des *Mashup* sous forme numérique comme il est indiqué ci-dessous.

III.4.1 Les données des services web

L'attribut « **Catégories »** contient tous les domaines d'intérêt (par exemple, social, design, countries) des différents services. Nous regroupons ces mots clés dans un ensemble de mots à partir desquels nous dérivons le vecteur *V1* pour chaque API_i, si la catégorie *j* est sa catégorie principale, alors $V\ 1[i]\ [j]\ =\ 1$, sinon si la catégorie j est une catégorie secondaire, alors $V1\ [i]\ [j]\ =\ 0{,}5$, sinon $V1\ [i]\ [j]\ =\ 0$.

L'attribut « **Authentification** » regroupe les *13* protocoles d'authentification tels que « *Oauth1, OAuth 2, ws-security »* dans un ensemble de mots. Nous dérivons un vecteur V2 binaire. Si une API_i adopte le protocole *j* alors $V2[i][j]\ =\ 1$, sinon $V2[i][j]\ =\ 0$. Nous adoptons la même approche pour obtenir les vecteurs binaires pour les attributs « **Request Format** » et « **Response Format** ».

Pour les attributs « **style** », « **scope** », qui représentent une liste dénombrable des valeurs, nous avons utilisé une approche similaire de l'approche précédente pour obtenir les vecteurs binaires. Nous avons utilisé une méthode appelée "***One-Hot-encoding*** " : qui est un vecteur creux qui se caractérise par le fait qu'il n'a qu'un seul élément possédant la valeur *1* alors que toutes les autres possèdent la valeur *0*. Par exemple, la valeur REST dans l'attribut Style qui contient six valeurs, sera encodé par '*000100*'.

Pour le reste des attributs : « **SSL Support** », « **Restricted Access** » , « **DeviceSpecific** », « **Hypermedia** », « **Non proprietary** » qui ont des valeurs binaires sous forme (Yes/No). Nous dérivons cinq (05) vecteurs binaires. Chaque entrée dans ces vecteurs prend '*01*' si la valeur de l'attribut correspondant est "Oui" (Yes) et '*10*' sinon. Conformément au codage "*One-Hot-Encoding*" de la bibliothèque *Scikit-learn* sur python.

Après la phase d'encodage des attributs, nous procédons à la concaténation de tous les vecteurs binaires obtenues pour avoir en résultat un vecteur global de taille 1373 représentant la description détaillée du service stocké dans un fichier nommé « *Encoded_API_Data* ». Illustré dans la Figure III.5 ci-dessous. Ce vecteur sera introduit en tant qu'entrée dans notre modèle de classification multi label [49].

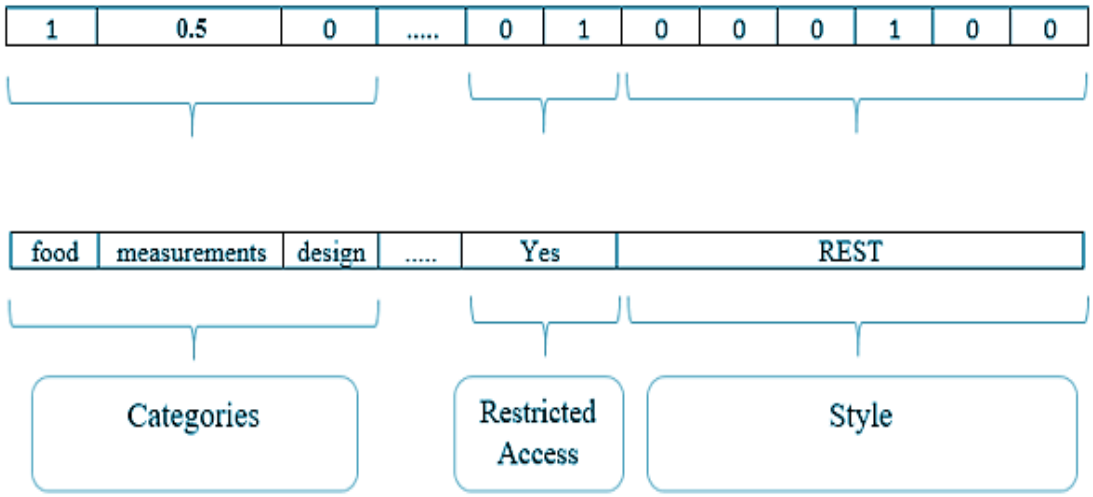


Figure III.5 – *Exemple d'un Encodage d'une API.*

III.4.2 Les données des Mashups

L'attribut « **Related API »** contient l'ensemble des services web qui se sont composés dans un *Mashup*, par exemple les deux services web « *LinkedIn et MeetUp* » ont participer dans le *Mashup « LinkedUp* », Nous avons construit une matrice binaire ***Target*** de taille (648,4211), ou les lignes représente les services web et les colonnes représente les *Mashups*, de telle façon que si le service i a participé dans le *Mashup* j, *alors* $Target[i][j] = 1$, ***sinon*** $Target[i][j] = 0$.

A la fin nous allons sauvegarder cette représentation dans un fichier nommé « *Encoded Mashup Data* ».

Nous avons remarqué que parmi les 648 services Web dont nous disposons, certains participent à un seul Mashup et d'autres participent à plusieurs. La nature des services Web leur permet de collaborer et de se composer entre eux pour des différents Mashups, un service Web peut réellement participer à plus d'un Mashup (classe), par conséquent, notre problème est considéré comme un problème de Classification Multi Label, nous allons donc présenter des définitions sur les diffèrent type de classification dans la section suivante.

III.5 Classification

Nous rappelons que la classification (chapitre 2, section 2.4.1) est un type d'apprentissage automatique supervisé basé sur l'association de labels aux donnée d'entrée, les méthodes de classification peuvent être classées en plusieurs classe :

III.5.1 La classification mono label

C'est une fonction d'approximation qui associe les données d'entrée à un label cible unique "l" à partir d'un ensemble de labels disjoints "L". Le problème de *Classification mono label* peut être divisé en deux catégories : classification binaire et classification multi-classe. Lorsque les données d'entrée sont classées dans l'une des deux classes, on parle de classification binaire. Lorsque les données d'entrée correspondent à l'un des groupes labels cibles, on parle de classification multi-classe [50].

III.5.1.1 La classification binaire

La classification binaire signifie une tâche de classification comportant deux classes. La sortie réelle de nombreux algorithmes de classification binaire est un score de prédiction. Ce score indique la certitude du système que l'observation donnée appartient à la classe positive. Pour décider si l'observation doit être classée comme positive ou négative, en tant que consommateur de ce score, vous devez interpréter le score en sélectionnant une limite de classification, ou seuil et comparer le score à ce seuil. Toute observation avec un score supérieur au seuil est alors prédite en tant que classe positive et tout score inférieur au seuil en tant que classe négative.

III.5.1.2 La classification multi classe

La classification multi classe signifie une tâche de classification comportant plus de deux classes ; par exemple, classer un ensemble d'images de fruits qui peuvent être des oranges, des pommes ou des poires. La classification multi classe repose sur l'hypothèse que chaque donnée est attribuée à une seule et même label : un fruit peut être une pomme ou une poire mais pas les deux à la fois [51].

III.5.2 La classification multi label

La classification multi-label est une généralisation de la classification mono-label classique où chaque exemple xi peut être étiqueté par un ou plusieurs labels simultanément. L'apprentissage à partir de données multi-label est un problème qui a suscité beaucoup d'attention ces dernières années par la communauté d'apprentissage automatique et les communautés connexes et a conduit au développement de plusieurs approches d'apprentissage avec différentes stratégies. Ces approches peuvent être organisées en trois grandes familles :

- **Approches d'apprentissage par transformation** : elles transforment le problème d'apprentissage multi-label en un ou plusieurs problèmes de classification ou de régression mono-label.
- **Approches d'apprentissage par adaptation** : elles adaptent des algorithmes d'apprentissage pour des données multi-label.
- **Approches d'apprentissage ensemble** : elles utilisent un ensemble de classificateurs issus de la première ou de la deuxième famille d'approches.

III.6 Notre démarche pour un système de recommandation des services web

Notre approche proposée a pour but d'améliorer le processus de découverte des services web, en détectant l'ensemble de services qui participe dans les mêmes *Mashups.* Notre approche consiste à chercher les services similaires qui peuvent être des substituts ou des concurrents possibles, sinon des services qui peuvent être composés ensemble, nous allons essayer de dresser un modèle, qui peut nous prédire quel sont les services web qui peuvent travailler ensemble selon la prédiction des *Mashups* qu'un service peut y participer.

Un problème de classification est un problème d'apprentissage supervisé, il nécessite un ensemble d'entrées qui sont étiquetés avec les sorties souhaitées, donc les données d'entrée de notre modèle sont stockées dans le fichier « *Encoded_API_Data* » et les données de sortie sont stockés dans le fichier « *Encoded_Mashup_Data* ».

III.6.1 Préparation des données d'apprentissage

Algorithme *3 :* *Data_Splitting*

Input_Train = 80% de l'ensemble de données d'entrée pour l'entrainement du modèle.

Input_Test = 20% de l'ensemble de données d'entrée pour le tests et l'évaluation du modèle.

Output_Train = 80% de l'ensemble de données de sortie pour l'entrainement du modèle.

Output_Test =20% de l'ensemble de données de sortie pour le tests et l'évaluation du modèle.

Model = Fit (Input_Train, Output_Train)

Training_Accuracy = Model.Evaluate (Input_Test, Output_Test)

Prediction = Model.Predict (Input_Test)

Testing_Accuracy = Compare (Output_Test, Prediction)

Description de l'algorithme 3

L'algorithme divise les deux ensembles donnés en une partie d'apprentissage qui représente 80% des données et 20% pour les données du test. Ensuite, nous introduisons les quatre (04) ensembles au modèle qui effectue la Classification Multi-Label que nous allons introduire par la suite.

III.6.2 Notre Modèle de classification Multi Label

Nous avons utilisé un *Réseau de Neurone FeedForward* aussi appelé *Multi Layer Perceptron (MLP)* un Classificateur Multi Label, comme le montre la Figure III.6, notre modèle est constitué d'une couche d'entrée comportant 1371 neurones, six (04) couche cachées et une couche de sortie comporte 1461 neurones avec « *Sigmoid* » comme fonction d'activation.

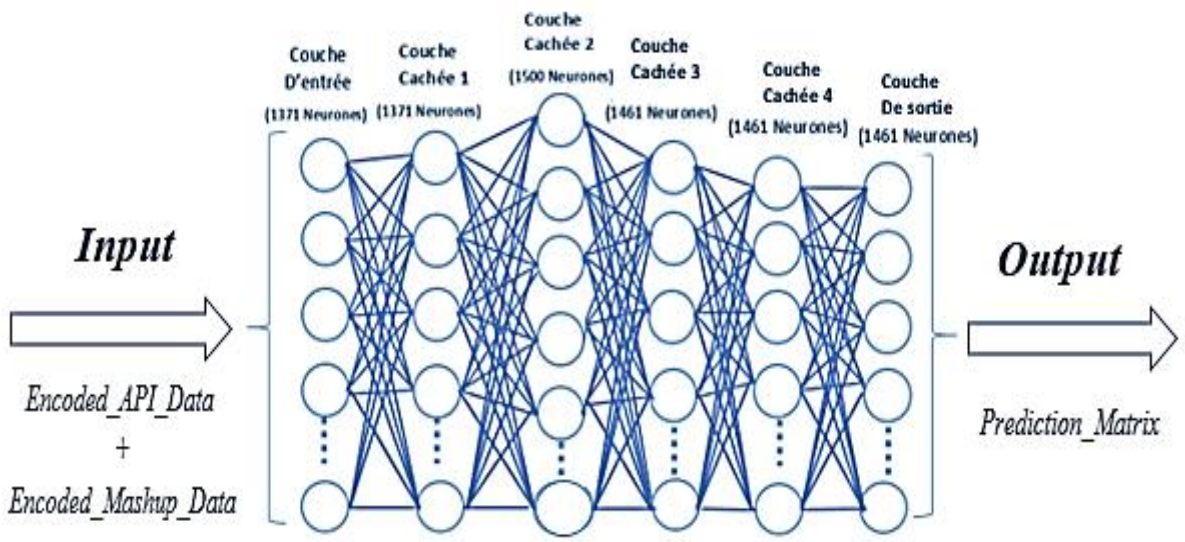


Figure III.6 – *l'architecture de notre modèle.*

Après avoir injecté l'ensemble de donnée au modèle de classification, nous obtiendrons les résultats de prédiction des Mahups qu'un service peut y participer, nous avons stocké ces résultats dans une matrice que nous avons appelé « Prediction_Matrix » de taille (648,1461).

A cause de la fonction sigmoïd de la dernière couche (couche de sortie), les valeurs de la matrice de prédiction sont des probabilités $\in]0,1[$ comme l'illustre la Figure III.7, Par conséquent, nous devons fixer un seuil qui facilitera la transformation de la matrice de prédiction en une matrice binaire afin de pouvoir confirmer plus facilement si le service i peut participer au Mashup j ou pas.

	0	1	2	3	4	5	6	7	8	9	...	1102	1103
0	0.314270	0.195677	0.148954	0.249152	0.200451	0.274389	0.266788	0.071823	0.234452	0.137123	...	0.669132	0.356911
1	0.339318	0.213783	0.177313	0.272655	0.224416	0.300928	0.289727	0.092447	0.254672	0.159234	...	0.650171	0.367889
2	0.327894	0.196157	0.155966	0.257888	0.205937	0.279832	0.273919	0.077890	0.237938	0.141698	...	0.654888	0.358172
3	0.328874	0.203423	0.151984	0.255444	0.196783	0.275611	0.278316	0.075586	0.235511	0.139511	...	0.658262	0.360043
4	0.322787	0.195601	0.157603	0.254225	0.205030	0.281187	0.271426	0.075848	0.233239	0.141366	...	0.665509	0.350768
5	0.325688	0.202379	0.155379	0.257172	0.204511	0.281787	0.279204	0.077964	0.240184	0.142293	...	0.661778	0.361998
6	0.334098	0.222106	0.172570	0.275428	0.226484	0.296004	0.290356	0.092867	0.256953	0.161663	...	0.648815	0.371362
7	0.377741	0.280058	0.234840	0.334383	0.293946	0.348878	0.340101	0.155964	0.313811	0.227759	...	0.617309	0.402092
8	0.336163	0.216108	0.177204	0.276554	0.222819	0.299184	0.287498	0.091486	0.254862	0.158667	...	0.650786	0.369593
9	0.348068	0.228529	0.185411	0.285730	0.240761	0.311825	0.294606	0.101424	0.265425	0.174064	...	0.642505	0.375028
10	0.382374	0.287311	0.234823	0.332387	0.289626	0.348905	0.348201	0.159153	0.319011	0.231221	...	0.608440	0.410463
11	0.352768	0.235582	0.192160	0.287714	0.240701	0.306476	0.305529	0.106674	0.275501	0.179800	...	0.641295	0.380438
12	0.353235	0.236999	0.191623	0.295888	0.244416	0.307969	0.306905	0.113724	0.276776	0.184282	...	0.633986	0.387204
13	0.351283	0.235204	0.192908	0.290937	0.246574	0.313826	0.303977	0.108520	0.272150	0.180281	...	0.637468	0.382046
14	0.344195	0.231944	0.187551	0.288383	0.239279	0.300324	0.302230	0.105511	0.272784	0.176248	...	0.638348	0.376833
15	0.393902	0.291922	0.252221	0.330402	0.300049	0.359920	0.354455	0.163109	0.325843	0.238942	...	0.596309	0.403582
16	0.337749	0.221879	0.177938	0.277406	0.227658	0.300136	0.293700	0.095066	0.257886	0.162796	...	0.645051	0.371369

Figure III.7 – *Un Exemple De Prédiction du Modèle.*

Dans le cas d'un nouveau service web qui n'avait aucun historique d'interaction, cette matrice nous permet de lui prédire des *Mashups* pour en faire partie et l'initialisé dans le système, sinon elle permet de découvrir de nouvelle possibilité pour qu'un service peut participer à d'autre *Mashups.*

Parmi les attributs d'un service web dans notre Dataset on trouve sa description que nous avons nettoyé au début dans la section (III.3.2), nous allons calculer le degré de similarité entre tous les services, à partir de ces deux paramètres nous pouvons dresser une matrice d'adjacence dont la valeur est basée sur l'historique d'interaction, ceci pour déterminer en quelle relation nous allons recommander les services (composition, substitution ou compétition).

III.6.3 Degré de similarité

Les ordinateurs traitent les données numériques mieux que les données textuelles. TF-IDF est l'une des techniques les plus utilisées pour traiter les données textuelles. Nous allons l'utiliser pour calculer le degré de similarité entre les descriptions des services web, cela nous aidera à savoir si deux services sont similaires ou pas, nous allons introduire la méthode de TF-IDF dans ce qui suit.

III.6.3.1 TF-IDF

TF-IDF signifie « *Fréquence de terme - Fréquence de données inverse* ». Premièrement, nous allons apprendre ce que ce terme signifie mathématiquement.

Term Frequency (TF) : donne la fréquence du mot dans chaque document du corpus. C'est le rapport entre le nombre de fois où le mot apparaît dans un document et le nombre total de mots dans ce document. Il augmente à mesure que le nombre d'occurrences de ce mot dans le document. Chaque document à sa propre TF.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}}$$

Fréquence de données inverse (IDF) : utilisée pour calculer le poids des mots rares dans tous les documents du corpus. Les mots qui apparaissent rarement dans le corpus ont un score IDF élevé. Il est donné par l'équation ci-dessous.

$$idf(w) = \log\left(\frac{N}{df_t}\right)$$

En combinant ces deux éléments, nous obtenons le score TF-IDF (w) pour un mot dans un document du corpus. C'est le produit de TF et IDF :

$$w_{i,j} = tf_{i,j} * \log\left(\frac{N}{df_t}\right)$$

III.6.4 Recommandation

Algorithme 4 : *Recommandation*

```
Nb _ Mashup = Length(Prediction)          # Nombre de ligne
Nb _ Mush = Length(Nb _ Mashup)           # Nombre de colonne
Recommend=[][]                            # Matrice d'adjacence
Threshold = 0.70
for j = 1 .. Nb _ Mush do
    while(i< Nb _ Mashup ) do
        for(k= i .. Nb _ Mashup ) do
            if( Prediction_Matrix [i][j]== Prediction_Martix[k][j]==1 ) &
              (Similarity_Degry_Matrix [i][k]>= Threshold) do
                Recommend[i][k]=(substitution/competition)
          else if ( Prediction_Matrix [i][j]== Prediction_Matrix [k][j]==1 ) &
                (Similarity_Degry_Matrix [i][k]< Threshold) do
                 Recommend[i][k]=(composition)
           else  Recommende[i][k]=(Nan)
       i=i+1
```

Description de l'algorithme 4

Si les deux services web (i, j) sont recommandés dans un même *Mashup* et qu'ils présentent un degré de similarité élevé qui dépasse le seuil donc ils sont considérés comme étant des services similaires et semblables, ils seront recommandé en tant que des services substituts ou concurrents, si leur degré de similarité est faible mais il peuvent participé dans le même *Mashup*, cela veut dire que la recommandation sera en terme de composition entre différents services, sinon s'ils n'ont aucune similarité et ne participent ensemble dans aucun *Mushup*, ils n'auront aucune interaction.

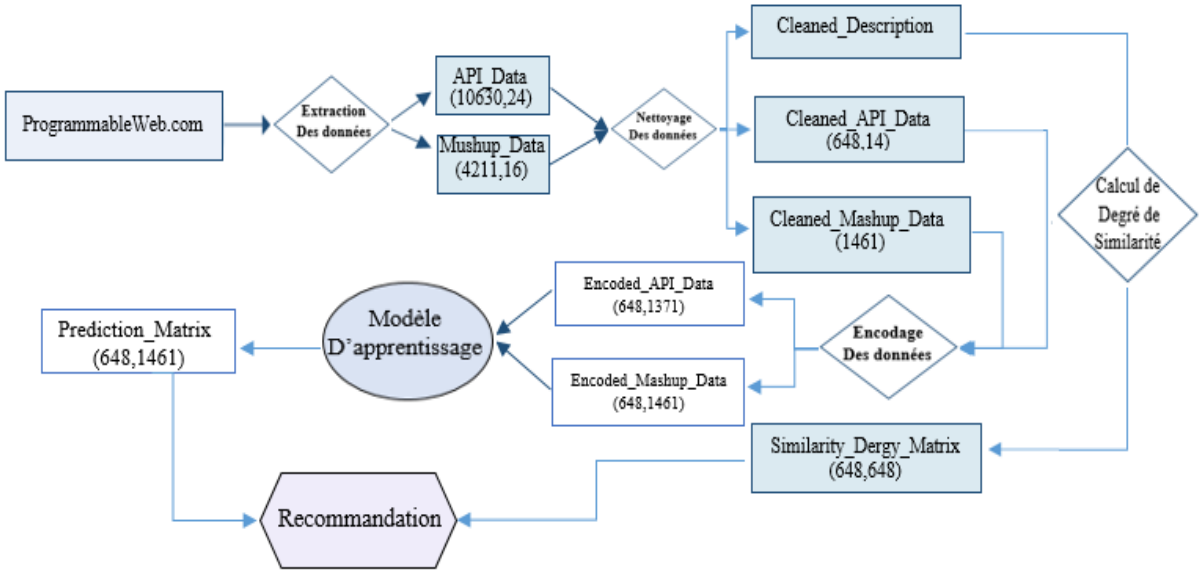


Figure III.8 – *L'Architecture de l'Approche Proposée.*

III.7 Environnement de Travail

III.7.1 Matériel

L'implémentation de notre approche a été réalisée sur deux stations ayant comme configuration matérielle ce qui suit :

La station 1 :

- Un CPU Intel ® Core ™ i7-7500 CPU @ 2.70 GHz 2.90GHz
- Une mémoire RAM de 16.00 Go DDR4 @ 2133MHz.

La station 2 :

- Un CPU Intel ® Core™ I5-5300 @ 2.30 GHz 2.29GHz
- Une mémoire RAM de 8 Go DDR4 @ 2133MHz.

III.7.2 Logiciel

Les différents logiciels utilisés pour mette en œuvre la solution sont :

- Système : Windows 10 64-bit
- Langage de programmation : Python v3.7.3 64-bit
- Deep Learning Library : Keras 2.2.4
- Autres Bibliothèques supplémentaires sur Python : Sklearn, NLTK...

III.8 Techniques Utilisées

III.8.1 Langage de Programmation Python

Pour développer notre approche, nous avons opté pour le langage de programmation Python v3.7.3 64-bit.

Python est un langage de programmation orienté objet clair et puissant, comparable à Perl, Ruby, Scheme ou Java. Il Utilise une syntaxe facile et simple à utiliser, facilitant la lecture des programmes. Python peut également être modifié et redistribué librement, car le langage est protégé par des droits d'auteur et disponible sous une licence open source.[50] En raison de sa popularité, un grand nombre des bibliothèques ont été publiées pour prendre en charge les nombreuses tâches de programmation courantes telles que la connexion à des serveurs Web, la recherche de textes avec des expressions régulières, traitement de textes et d'ensembles de données, la lecture et la modification de fichiers. Et notamment les bibliothèques et les FrameWorks d'apprentissage automatique et profond que nous avons utilisé.

Dans les prochaines sections nous allons présenter les bibliothèques et les principaux outils utilisés pour le développement de notre solution.

III.8.2 Deep Learning Library Keras

Python Deep Learning Library Keras 2.2.4 [51] est une API de réseaux de neurones de haut niveau, écrite en Python et capable de fonctionner sur un backend TensorFlow (Machine Learning Framework) pour simplifier le processus de construction d'applications d'apprentissage profonds. Il est Conçu pour permettre une expérimentation facile et rapide avec des réseaux neurones profonds, il se concentre sur la modularité, l'extensibilité et une conception conviviale des applications. Keras offre des API simples et cohérentes et minimise le nombre d'actions nécessaires de l'utilisateur pour les cas d'utilisation commune et il fournit un retour d'information clair et pratique sur les erreurs de l'utilisateur. Il a été développé dans le cadre du projet ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System) à Google [52].

Nous avons utilisé Keras pour créer le réseau de neurones FeedForward et entraîner ce modèle sur un environnement ayant un support CPU.

III.8.3 Scikit-Learn

Scikit-learn est une bibliothèque libre open-source pour Python dédiée à l'apprentissage automatique, cet outil est simple et efficace pour l'exploration de données (Data Mining) et l'analyse de données (Data Analysis). Elle comporte notamment divers algorithmes de classification, de régression et de clustering, y compris les machines à vecteurs de support (*SVM :Support Vector Machines), Random Decision Forests Classifier, Gradient Boosting Classifier, K-Means* et plusieurs d'autre algorithme et fonctions utiles pour l'analyse et le prétraitement de données. Cette Bibliothèque est conçue pour s'harmoniser avec les bibliothèques numériques et scientifiques de Python à savoir NumPy et SciPy.

Nous avons utilisé cette bibliothèque dans la phase de représentation de données et l'encodage des attributs, nous l'avons utilisé aussi pour diviser l'ensemble de données en deux parties, une dédiée à l'apprentissage et l'autre aux tests et pour le calcul de degré de similarité.

III.8.4 NLTK

NLTK : Natural Language Tool Kit est une plate-forme leader pour la création de programmes Python utilisant des données en langage humain. Il fournit des interfaces faciles à utiliser avec plus de 50 corpus et ressources lexicales tels que WordNet, ainsi qu'une suite de bibliothèques de traitement de texte pour la classification, la création de jetons, le suivi, le balisage, l'analyse et le raisonnement sémantique. Et un forum de discussion actif.

Grâce à un guide pratique présentant les bases de la programmation, des sujets en linguistique informatique et une documentation complète sur les API, NLTK convient aux linguistes, ingénieurs, étudiants, enseignants, chercheurs et utilisateurs du secteur. NLTK est disponible pour Windows, Mac OS X et Linux. Mieux encore, NLTK est un projet gratuit, à code source ouvert et piloté par la communauté [53]. Nous avons utilisé la bibliothèque NLTK pour le nettoyage des descriptions des services web.

III.8.5 Numpy

Numpy est un package de traitement de tableau à usage général. Il fournit un objet de tableau multidimensionnel hautes performances et des outils pour utiliser ces tableaux. C'est le paquet fondamental pour l'informatique scientifique avec Python. Outre ses utilisations scientifiques évidentes, Numpy peut également être utilisé comme un conteneur multidimensionnel efficace de données génériques.

III.8.6 Pandas

Pandas est une bibliothèque open source sous licence BSD fournissant des structures de données hautes performances et faciles à utiliser, ainsi que des outils d'analyse des données pour le langage de programmation Python [54].

III.9 Discussion des Résultats

III.9.1 Entraînement et Paramétrage des Modèles

Après plusieurs exécutions, nous avons fini par identifier les paramètres qui retournent la meilleure prédiction pour notre modèle de classification « multi-label ». Dans ce qui suit, nous allons présenter et argumenter le choix de ces paramètres.

III.9.1.1 Fonction d'optimisation SGD

Descente de gradient stochastique (SGD) effectue une itération avec chaque échantillon d'apprentissage (c'est-à-dire qu'après le passage de chaque échantillon d'apprentissage, il calcule la perte et met à jour le poids). Comme les poids sont mis à jour trop fréquemment, la courbe de perte globale serait très bruyante. Cependant, l'optimisation est relativement rapide par rapport aux autres. La formule pour les mises à jour de poids peut être exprimée de la manière suivante : $Poids = Poids - vitesse\ d'apprentissage * Perte$, Lorsque la vitesse d'apprentissage est un paramètre que nous définissons dans l'architecture de réseau.

III.9.1.2 Fonction d'Activation Sigmoïde

Sigmoïde est utilisée dans la couche de sortie d'un réseau de neurones si de préférence la classe est binaire. Cette fonction est la plus adéquate dans notre cas parce que toutes les valeurs de l'ensemble de données sont comprises entre 0 et 1 après la codification. Cette fonction convient parfaitement à notre traitement étant donné que nous utilisons la représentation *one-hot vector* pour les données en entrée. La fonction transforme l'entrée X comme suit :

$$y = \frac{1}{1+e^{-x}}.$$

III.9.1.3 Fonction d'Erreurs BinaryCross Entropy

Aussi appelée Sigmoid Cross-Entropy loss. C'est une activation de la fonction Sigmoid plus du Cross-Entropy loss. Contrairement à la Softmax loss, elle est indépendante pour chaque composant de vecteur (classe), ce qui signifie que la perte calculée pour chaque composant de vecteur de sortie de modèle n'est pas affectée par les valeurs des autres composants. C'est la raison pour laquelle il est utilisé pour la classification multi-labels, car en vue d'un élément appartenant à une classe donnée ne devrait pas influencer la décision prise par une autre classe. Elle s'appelle BinaryCross Entropy avec sa formule :

$$BCE = -\frac{1}{N}\sum_{i=0}^{N} y_i \,.\log(\widehat{y_i}) + \left(1_{y_i}\right).\log(1-\widehat{y_i})$$

III.9.1.4 Taux d'Apprentissage (Learning Rate)

Le taux d'apprentissage est un paramètre qui contrôle la vitesse de convergence vers l'erreur minimale du modèle. Un taux d'apprentissage trop élevé peut faire rebondir la solution et empêcher la convergence. D'autre part, si le taux d'apprentissage est trop faible il conduit à une convergence très lente. Pour cela nous avons choisi un taux d'apprentissage de 0.01 (10^{-2}) qui est prouvé par l'expérimentation comme un taux idéal pour notre modèle d'apprentissage.

III.9.1.5 Nombre d'Itérations (Epochs)

Une itération correspond à une passe en avant à travers tout le réseau de neurone plus une passe en arrière (Back-propagation) pour modifier les poids. Nous étions contraints à limiter le nombre d'itérations à 40 car ils étaient largement suffisants pour entraîner le modèle et avoisiner une très bonne précision (99.79 %) et une très faible perte (1.84%) comme il est illustré dans la Figure III.9.

```
130/130 [==============================] - 1s 7ms/step
Accuracy: 99.79% loss: 1.84%
```

Figure III.9 – *Précision et Perte du Modèle FeedForward.*

III.9.1.6 Taille de Batch (Batch Size)

La taille du Batch correspond au nombre total d'exemples d'entraînements présents dans un seul paquet. Pour concevoir notre modèle, nous avons un ensemble de données de 648 instances (Services Web), nous divisons l'ensemble de données en paquets de 32 instances chacun appelé Batch. La division de Dataset en mini-Batches permet au réseau de neurones de faire un apprentissage plus rapide et nécessite moins de mémoire pour stocker les résultats (exemples d'apprentissage).

III.9.1.7 Nombre de couches cachées

Dans le but d'aboutir à des meilleurs résultats. Nous avons défini plusieurs configurations du réseau de neurones en variant le nombre de couches cachées et le nombre de nœuds dans chaque couche cachée. Ceci nous a permis de comparer les performances du modèle pour chaque configuration et ainsi aboutir à la meilleure configuration. Nous avons remarqué dans un premier temps le fait d'augmenter le nombre de couches ne donne pas forcément une bonne précision (accuracy) et peut amener au problème de sur-apprentissage (Overfitting).

Par contre, le modèle n'arrive pas à extraire les caractéristiques utiles avec un nombre réduit de couches (une ou deux). Ce qui conduit dans ce cas à juste copier l'entrée par la couche intermédiaire (problème de généralisation). Nous avons remarqué qu'un nombre de couches égal à 4 ou 5 permet d'obtenir la meilleure précision d'apprentissage. A noter que la différence du nombre de neurones entre une couche et la suivante ne descend pas sous le seuil de 40%, par exemple si la couche $h1$ contient 100 neurones alors $h2$ va contenir au minimum 40 neurones.

III.9.1.8 Le Dropout

C'est un paramètre additionnel qui consiste à ignorer des unités (neurones) aléatoirement en les mettant à 0 pour chaque mise à jour des poids pendant l'apprentissage ce qui permet d'éviter le sur-apprentissage (Overfitting).

III.9.2 Entraînement du modèle FeedForward

Dans cette phase nous allons entraîner notre modèle FeedForward pour la classification multi-label Nous commençons par diviser notre ensemble de données obtenu de la phase de représentation en deux sous-ensembles, un ensemble qui contient 80% des services (équivalent à 518 services) que nous allons utiliser pour l'apprentissage du modèle FeedForward et les 20% services restant pour l'ensemble de tests.

III.9.2.1 Paramétrage du Modèle FeedForward

Pour créer notre modèle FeedForward nous avons utilisé la bibliothèque Keras sur un Backend TensorFlow, nous avons créé six (06) couches "Dense" qui sont des couches entièrement connectées (Fully-Connected Layers) qu'on l'utilise pour prédire les données en sortie. La première couche est une couche d'entrée de taille 1371 qui correspond à la taille d'encodage des attributs d'un service Web, la première couche cachée est aussi de taille 1371, la deuxième couche cachée est de taille 1500, la troisième et la quatrième couche cachée sont de taille 1461 et enfin une couche de sortie de taille 1461 qui correspond au nombre de *Mashups*, Dans notre cas, l'input du modèle FeedForward est un service Web et l'output est un *Mushup* auquel il pourra participer.

III.9.2.2 Paramètres d'apprentissage

Pour réaliser la classification multi labes des services Web nous avons opté pour les paramètres suivants :

- Optimiseur d'erreur : Stochastic Gradient Descente
- Fonction d'erreur : Binary CrossEntropy
- Taille de Batch : 32
- Nombre d'itérations : 30
- Taille de Drop-out : 0.1 (10%)

III.9.2.3 Résultat d'apprentissage

Nous avons exécuté le modèle avec 30 itérations, nous avons obtenu une précision autour de 0.9, Les Figure III.10 et III.11 affichent respectivement le graphe de précision et d'erreur de modèle FeedForward.

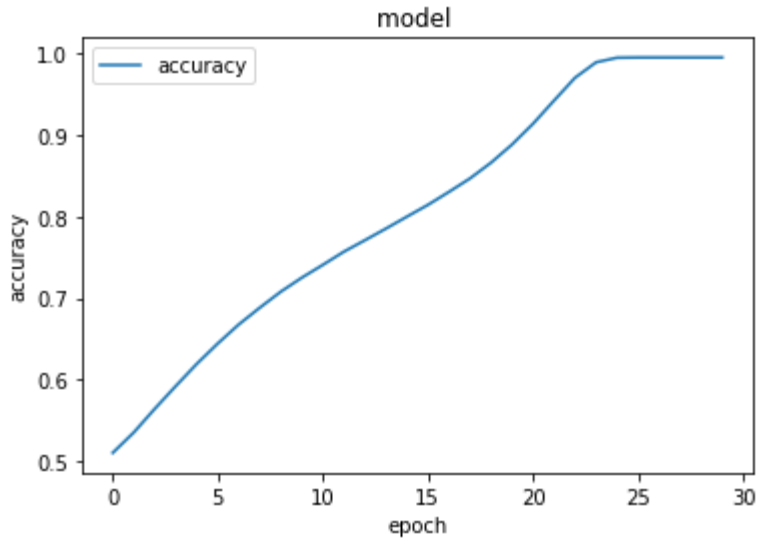


Figure III.10 – *Graphe de précision.*

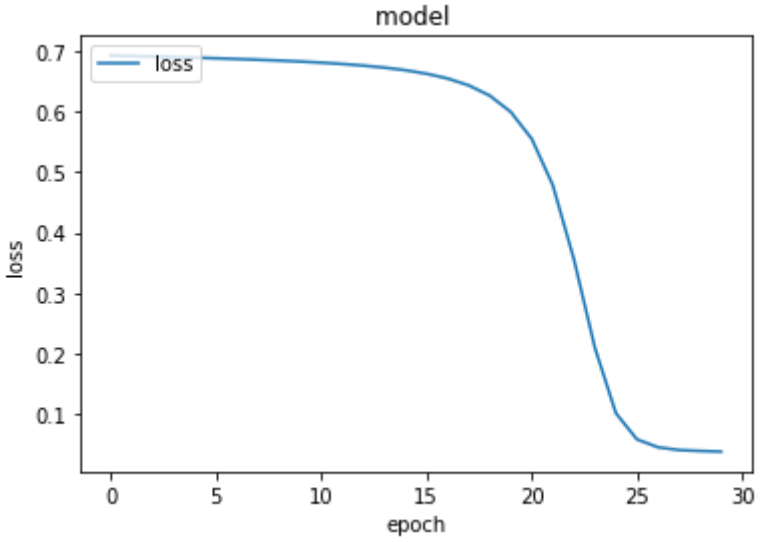


Figure III.11 – *Graphe de perte*

III.10 Conclusion

Tout au long de ce chapitre, nous avons présenté notre approche pour la recommandation des services web, en expliquant toutes les étapes de réalisation, en commençons par l'extraction, le nettoyage et le codage de nos données qui sont sauvegardés au niveau des deux fichiers « APIs_Data » et « Mashup_Data », qui seront introduit comme entrée de notre modèle de Deep Learning « Feed Forward Network » qui fait la classification multi-label pour qu'il puisse nous prédire les classes dont un service pourra y participer. Nous avons, également calculé le degré de similarité entre les descriptions des services web, qui va nous aider largement à définir les relations de notre recommandation (composition, substitution et compétition), finalement nous avons présenté les résultats de notre modèle.

Conclusion Générale & Perspectives

Dans le présent travail, nous avons essayé de proposer une nouvelle approche pour la recommandation des services web basée sur le profil de chaque service et son historique d'interaction avec ses semblables (compositions, substitutions & compétitions). Nous avons introduit la technologie des services web ainsi que l'objet des travaux de recherche qui ont donné naissance à ses deux versions : *sémantique* et *sociale.* Ensuite, nous avons fait un survol sur les différentes techniques d'analyse de grandes masses de données en se focalisant plus précisément sur l'apprentissage profond (Deep Learning) qui donnent la possibilité dans notre cas de proposer des modèles pour l'analyse de la mémoire du système, cette dernière est établie à partir des données d'interactions des services web.

Pour commencer, nous avons dressé un réseau de neurones *FeedForward* qui fait une classification « multi-labels », ce modèle permet de prédire les classes ou les *Mashups* dont un service a de fortes chances d'y participer, en l'entrainant avec un Dataset issu du monde réel collecté à partir du répertoire des APIs « *ProgrammableWeb.com* ». Nous avons, par la suite, calculé le degré de similarité entre les descriptions des services web, ces deux paramètres nous aident largement à faire notre recommandation. Si les descriptions des deux services web présentent une similarité élevée, ils sont recommandés comme concurrents ou substituteurs. Dans le cas contraire, ils seront considérés comme étant des services coopérateurs.

Comme tout travail de recherche dans le domaine, nous avons eu des difficultés à se procurer un Dataset avec des données historisées et encore plus un Dataset assez grand pour permettre une phase d'apprentissage complète.

En effet, l'évaluation d'un système de recommandation en utilisant des critères de performances adéquats est une étape importante pour démontrer la qualité du système proposé. Dans notre cas, nous avons travaillé avec un Dataset où le nombre de services web que nous avons pu obtenir était trop faible par rapport à un problème de classification Multi-Label avec près de 4000 classes distinctes. Néanmoins, le modèle, malgré plusieurs déviations lors de la phase d'apprentissage, a pu prédire pour certains services de nouvelles possibilités de participations dans des nouvelles classes dont il ne faisait pas partie, qui se sont avérées juste.

Comme perspective à notre travail, il serait plus intéressant de dresser un autre modèle de Deep Learning qui prends comme entrée les données encodées des services web et essaie de prédire leur nombre de participations dans les *Mashups*, ce modèle permet de contourner le problème de « comment choisir le seuil à partir duquel un *Mashup* sera recommandé ? ». Ce qui réduit énormément la complexité du problème.

Bibliographie

[2] Approche basée sur la détection des communautés cachées pour l'amélioration du processus de découverte des services web, Université Ibn Khaldoun Tiaret.

[3] Jean-Marie Chauvet, Services web avec SOAP,WSDL,UDDI,ebXML …,Edition EYROLLES 61,Bld Saint-Germain 75240 Paris cedex 05, page 19

[4] A. Polleres J. Bruijn M. Stollberg D. Roman D. Fensel H. Lausen and J. Domingue. Enabling semantic web services: The web service modeling ontology, springerverlag berlin heidelberg. 2007.

[5] T R Gruber. A translation approach to portable ontology specifications knowledge acquisition. 1993.

[6] Michael Daconta Leo Obrst et Kevin Smith. Developing the semantic web: a guide to the future of xml web services and knowledge management. 2003

[7] Patrick Kellert et Farouk Toumani. Les web services sémantiques, 2004.

[8] R Akkiraju J Farrell and Al. Web service semantics. 2005.

[9] Hakim Hacid Zakaria Maamar and Michael N.Huns. Why web services need social networks. IEEE Computer Society,90-94, PDF: 05731594.

[10] Kathleen M. Carley andWenji Mao Fei-YueWang, Daniel Zeng. Social computing: From social informatics to social intelligence. IEEE Intelligent systems, vol. 22, no. 2, pages 79,83, 2007.

[11] Singh and M.N. Huhns. Service-oriented computing: Semantics, processes, agents. John Wiley et Sons, 2005.

[12] Johann Stan, Fabrice Muhlenbach, Christine Largeron. Recommender Systems using Social Network, Vol 23, page 5,6.

[19] Régression Linéaire simple et multiple, BOUKRIF Nouara, Université Abderrahmane MIRA-Bejaia 2016,Vol 52,Page 2

[22] Réduction des dimensions des données en apprentissage artificiel, Y. Bennani, S. Guérif, E. ViennetUniversité Paris 13, LIPN - CNRS UMR 703099, avenue J.B. Clément, F-93430 Villetaneuse, Vol 28, Page 2

[28] A. Schmitt, B. Le Blanc, M.-M. Corsini, C. Lafond et J. Bruzek, « Les réseaux de neurones artificiels », Bulletins et mémoires de la Société d'Anthropologie de Paris [En ligne], 13 (1-2) | 2001, 14 janvier 2010.

[30] Tommaso Teofili , Deep Learning for Search ,November 2017 , Publication in June 2019 *(*estimated*)*,ISBN 9781617294792 , 325 pages *(estimated)*

[31] Deep Learning with Python - A Hands-on Introduction, Nikhil Ketkar ,18 avr. 2017, Edition Apress

[32] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. Journal of Machine Learning Research, 12, 2121–2159

[33] Geoff Hinton, Neural Networks for Machine Learning, Lecture 6a, Overview of mini-batch gradient descent, Page 15

[34] Dauphin, Y., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. arXiv, 1–14

[35] Sebastian Ruder, "An overview of gradient descent optimization algorithms"

[36] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove : Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 1532

[38] Sutton, R. S. (1986). Two problems with backpropagation and other steepestdescent learning procedures for networks. Proc. 8th Annual Conf. Cognitive Science Society

[39] ADAM : "A METHOD FOR STOCHASTIC OPTIMIZATION", ICLR 2015. Diederik P. Kingma, University of Amsterdam, OpenAI Jimmy Lei Ba, University of Toronto

[42] Kingma, D. P. (2013). Fast gradient-based inference with continuous latent variable models in auxiliary form. Technical report, arxiv :1306.0733. 652, 689, 696

[43] Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In ICML’2014.

[44] Peilu Wang Yao Qian Frank K. Soong Lei He Hai Zhao, A Unified Tagging Solution: Bidirectional LSTM Recurrent Neural Network with Word Embedding, Nov 2015.

[48] Etat de l’art sur les Systemes de Recommandation, Nicolas Bechet, Projet AxIS de l’INRIA, dans le cadre du projet Addictrip

[49] Hamza Labbaci, Brahim Medjahed, Faisal Binzagr, and Youcef Aklouf. 2017. A Deep Learning Approach for Web Service Interactions. In Proceedings of ACM Web Intelligence conference, Leipzig, Germany, August 2017 (WI’17).

Webographie

[1] Chabane Refes, Les services web, 22 février 2017 , https://openclassrooms.com/fr/courses/219329-les-services-web, Consulté «Mars 2019»

[13] Analytique Big Data, (2016) , https://www.lemagit.fr/definition/Analytique-Big-Data , Consulté « Avril 2019 »

[14] Data mining : comment exploiter au mieux le potentiel des données, (2016), https://www.ionos.fr/digitalguide/web-marketing/analyse-web/data-mining-la-methode-danalyse-du-big-data/, Consulté « Avril 2019 »

[15] Analyse en composantes principales (ACP) , https://fr.mathworks.com/discovery/ principal-component-analysis.html, Consulté « Avril 2019 »

[16] Machine Learning : Introduction à l'apprentissage automatique , Metomo JOSEPH BERTRAND RAPHAËL (2017), École Supérieure d'Informatique de Paris , https://www.supinfo.com/articles/single/6041-machine-learning-introduction-apprentissage-automatique , Consulté « avril 2019 »

[17] Les fondamentaux du Machine Learning, Baptiste Pesquet, École Nationale Supérieure de Cognitique de Bordeaux , https://www.bpesquet.fr/fondamentaux-ml, Consulté « Avril 2019 »

[18] Data mining : comment exploiter au mieux le potentiel des données, (2016), https://www.ionos.fr/digitalguide/web-marketing/analyse-web/data-mining-la-methode-danalyse-du-big-data/, Consulté « Avril 2019 »

[20] Classification supervisée Aperçu de quelques méthodes avec le logiciel R ,(2015), http://www.math.univangers.fr/~labatte/ classificatio nsupervisee.pdf , Consulté « Avril 2019 »

[21] Analyse de cluster, https://fr.mathworks.com/discovery/cluster-analysis.html, Consulté « Avril 2019 »

[23] Explorez vos données avec des algorithmes non supervisés , Yannis Chaouche, Chloé-Agathe Azencott, 2019, https://openclassrooms.com/fr/courses/4379436-explorez-vos-donnees-avec-des-algorithmes-non-supervises/ , Consulté « Avril 2019 »

[24] Apprentissage par renforcement, https://dataanalyticspost.com/Lexique/apprentissage-par-renforcement, Consulté « Avril 2019 »

[25] Deep learning (apprentissage par réseau neuronal profond), TechTarget ,Margaret Rouse, avril 2018 , https://whatis.techtarget.com/fr/definition/deep-learning-reseau-neuronal profond, Consulté « Mai 2019 »

[26] Différence entre Intelligence Artificielle, Machine Learning et Deep Learning, penseeartificielle.fr, Lambert R.,20 décembre 2018,http://penseeartificielle.fr/difference-intelligence-artificielle-machine-learning-deep-learning, Consulté «Mai 2019»

[27] artificial neural network (ANN), Margaret Rouse,july 2018, TechTarget, https://searchenterpriseai.techtarget.com/definition/neuralnetwork, Consulté «Mai 2019 »

[29] Introduction to FeedForward Neural Networks,Yash Upadhyay, Mars 2019 , https://towardsdatascience.com/feed-forward-neural-networks-c503faa46620, Consulté « Mai 2019 »

[37] Dean, J., Corrado, (2012). Large Scale Distributed Deep Networks. NIPS 2012 : Neural Information Processing Systems, 1–11. http://papers.nips.cc/paper/4687-large-scale-distributed-deep-networks.pdf, Consulté « Mai 2019 »

[40] Deep Learning Book, November 2016 , by Ian Goodfellow, Yoshua Bengio, And Aaron Courville ,http://www.deeplearningbook.org, Consulté « Mai 2019 »

[41] Andrew Ng, Jiquan Ngiam, Chuan Yu Foo, 26 Sep 2013,Deep Learning Tutorial, http://ufldl.stanford.edu , Consulté « Mai 2019 »

[45] Niklas Donger, February 26,2018, Recurrent Neural Networks and LSTM, https://towardsdatascience.com/recurrent-neural-networks-and-lstm-4b601dd822a5 Consulté « Mai 2019 »

[46] Long Short-Term Memory (LSTM), https://www.techopedia.com/definition/33215/long-short-term-memory-lstm , Consulté « Mai 2019 »

[47] Understanding LSTM Networks, Christopher Olah, August 27, 2015 http://colah.github.io/posts/2015-08-Understanding-LSTMs/, Consulté « Mai 2019 ».

[50] Python Language Copyright c 2001-2018 Python Software Foundation. All rights reserved. Licence : https://docs.python.org/3/license.html , Consulté « Mai 2019 »

[51] Keras Framework: https://keras.io/. Github: https://github.com/keras-team/keras Consulté « juillet 2019 »

[52] Keras Licence : https://github.com/keras-team/keras/blob/master/LICENSE, Consulté « juillet 2019 »

[53] NLTK , https://www.nltk.org/ ,Consulté « juillet 2019 »

[54] Pandas, https://pandas.pydata.org/ ,Consulté « juillet 2019 »