

Réf : Ing-GI-2016-23

Rapport de Projet de Fin d'Études

Pour obtenir le

Diplôme d'Ingénieur en Génie Informatique

Option : Génie Logiciel & Informatique Décisionnelle

Présenté et soutenu publiquement le 09 septembre 2016

Par

Mariem KAMMOUN

Réalisation d'une plateforme d'apprentissage linguistique de textes français par Réseaux de Neurones

Composition du jury

Madame	Emna SOUISSI	Présidente
Madame	Narjes HACHANI	Rapporteuse
Monsieur	Makram MESTIRI	Encadrant Entreprise
Madame	Afef KACEM ECHI	Encadrante ENSIT

Année universitaire : 2015-2016

Dédicaces

Du profond de mon cœur, je dédie ce travail à tous ceux qui me sont chers,

À MES CHERS PARENTS

Que ce travail soit l'expression de ma reconnaissance pour vos sacrifices consentis, votre soutien moral et matériel que vous n'avez cesse de prodiguer. Vous avez tout fait pour mon bonheur et ma réussite. Que dieu vous préserve en bonne santé et vous accorde une longue vie.

À MON CHER FRÈRE

Que j'aime tant et je le souhaite un bon avenir et une vie pleine de succès.

À toute ma famille et tous mes amis et tous ceux qui n'ont jamais cessé de croire en moi, m'ont soutenu et encouragé.

Je dédie ce modeste travail, le fruit de mes efforts et de longues années d'études, qu'ils y trouvent le couronnement de leur assistance et l'expression profonde de ma gratitude.

Mariam.

Remerciements

Au terme de ce projet de fin d'études, je remercie Dieu le tout puissant de m'avoir donné la santé et la volonté d'entamer et de terminer ce travail.

Tout d'abord, je m'adresse aux membres de l'honorable jury que je remercie d'avoir accepté d'examiner ce rapport.

Je remercie vivement mes encadrants **Mr. Makrem MESTIRI** et **Dr. Afef KACEM ECHI**. Ce travail ne serait pas aussi riche et n'aurait pas pu voir le jour sans leur aide, leur encadrement exceptionnel et leur disponibilité.

Je remercie également tous les membres de la société Cube 3D Technology pour avoir fourni un cadre de travail très convivial.

Je ne laisserai pas cette occasion passer sans exprimer ma reconnaissance envers tous mes professeurs et au personnel de l'École Nationale Supérieure d'Ingénieurs de Tunis.

Table des matières

Introduction générale	1
Chapitre 1 : Présentation du cadre du projet	3
Introduction	4
1.1 Cadre général du projet	4
1.2 Présentation de l'organisme d'accueil	4
1.3 Présentation de la problématique	5
1.4 Méthodologie adoptée	6
Conclusion	7
Chapitre 2 : État de l'art	8
Introduction	9
2.1 Apprentissage automatique	9
2.1.1 Apprentissage profond	10
2.1.2 Domaines d'application de l'apprentissage automatique	10
2.1.3 Types d'apprentissage	11
2.2 Linguistique informatique	12
2.2.1 Applications de TALN	13
2.2.2 Méthodes de TALN	13
2.3 Aperçu sur quelques applications existantes reliant l'apprentissage et le TAL.....	14
2.3.1 Langue comme un objet statistique.....	14
2.3.2 Apprentissage d'une phrase.....	15
2.4 Réseaux de neurones.....	16
2.4.1 Introduction.....	16
2.4.1.1 Origines.....	16
2.4.1.2 Définition.....	17
2.4.1.3 Neurone biologique.....	17
2.4.1.4 Neurone artificiel.....	18
2.4.2 Architectures des réseaux de neurones artificiels.....	19
2.4.3 Apprentissage.....	20
2.4.3.1 Définition	20

2.4.3.2 Étapes d'apprentissage	21
2.4.3.3 Types d'apprentissage.....	22
2.4.4 Perceptron Multicouche.....	23
2.4.4.1 Perceptron simple	23
2.4.4.2 Structure du Perceptron Multicouche	24
2.4.4.3 Algorithme d'apprentissage de PMC: Règle de rétro-propagation du gradient...	25
2.4.5 Exemples généraux d'application des RN.....	26
Conclusion.....	26
Chapitre 3 : Approche et solution proposées.....	27
Introduction.....	28
I. Analyse et spécification des besoins.....	28
I.1 Spécification des besoins.....	28
I.1.1 Spécification des besoins fonctionnels.....	28
I.1.2 Spécification des besoins non fonctionnels.....	28
I.2 Analyse des besoins.....	29
I.2.1 Identification des acteurs.....	29
I.2.2 Langage de modélisation.....	29
I.2.3 Diagrammes des cas d'utilisation.....	29
I.2.3.1 Diagramme des cas d'utilisation général	30
I.2.3.2 Description détaillée des cas d'utilisation.....	30
II. Conception.....	34
II.1 Conception générale.....	34
II.1.1 Architecture globale de la solution.....	34
II.1.2 Choix de l'approche Orientée Objet.....	35
II.2 Conception détaillée.....	35
II.2.1 Conception de la base de données.....	35
II.2.1.1 Modèle relationnel de la base de données.....	35
II.2.1.2 Descriptions des tables.....	36
II.2.1.3 Diagramme de classes.....	37
II.2.2 Diagrammes de séquences.....	38
II.2.2.1 Diagramme de séquence Saisir une phrase, ses mots et leurs propriétés.....	38

II.2.2.2 Diagramme de séquence Afficher les statistiques.....	40
II.2.2.3 Diagramme de séquence Afficher les statistiques du chat.....	40
III. Procédure de développement d'un réseau de neurones.....	41
Conclusion.....	44
 Chapitre 4 : Réalisation.....	 45
Introduction.....	46
4.1 Environnement de travail.....	46
4.1.1 Environnement matériel.....	46
4.1.2 Environnement logiciel.....	46
4.2 Choix techniques.....	46
4.2.1 Choix du langage de programmation java.....	46
4.2.2 Choix de l'environnement de développement Netbeans.....	47
4.2.3 Choix de la base de données MySQL.....	47
4.2.4 Choix du chat Spark.....	47
4.2.5 Choix de Neuroph Studio.....	47
4.3 Phases d'implémentation.....	48
4.3.1 Collecte et recueil de données.....	48
4.3.2 Statistiques et calcul de taux d'apprentissage.....	52
4.3.3 Préparations des données, construction de la base d'apprentissage et choix du RNA.....	63
4.3.4 Création du RNA et apprentissage.....	68
4.3.5 Tests et interprétation.....	73
4.3.6 Construction de la base de test.....	74
4.3.7 Interprétation des résultats.....	79
Conclusion.....	79
Conclusion générale.....	80

Liste des figures

Figure 1.1 : Cycle prototypage évolutif.....	6
Figure 2.1: Exemples d'applications en TALN.....	13
Figure 2.2: Graphique à barres présentant les fréquences moyennes de lettres pour l'anglais, le français et le polonais.....	15
Figure 2.3: Exemple d'analyse d'une phrase avec l'apprentissage.....	16
Figure 2.4: Neurones biologiques.....	17
Figure 2.5: Mise en correspondance entre neurone biologique et neurone artificiel.....	18
Figure 2.6: Neurone artificiel et les notations utilisées.....	18
Figure 2.7: Fonctions d'activation les plus utilisées.....	19
Figure 2.8: Réseau multicouche.....	19
Figure 2.9: Réseau à connexions locales.....	20
Figure 2.10: Réseau à connexions récurrentes.....	20
Figure 2.11: Réseau à connexions complètes.....	20
Figure 2.12: Le Perceptron simple.....	23
Figure 2.13: Architecture d'un Perceptron Multicouches à une seule couche cachée.....	24
Figure 3.1: Diagramme de cas d'utilisation du système.....	30
Figure 3.2: Architecture globale du système.....	34
Figure 3.3: Diagramme de classes.....	38
Figure 3.4: Diagramme de séquence de Saisir une phrase, ses mots et leurs propriétés.....	39
Figure 3.5: Diagramme de séquence d'Afficher les statistiques.....	40
Figure 3.6: Diagramme de séquence d'Afficher les statistiques du chat.....	41
Figure 4.1: Le menu de notre plateforme d'apprentissage linguistique.....	48
Figure 4.2: La boîte de dialogue pour vérifier le nombre de mots d'une phrase.....	49
Figure 4.3: Un exemple décrivant le cas d'utilisation "Saisir une phrase, ses mots et leurs propriétés".....	49
Figure 4.4: La phrase saisie est ajoutée à la table "phrase" de la BD.....	50
Figure 4.5: Les mots de la phrase sont ajoutés à la table "mots" de la BD.....	50
Figure 4.6: Ajout d'un mot avec un pourcentage incorrect.....	51

Figure 4.7: Exemple de modification d'une propriété (la fonction) d'un mot.....	51
Figure 4.8: La modification de la fonction du mot dans la BD.....	51
Figure 4.9: La suppression d'un mot dans la table de l'interface.....	52
Figure 4.10: La suppression d'un mot dans la BD.....	52
Figure 4.11: L'interface du cas d'utilisation "Afficher les statistiques"	52
Figure 4.12: Les statistiques des mots d'une phrase.....	53
Figure 4.13: Une fonction que prend un mot et son nombre d'apparitions.....	53
Figure 4.14: Les boutons des graphes de statistiques.....	53
Figure 4.15: Histogramme de nombre d'apparitions des mots.....	54
Figure 4.16: Histogramme de taux d'apprentissage des mots.....	54
Figure 4.17: Camembert de nombre d'apparitions de fonctions pour chaque mot.....	55
Figure 4.18: Imprime-écran pour l'ajout d'un nouveau mot.....	55
Figure 4.19: Calcul du taux d'apprentissage d'un nouveau mot.....	57
Figure 4.20: Taux d'apprentissage d'un nouveau mot.....	58
Figure 4.21: Les taux d'apprentissage de notre exemple.....	58
Figure 4.22: Exemple 1 de découpage en mots.....	59
Figure 4.23: Exemple 2 de découpage en mots.....	59
Figure 4.24: Interface pour afficher les statistiques d'une phrase écrite en chat.....	60
Figure 4.25: Ouverture du chat.....	60
Figure 4.26: Connexion au chat et les différentes situations d'un utilisateur du chat	61
Figure 4.27: Le chat.....	61
Figure 4.28: Les phrases dans la BD.....	61
Figure 4.29: Autre discussion et ses phrases dans la BD	62
Figure 4.30: Liste de smileys offerte par Spark.....	62
Figure 4.31: Les phrases dans la plateforme.....	63
Figure 4.32: Exemple qui traite les smileys.....	63
Figure 4.33: Notre dictionnaire (partie 1).....	65
Figure 4.34: Notre dictionnaire (partie 2).....	66
Figure 4.35: Notre dictionnaire (partie 3).....	67

Figure 4.36: Création du réseau de neurones (1): choix de son nom et son type.....	68
Figure 4.37: Création du réseau de neurones (2): choix de nombre de neurones dans chaque couche, la fonction de transfert et la règle d'apprentissage.....	69
Figure 4.38: Imprime-écran du réseau de neurones.....	70
Figure 4.39: Création de l'ensemble d'apprentissage	71
Figure 4.40: Paramètres d'apprentissage.....	72
Figure 4.41: Graphe d'erreur totale de notre réseau.....	72
Figure 4.42: L'erreur quadratique moyenne totale de notre réseau après apprentissage.....	73
Figure 4.43: Interface pour tester le réseau de neurones.....	75
Figure 4.44: Table des entrées de la base de test.....	76
Figure 4.45: Table des sorties de la base de test.....	76

Liste des tableaux

Tableau 2.1: Représentation d'un tableau à deux entrées: les attributs et les données.....	21
Tableau 3.1: Tableau descriptif du cas d'utilisation "Saisir une phrase, ses mots et leurs propriétés"	30
Tableau 3.2: Tableau descriptif du cas d'utilisation "Mise à jour d'une phrase et/ou ses mots et/ou leurs propriétés"	31
Tableau 3.3: Tableau descriptif du cas d'utilisation "Afficher les statistiques"	32
Tableau 3.4: Tableau descriptif du cas d'utilisation " Générer des phrases selon un dictionnaire bien précis pour le test d'un RN "	33
Tableau 4.1: Table de statistiques.....	56
Tableau 4.2: Base d'apprentissage pour le réseau de neurones.....	64
Tableau 4.3: Résultats de différents apprentissages de quelques réseaux de neurones.....	73
Tableau 4.4: Base de généralisation pour le réseau de neurones.....	75
Tableau 4.5: Comparaison entre les sorties trouvées et les sorties désirées pour le RN avec 100 neurones cachés, LR= 0.5 et momentum = 0.7.....	77

Liste des acronymes

AA	Apprentissage automatique
BD	Base de Données
DL	Deep Learning
IA	Intelligence artificielle
IHM	Interface Homme-Machine
ML	Machine Learning
MLP	Multi-Layer Perceptron
MVC	Modèle Vue Contrôleur
PMC	Perceptron Multicouches
RN	Réseaux de Neurones
RNA	Réseaux de Neurones Artificiels
SLP	Simple Layer Perceptron
TAL	Traitement Automatique de Langue
TALN	Traitement Automatique du Langage Naturel

Introduction générale

Aujourd'hui, le domaine de l'intelligence artificielle (IA) est en plein essor. Il revient néanmoins en force dans de multiples laboratoires prestigieux et de nombreuses start-up.

L'intelligence artificielle est un ensemble de techniques permettant à des machines d'accomplir des tâches et de résoudre des problèmes normalement réservés aux humains et à certains animaux [1].

S'il y a un réel renouveau des recherches sur l'IA et des peurs qu'elles suscitent, c'est qu'il y'a beaucoup de changements, comme l'apparition de nouvelles technologies d'apprentissage automatique (Machine Learning ou ML), la démocratisation des moyens de calculs puissants, l'accès à des données massives (Big Data), etc. Sans parler des progrès enregistrés dans la conception et la fabrication des puces électroniques qui imitent le fonctionnement du cerveau. Des géantes entreprises, comme IBM, Google et Facebook, se sont bien décidées à intégrer toutes ces avancées technologiques pour développer des applications qui vont de la médecine à la robotique, en passant par les fonctions d'assistants numériques personnels, les réseaux sociaux, etc.

Avec le grand retour de l'IA, un autre terme revient lui aussi sur le devant de la scène: les réseaux de neurones. Ces logiciels qui imitent le fonctionnement du cerveau, ont eux aussi une longue histoire. Après plusieurs vagues d'engouements et de déception sur leurs capacités à effectuer des tâches vraiment utiles, ils accompagnent aujourd'hui les avancées du “Deep Learning” ou “apprentissage profond”. Il s'agit d'une technique de Machine Learning (apprentissage automatique) née vers 2006, qui utilise des réseaux de neurones à multiples couches pour effectuer une série de traitements hiérarchisés, dans le but de classer des milliers d'objets en catégories, sans critères prédéfinis [2].

C'est l'apprentissage qui anime les systèmes de toutes les grandes entreprises d'internet. En effet, elles utilisent les technologies de reconnaissance de formes au sens large, comme la reconnaissance optique de caractères (OCR), celle des visages (reconnaissance faciale), de reconnaissance vocale et de reconnaissance du langage naturel, ou la traduction automatique. Des efforts considérables de R&D (recherche et développement) sont consacrés au traitement du langage naturel: la compréhension de texte, les systèmes de question-réponse, les systèmes de dialogue pour les agents virtuels et la traduction automatique. Dans ce domaine, la

révolution de l'apprentissage profond a été annoncée, mais n'est pas encore achevée. Néanmoins, nous assistons à des progrès rapides [1].

Dans ce contexte s'inscrit notre projet de fin d'études qui consiste à réaliser une plate-forme d'apprentissage linguistique de textes français par réseaux de neurones au profit de la société Cube 3D Technology. Il entre dans le cadre de développement d'un assistant virtuel. Afin de mieux clarifier et structurer notre travail, notre rapport s'articule autour de quatre chapitres. Le premier chapitre décrit le cadre général du projet et de la problématique. Le deuxième chapitre présente les notions théoriques pour la compréhension de ce projet de recherche. Nous présentons d'abord l'apprentissage automatique, puis la linguistique informatique. Nous définissons aussi les concepts de base des réseaux de neurones. Le troisième chapitre est consacré à présenter la solution proposée : analyse et la spécification des besoins et approche conceptuelle pour le développement de l'application. Le quatrième chapitre couronne le travail réalisé en exposant l'environnement matériel et logiciel utilisé ainsi qu'une analyse des résultats récoltés. Ce rapport est clôturé par une conclusion générale qui présente le bilan de ce projet, les apports de point de vue savoir-faire ainsi que d'éventuelles perspectives.

Chapitre 1

Présentation du cadre du projet

Introduction

Dans ce premier chapitre, nous commençons par présenter brièvement l'organisme d'accueil au sein duquel nous avons effectué le stage relatif au présent projet. La suite du chapitre est consacrée à présenter la problématique du projet.

1.1 Cadre général du projet

Ce travail s'inscrit dans le cadre du projet de fin d'études d'ingénieur en génie informatique à l'École Nationale Supérieure des Ingénieurs de Tunis. Il a été effectué au sein de l'organisme Cube 3D Technology.

1.2 Présentation de l'organisme d'accueil

Cube 3D Technology est une société spécialisée dans les nouvelles technologies et leur développement. Forts de leur savoir-faire dans différents domaines, ses équipes d'ingénieurs et de techniciens spécialisés ont fait de Cube une société tunisienne capable de produire des réalisations à l'échelle mondiale et reconnue par un large public.

Étant conscient qu'aujourd'hui les entreprises ne voient plus la technologie comme une simple dépense mais plutôt comme un investissement à court et long terme, leur permettant d'augmenter la notoriété de leur marque, de conquérir de nouveaux clients et ainsi de vendre plus, Cube développe et propose aux entreprises un ensemble de solutions permettant à ces dernières d'atteindre leurs objectifs en faisant passer leur message.

De la production vidéo à la réalisation de jeux vidéo en 3D en passant par la réalisation et l'installation de laboratoire dédié à la réalité virtuelle et augmentée, Cube aide ses clients à faire passer leurs messages d'une façon innovante et ciblée. Média à part entière, les jeux vidéo font bénéficier les clients de Cube du " in-game advertising " en délivrant un message pour un produit ou une marque en proposant au client, annonceur, d'acheter ou de louer un espace de communication au sein des visualisations du jeu vidéo. Nous voyons ainsi des sportifs en compétition portant des maillots à l'image de la marque, une voiture de course aux couleurs de la marque, une ville où se déroule l'action avec des panneaux publicitaires au nom de la marque, etc.

Au delà de "l'in-game advertising", Cube offre également à ses clients des solutions de type "Advergaming " en mettant en scène les produits de la marque autour de jeux vidéo sur les sites internet des annonceurs. Ainsi le consommateur a la possibilité de se divertir en jouant tout en ayant la possibilité de gagner de coupons de réduction par exemple pour acheter en

magasin ou directement grâce à l'e-dinar. L'Advergaming permet donc d'augmenter la notoriété d'une marque et d'augmenter également les ventes de produits de cette dernière [3].

Fiche d'identité de Cube 3D Technology

Raison social : CUBE 3D TECHNOLOGY



Logo :

Adresse : 36, Rue Rchid El Ayachi, Ennasr 2

Secteur d'activité : Informatique

Brève description : Société leader dans le développement de jeux vidéo en Tunisie avec plusieurs réalisations pour plusieurs sociétés de renommées mondiales.

Responsable : Mokhtar Mestiri

Produits : Citroën immersion days, Arum, Gordion and the rise of Thysdrus, Nichan, Carthage, Smart Medecine et Muslim Way sur Androïd, Jahjouh wa arous al bahr, Vidéo 3D, documentaires, etc.

Clients : Citroën, Orange, Tunisie Telecom, Ooredoo, Orangina, Tigo, Gonser group, ID COM, GET Wireless, CGS 3D Visual Effects, etc.

Téléphone : 70039601 ; **Fax :** 70039601

Email : contact@cube3dtech.com ; **Site web :** www.cube3dtech.com

1.3 Présentation de la problématique

Il s'agit de réaliser une plateforme d'analyse de l'importance linguistique des mots dans une phrase. À partir de la saisie d'une phrase, le système doit pouvoir identifier chaque mot, sa fonction, sa position et sa pondération qui reflète son importance linguistique dans cette phrase. Cette pondération permettra la compréhension par un système intelligent qui analyse le texte par des agents afin de comprendre sa signification. Ce projet entre dans le cadre de développement d'un avatar virtuel 3D capable d'interagir avec un interlocuteur et de l'assister afin d'effectuer plusieurs tâches.

1.4 Méthodologie adoptée

Dans le domaine de l'ingénierie de logiciel, il est indispensable de définir une méthode de travail et suivre le processus de développement le mieux adapté pour la réalisation de l'application. Pour garantir un niveau de qualité acceptable et éviter tout débordement au niveau du délai, et devant le grand nombre des méthodologies existantes, notre choix s'est porté sur le prototypage évolutif pour plusieurs raisons.

En développement évolutif, un logiciel est rapidement prototypé, ses différentes abstractions étant implantées le plus simplement et rapidement possible. Au besoin, comme c'est souvent le cas ces abstractions sont revues au fur et à mesure de l'évolution du projet. Lorsqu'une abstraction a démontré sa valeur, son implantation est améliorée en fonction de son importance et des compromis habituels (performance, coût, etc.).

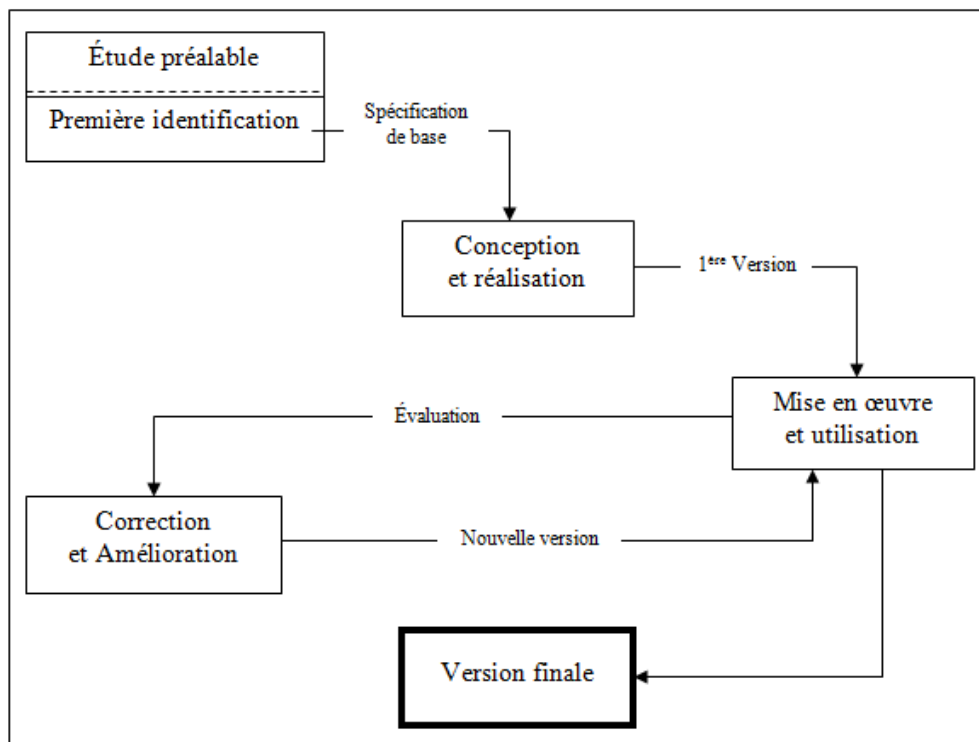


Figure 1.1: Cycle prototypage évolutif

Le long de la réalisation de ce projet, nous avons établi des prototypes évolutifs construits les uns à partir des autres et continuellement modifiés en fonction de l'avancement dans la connaissance du domaine. Chaque prototype fait l'objet d'un cycle spécification - conception - implantation - test. Ainsi, d'itérations en itérations, nous avons obtenu une version finale, complète et fidèle par rapport aux attentes formulées.

Conclusion

Dans ce chapitre, nous avons situé notre projet dans son cadre général à savoir la société d'accueil, identifié la problématique soulevée et présenté la méthodologie de développement adoptée. Le chapitre suivant sera consacré à la présentation des différents concepts théoriques sur lesquels se base ce projet.

Chapitre 2

État de l'art

Introduction

Préalablement à toute élaboration d'un projet, il est nécessaire d'éclaircir ses concepts clés. Dans ce cadre, ce chapitre vient de limiter le périmètre du travail en présentant une analyse approfondie de la documentation élaborée pour la compréhension de ce projet. Nous présentons d'abord l'apprentissage automatique puis la linguistique informatique. Enfin, nous exposons les différents concepts de base des réseaux de neurones.

2.1 Apprentissage automatique

Aujourd'hui, le Machine Learning ou l'apprentissage automatique est un domaine de recherche en plein essor. Il a connu des avancées significatives ces dernières années, poussé notamment par de grandes entreprises aux moyens inédits. C'est une discipline qui a pris tout son sens avec l'arrivée des Big Data; avec comme icône médiatique le Google Brain, qui a réussi la prouesse, en 2012, de découvrir le concept de chat en analysant des millions d'images issues du Web [4].

L'engouement récent pour cette discipline tient en grande partie à une observation qui a surpris les spécialistes en IA eux-mêmes: « L'utilisation conjointe de quantités massives d'informations et d'algorithmes d'apprentissage relativement simples rend possible la solution de problèmes considérés il y a peu comme inaccessibles » [5].

Vu que l'apprentissage est souvent considéré comme la caractéristique principale de l'IA, de nombreuses définitions et citations ont été proposées, parmi lesquelles nous citons :

☆ « L'apprentissage automatique, c'est la capacité d'un ordinateur à apprendre sans avoir été explicitement programmé » Arthur Samuel [6].

☆ « L'apprentissage dénote des changements dans un système qui lui permet de faire la même tâche plus efficacement la prochaine fois » Herbert Simon [7].

☆ « Cette notion englobe toute méthode permettant de construire un modèle de la réalité à partir de données, soit en améliorant un modèle partiel ou moins général, soit en créant complètement le modèle. Il existe deux tendances principales en apprentissage, celle issue de l'intelligence artificielle et qualifiée de « symbolique », et celle issue des statistiques et qualifiée de « numérique » (Cornuéjole & Miclet, 2002) [B2].

La définition suivante est proposée en référence [9] : « L'apprentissage automatique (machine Learning en anglais), un des champs d'étude de l'intelligence artificielle, est la discipline

scientifique concernée par le développement, l'analyse et l'implémentation de méthodes automatisables qui permettent à une machine (au sens large) d'évoluer grâce à un processus d'apprentissage, et ainsi de remplir des tâches qu'il est difficile ou impossible de remplir par des moyens algorithmiques plus classiques ».

Donc, dans l'apprentissage automatique, un programme analyse un ensemble de données afin de tirer des règles qui permettront de tirer des conclusions sur de nouvelles données [10].

D'après tout ce que nous avons présenté, nous constatons que l'objectif de ML est de concevoir des programmes pouvant s'améliorer automatiquement avec l'expérience.

2.1.1 Apprentissage profond

L'apprentissage profond ou Deep Learning (DL), est un ensemble de techniques d'apprentissage automatique, qui a permis des avancées importantes en IA dans les dernières années. Il est basé sur les réseaux de neurones artificiels, composés de milliers d'unités (les neurones) qui effectuent chacune de petites opérations simples. Les résultats d'une première couche de neurones servent d'entrée aux calculs d'une deuxième couche et ainsi de suite [10].

2.1.2 Domaines d'application de l'apprentissage automatique

Comme l'apprentissage automatique a connu un progrès rapide, il se trouve au carrefour de nombreux domaines et il est devenu très utilisé dans plusieurs applications parmi lesquelles nous citons :

- ❖ Le développement de système de reconnaissance vocale par exemple la reconnaissance vocale Siri d'Apple.
- ❖ La reconnaissance de formes au sens large comme la reconnaissance optique de caractères (OCR) ou celle des visages et d'images: des premières couches d'unités (les neurones) identifient des lignes, des courbes, des angles... des couches supérieures identifient des formes, des combinaisons de formes, des objets et des contextes...
- ❖ Dans le domaine de fouilles de textes: La reconnaissance du langage naturel et son traitement. Par exemple, pour taper du texte, les smartphones suggèrent des mots.
- ❖ La traduction automatique : la traduction en temps réel de Skype, Google Traduction.
- ❖ Dans le champ des études sociales : L'analyse de la personnalité des utilisateurs des réseaux sociaux et l'analyse des sentiments qui cherche à classer des documents selon leur tonalité émotionnelle dominante.
- ❖ Dans la finance et l'e-business : La détection de fraude par cartes de crédit dans les transactions bancaires ou l'estimation du risque de non-remboursement d'un prêt en

fonction du passé financier d'un demandeur de crédit, l'analyse prédictive d'un panier d'achat d'un consommateur et l'analyse des marchés boursiers.

- ❖ La classification des e-mails dans gmail, les filtres anti-spam.
- ❖ La voiture autonome de Google.
- ❖ Les interactions sociales des robots, les jeux.
- ❖ Les modèles d'apprentissage machine ont notamment permis le développement du diagnostic médical, la bioinformatique, la biochimie, la biométrie, etc.

Le Machine Learning constitue donc une vraie révolution dans le logiciel: probabilités et statistiques sur les masses de données permettent au logiciel d'apprendre et non seulement de calculer. Nouvelle frontière, nouvelle façon de penser le logiciel et le programmer, c'est une révolution qui replace la Big Science au cœur de l'innovation [7].

2.1.3 Types d'apprentissage

Il existe plusieurs types d'apprentissage. Nous discernons habituellement trois types :

➤ L'apprentissage supervisé

Le système obtient la valeur à retenir par un élément extérieur, généralement un expert ou un utilisateur. Imaginons un cas (1) où le système doit décider de tourner une clé vers la gauche ou vers la droite. Toutefois, il ignore que l'objectif est de déverrouiller la porte, il lui sera alors fourni par l'utilisateur. Une fois cet objectif en main, le système essaiera de tourner la clé et déterminera le bon sens de rotation permettant d'aboutir à l'état indiqué par le superviseur [B3]. L'apprentissage supervisé passe par deux phases. Une phase d'apprentissage où un expert transmet des données rattachées à des étiquettes au programme, et une phase où le programme détermine un comportement en fonction d'une situation et de ce qu'il a appris [B4].

➤ L'apprentissage non supervisé

Le programme apprend sans expert en fonction des données non étiquetées qu'il perçoit lors de son exécution [B4]. Ainsi dans le cas (1) vu précédemment, le système va tourner la clé dans un sens puis dans l'autre et déverrouiller la porte, mais sans comprendre que ce qu'il a fait est correct. Un apprentissage artificiel sans aucune supervision ne saura jamais si ses actions sont correctes ou non. Il est intéressant d'utiliser ce type de système en compétition avec d'autres ou de lui fournir des statistiques comme support d'apprentissage [B3].

➤ **L'apprentissage par renforcement**

L'apprentissage par renforcement qui fonctionne également sans superviseur, apprend grâce au renforcement c.à.d. l'algorithme apprend de l'environnement dans lequel il évolue. Le système est guidé au moyen de récompenses qui sont attribuées selon ses actions. Le système cherche au travers d'expériences itérées, une stratégie optimale dans le sens de maximiser la somme des récompenses au cours du temps [B3].

Pour notre projet, nous avons opté pour l'apprentissage supervisé. Comme notre objectif est la réalisation d'une plate-forme d'apprentissage linguistique de phrases en français, il importe d'introduire la linguistique informatique.

2.2 Linguistique informatique

La linguistique informatique et le traitement automatique des langues (TAL) désignent l'application de programmes et techniques informatiques à tous les aspects du langage humain depuis la reconnaissance de la parole jusqu'à l'analyse sémantique du contenu d'un texte. Il s'agit à la fois d'un domaine technologique et d'un domaine de recherche [11].

D'après [12], la linguistique informatique constitue des champs d'études regroupant plusieurs disciplines, et qui se base sur la modélisation statistique du langage humain, qui est établi dans une approche informatique.

Selon [13], la linguistique informatique peut être définie comme l'application des méthodes de l'intelligence artificielle pour traiter les questions linguistiques. En d'autres termes, c'est un domaine interdisciplinaire dans lequel interviennent des linguistes, des spécialistes en informatique, des spécialistes en logique, etc.

Le traitement automatique du langage naturel (TALN) est une discipline qui a pour objectif de modéliser grâce à l'informatique le langage qu'il soit écrit ou parlé. Cette discipline a réellement commencé lors de la guerre froide quand les américains avaient besoin de comprendre les messages russes. Ainsi est né le premier traducteur automatique, très rudimentaire, avec ses 250 mots et ses 6 règles de grammaire. La principale difficulté de cette discipline est de traiter les langages qui sont des langages ambigus par l'informatique qui est un langage binaire et donc non-ambigu. Comment par exemple, déterminer automatiquement dans quels contextes le mot "avocat" se rapporte au domaine juridique ou au domaine alimentaire? Aujourd'hui, le TALN a bien évolué mais il reste encore beaucoup de découvertes à faire [14].

2.2.1 Applications du TALN

Le langage étant présent partout, écrit ou oral, les applications du TALN sont multiples et variées. Aussi, depuis l'avènement d'internet, nous dénombrons près de 920 millions de sites Web. Cette profusion d'information (et notamment de texte) a vu la nécessité de créer de nouveaux outils de TAL afin de la gérer le mieux possible [14].

Voici quelques exemples d'applications en TALN :

- La traduction automatique,
- La correction orthographique,
- La recherche d'information, les moteurs de recherche,
- Le résumé de texte,
- La génération de texte,
- La synthèse de la parole,
- La reconnaissance vocale.

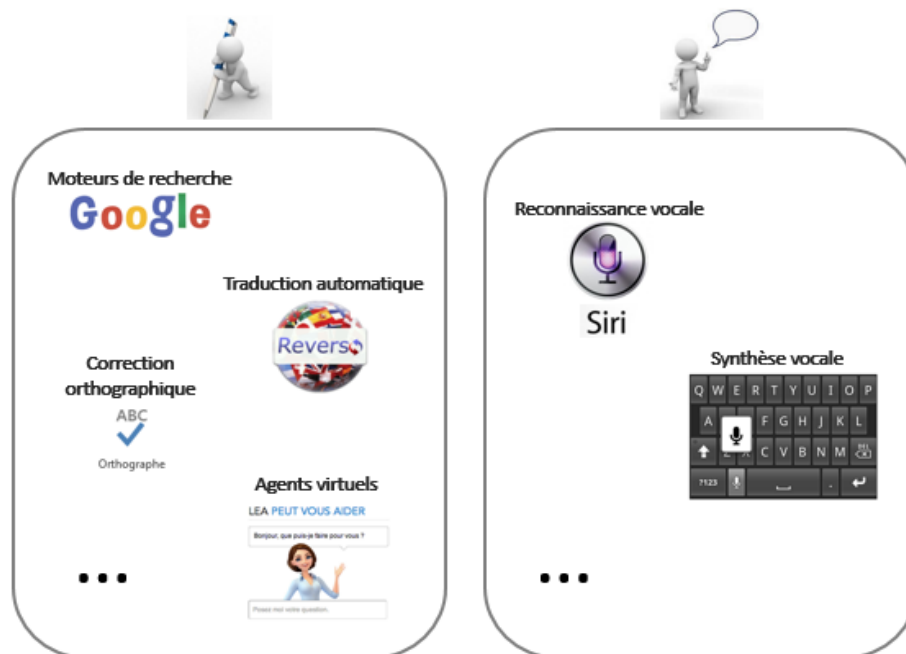


Figure 2.1: Exemples d'applications en TALN

2.2.2 Méthodes de TALN

En TAL, deux méthodes peuvent être employées, soit de façon indépendante ou hybride: les méthodes statistiques et linguistiques.

★ Méthodes statistiques

Le principe de ces méthodes est de faire de la langue un objet mathématique. Nous appliquons alors différentes formules probabilistes et statistiques aux données de la

langue (mots, phrases...). Nous allons par exemple, s'intéresser à la fréquence d'apparition des mots dans un texte pour en déduire les thèmes principaux de celui-ci. L'avantage de ces méthodes est qu'elles sont rapides à implémenter et les résultats sont très bons. Le seul inconvénient est qu'elles ne sont efficaces que sur des corpus des données de très grandes tailles.

★ Méthodes linguistiques

Ces méthodes vont utiliser tous les différents niveaux d'analyse utilisés en linguistique comme la phonétique, la syntaxe ou la sémantique.

Les méthodes linguistiques ont l'avantage d'être très performantes mais elles sont très coûteuses en temps. En effet, constituer des lexiques ou des grammaires pour décrire de façon la plus exhaustive possible une langue peut prendre des années [14].

Pour notre projet, nous choisissons la méthode statistique.

2.3 Aperçu sur quelques applications existantes reliant l'apprentissage et le TAL

Il existe de nombreux exemples pour traiter automatiquement le langage en écrivant des programmes qui s'améliorent par l'expérience afin de gagner le temps de développement et l'efficacité. Parmi ces exemples, nous retenons:

2.3.1 Langue comme un objet statistique

Grâce à l'internet, la communication à l'échelle mondiale fait partie intégrante de la vie quotidienne de bon nombre de personnes, mais il a également soulevé le problème de la langue. Les outils d'aide à la traduction, peuvent aider à franchir cette barrière linguistique, mais pour que de tels outils fonctionnent, ils ont besoin de connaître la langue dans laquelle est écrit le passage d'un texte.

Une façon de déterminer la langue consiste à examiner la fréquence des lettres qui apparaissent dans un texte. Alors que cette approche peut sembler naïve en matière de détection de langues, elle s'est révélée très efficace. Pour de nombreuses langues européennes, il suffit de regarder la fréquence des lettres de A à Z, même si certaines langues utilisent d'autres lettres. En utilisant un réseau de neurones (que nous allons détailler ensuite ses concepts de base), nous avons pu générer une représentation graphique des fréquences de chaque lettre pour un fichier d'apprentissage contenant des textes écrits dans différentes

langues sachant que nous avons pris en considération que les lettres comprises entre A et Z car elles sont communes entre ces trois langues. La figure 2.2 montre cette représentation.

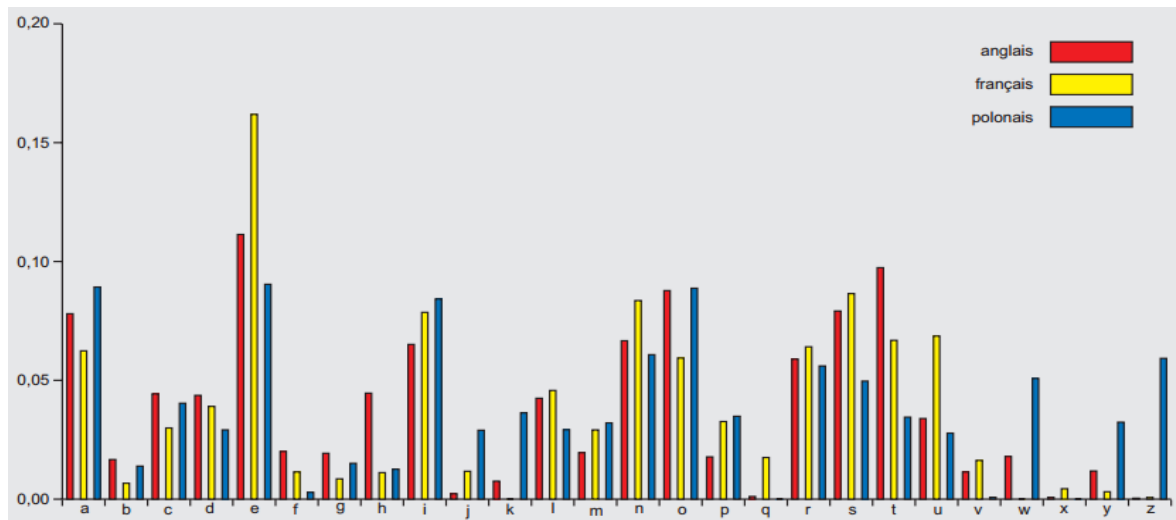


Figure 2.2: Graphique à barres présentant les fréquences moyennes de lettres pour l'anglais, le français et le polonais

Une analyse approfondie de ce fichier fait ressortir des tendances nettes: l'anglais présente ainsi plus de H que les deux autres langues, le français ne présente presque pas de K et le polonais a plus de W et de Z que les deux autres langues. Donc avec ces données, en observant les fréquences des caractères, nous pouvons identifier automatiquement la langue [15].

2.3.2 Apprentissage d'une phrase

Pour faire l'apprentissage, comme nous avons dit auparavant, nous avons deux phases à faire:

- ★ Phase d'apprentissage: C'est la phase où nous fournissons à la machine un corpus d'apprentissage: un grand corpus avec des textes et analyses correctes. Par exemple, pour un analyseur morphosyntaxique, c'est un grand corpus de phrases qui sont analysées tout en connaissant la catégorie de chaque mot. Et c'est l'algorithme qui calcule des fréquences souvent des uni-, bi-, ou tri- grammes dans le corpus d'apprentissage.
- ★ Phase de test: Dans cette phase, nous testons le système sur un petit corpus de test. Pour ce corpus, nous connaissons l'analyse correcte, mais la machine doit l'obtenir uniquement à partir du texte. Nous comparons la sortie du système avec l'analyse correcte et si le résultat n'est pas satisfaisant, nous recommençons l'apprentissage avec un nouveau corpus plus grand.

Voici un exemple d'une phrase à analyser et à apprendre: « Julie ouvre la porte. »

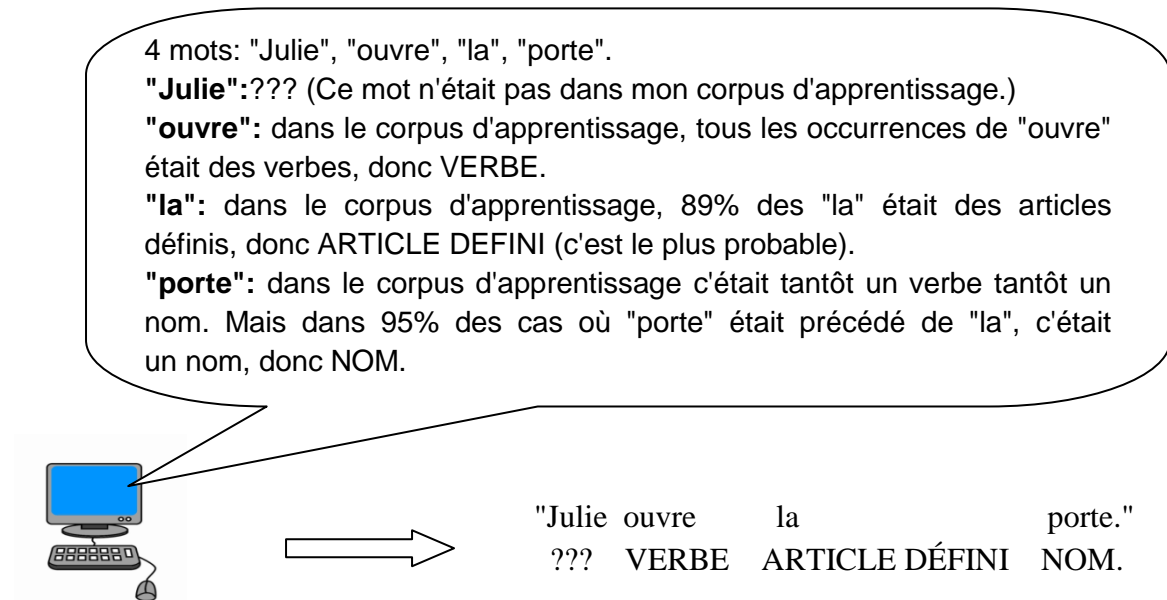


Figure 2.3: Exemple d'analyse d'une phrase avec l'apprentissage

Cet exemple nous montre comment l'ordinateur analyse chaque mot contenant dans la phrase pour nous donner les natures de chacun de ces mots. Nous nous sommes inspirés de la référence [B5] pour cet exemple qui met en relief l'apprentissage et le traitement automatique de la langue. Nous passons dans la partie suivante à introduire les concepts de base des réseaux de neurones que nous utilisons dans notre projet.

2.4 Réseaux de neurones

2.4.1 Introduction

2.4.1.1 Origines

Étant donné que les chercheurs et les informaticiens ont trouvé des difficultés dans le développement des logiciels, ils ont recours à une approche qui est le traitement automatique de l'information et qui consiste à s'inspirer du traitement de l'information effectué par le cerveau. L'hypothèse principale, à la base des réseaux de neurones artificiels, est que le comportement "intelligent" est engendré par un ensemble de mécanismes mentaux. L'organisation en neurones permet d'illustrer les notions d'apprentissage et de mémorisation [B6].

Dans ce qui suit, nous verrons comment passer des modèles biologiques à des modèles mathématiques: les réseaux de neurones artificiels RNA et quelques topologies d'organisation en réseaux. Nous découvrons aussi les concepts d'apprentissage, le Perceptron qui est

historiquement le premier modèle de neurone, et une version plus complexe, évolutive et efficace du Perceptron: le perceptron multicouche. Nous étudierons aussi leur algorithme d'apprentissage: la règle de rétro-propagation du gradient et enfin nous citons quelques exemples d'application des RN.

2.4.1.2 Définition

Nous nous basons sur cette définition adoptée par ces deux références [B6] et [B7].

" Les réseaux de neurones artificiels sont des réseaux fortement connectés de processeurs (neurones) élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Toute structure hiérarchique de réseaux est évidemment un réseau."

2.4.1.3 Neurone biologique

Comme la figure 2.4 le montre, le neurone biologique comprend:

- ★ Le corps cellulaire, qui fait la somme des influx qui lui parviennent; et si cette somme dépasse un certain seuil, il envoie lui-même un flux par l'intermédiaire de l'axone,
- ★ L'axone, qui permet de transmettre les signaux émis par le corps cellulaire aux autres neurones,
- ★ Les dendrites, qui sont les récepteurs principaux du neurone, captant les signaux qui lui parviennent,
- ★ Les synapses, qui permettent aux neurones de communiquer avec les autres via les axones et les dendrites [B8].

Le cerveau humain est composé d'environ 10^{12} neurones (mille milliards), avec 1000 à 10000 synapses (connexions) par neurone [B6].

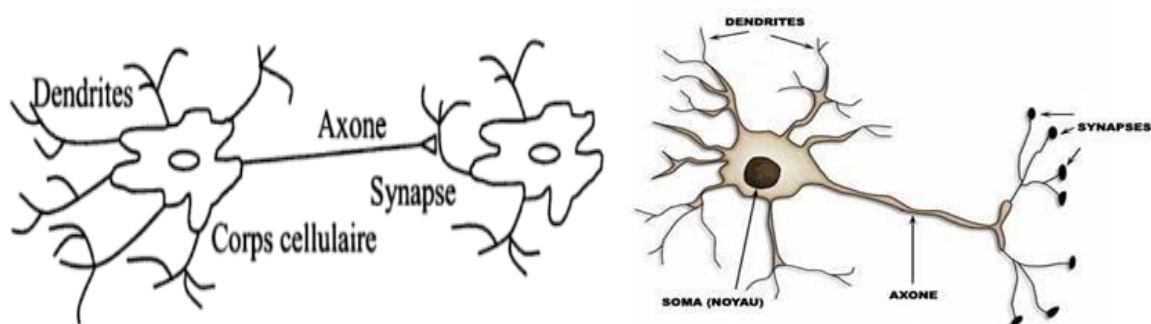


Figure 2.4: Neurones biologiques

2.4.1.4 Neurone artificiel

Le neurone artificiel est l'élément de base d'un réseau de neurones. Il est un processeur élémentaire. Il reçoit un nombre variable d'entrées en provenance de neurones amont. À chacune des entrées est associé un poids w (Wight) représentatif de la force de la connexion. Chaque processeur élémentaire comprend une sortie unique, qui se ramifie ensuite pour alimenter un nombre variable de neurones avals (figure 2.5).

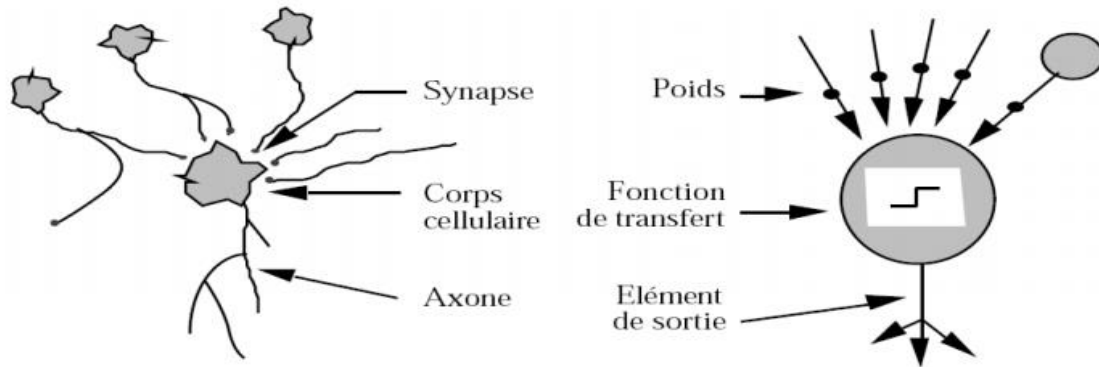


Figure 2.5: Mise en correspondance entre neurone biologique et neurone artificiel

Le comportement d'un neurone artificiel: Nous distinguons deux phases. La première est généralement la somme pondérée des entrées. À partir de cette valeur, une fonction de transfert calcule la valeur de l'état du neurone. C'est cette valeur qui est transmise aux neurones avals [B6]. La figure 2.6 montre un neurone artificiel et les notations utilisées.

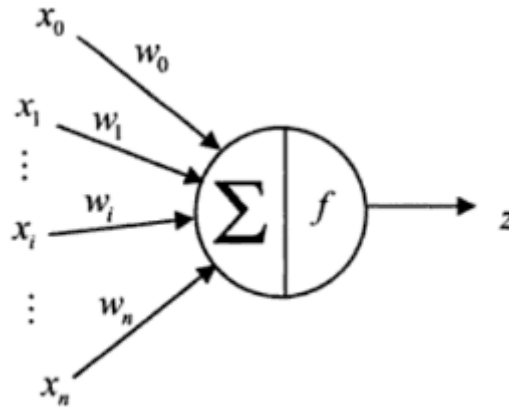


Figure 2.6: Neurone artificiel et les notations utilisées

Le neurone artificiel réalise une fonction $f(y)$ d'une sommation pondérée y des $n + 1$ signaux x_0, x_1, \dots, x_n qui lui parviennent. Les coefficients de pondération w_i , $i = 0, 1, \dots, n$ s'appellent les poids synaptiques. Si w_i est positif, l'entrée x_i est excitatrice alors que si w_i est négatif, elle est inhibitrice.

$$y = \sum_{i=0}^n w_i x_i \text{ et } z = f(y)$$

Le seuil de déclenchement est en général provoqué par une entrée inhibitrice x_0 , parfois appelée biais [B8].

Il existe de nombreuses formes possibles pour la fonction d'activation (la fonction de transfert). La plupart sont des fonctions continues dans l'intervalle $[0,1]$ ou $[-1,1]$ ([B6]).

Les fonctions d'activation les plus utilisées sont présentées dans la figure 2.7.

- a) tout ou rien ;
- b) fonction signe ;
- c) plus ou moins à seuil ;
- d) fonction affine ;
- e) saturation ;
- f) sigmoïde ;
- g) fonction arc tangente ;
- h) fonction radiale de base du type gaussien.

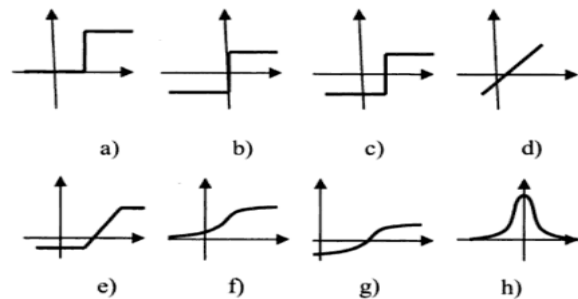


Figure 2.7: Fonctions d'activation les plus utilisées

2.4.2 Architectures des réseaux de neurones artificiels

Un réseau de neurones est un graphe. Les connexions entre les neurones qui composent le réseau décrivent la topologie du modèle. Plusieurs architectures des réseaux existent :

❖ Réseau multicouche

Les neurones sont arrangés par couche. Il n'y a pas de connexion entre neurones d'une même couche et les connexions ne se font qu'avec les neurones des couches avales. Généralement chaque neurone d'une couche est connecté à tous les neurones de la couche suivante et celle-ci seulement. Ceci nous permet d'introduire la notion de sens de parcours de l'information et donc de définir les concepts de neurone d'entrée, neurone caché et neurone de sortie. Par extension, nous appelons couche d'entrée l'ensemble des neurones d'entrée, couche de sortie l'ensemble des neurones de sortie.

Les couches intermédiaires n'ayant aucun contact avec l'extérieur sont appelées couches cachées.

A la lecture du graphique à coté, nous distinguons donc trois nœuds d'entrée, quatre nœuds dans la couche cachée et deux nœuds de sortie. Chaque nœud d'entrée est associé à une variable.

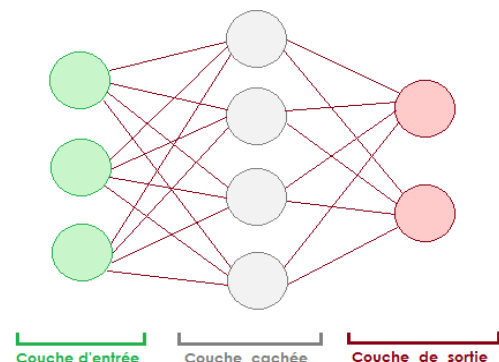


Figure 2.8: Réseau multicouche

❖ Réseau à connexions locales

Il s'agit d'un réseau multicouche où chaque neurone a des relations avec un nombre réduit et localisé de neurones de la couche avale.

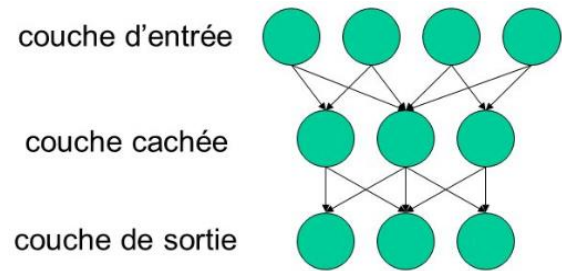


Figure 2.9: Réseau à connexions locales

❖ Réseau à connexions récurrentes

Les connexions récurrentes ramènent l'information en arrière par rapport au sens de propagation défini dans un réseau multicouche. Ces connexions sont le plus souvent locales.

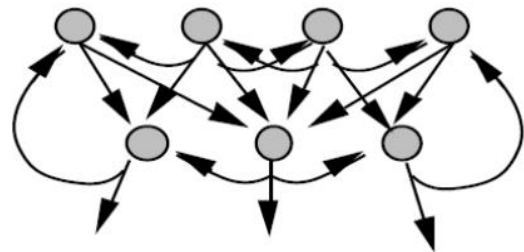


Figure 2.10: Réseau à connexions récurrentes

❖ Réseau à connexions complètes

Chaque neurone est connecté à tous les neurones du réseau [B6].

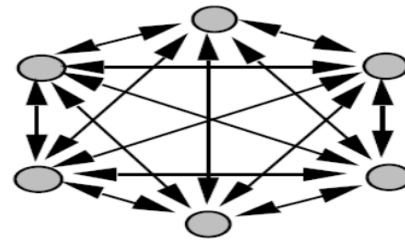


Figure 2.11: Réseau à connexions complètes

2.4.3 Apprentissage

L'apprentissage est la propriété la plus intéressante des réseaux neuronaux.

2.4.3.1 Définition

La définition suivante est extraite des deux références [B6] et [B7] :

" L'apprentissage est une phase de développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré. L'apprentissage neuronal fait appel à des exemples de comportement."

Le réseau de neurones est utilisé pour réaliser une fonction particulière, par exemple dans le problème de reconnaissance de formes, il s'agit d'une classification. Cette fonction va être élaborée lors d'une phase d'apprentissage. Le résultat de cette fonction est obtenu lors d'une phase d'utilisation (ou propagation) du réseau.

La propagation à travers le réseau s'effectue par modification de l'état de neurones, de la première couche cachée jusqu'à la sortie du réseau.

Dans un réseau de neurones, l'information est codée par les poids liés aux connexions.

L'apprentissage est réalisé par des algorithmes de calcul dont le but est d'adapter ces poids en fonction des stimuli présentés à l'entrée du réseau. Une fois l'apprentissage fini, les poids ne sont plus modifiés [B9].

2.4.3.2 Étapes d'apprentissage

L'apprentissage d'un réseau de neurones artificiels passe par les étapes suivantes:

- Avant le traitement des données, nous devons effectuer le choix des objets, la définition des attributs caractérisant les objets et la construction de la base d'apprentissage.
- Acquisition des données formant la base d'apprentissage : Le choix des données initiales est très délicat car il conditionne en grande partie l'efficacité des performances du réseau de neurones. L'acquisition des données permet de convertir les données de leurs formes physiques (son, image, etc.) en une forme acceptable par l'ordinateur.
- Prétraitement : Il consiste à localiser, segmenter et normaliser les représentations. Par exemple pour une image (en occurrence l'image d'un caractère), nous chercherons à supprimer le bruit et à augmenter le contraste.
- Choix des attributs : Après le prétraitement, on doit extraire des attributs qui définissent les données. Ces attributs servent comme des entrées au réseau de neurones. Le choix d'un vecteur d'attributs pour caractériser un objet peut se révéler crucial. En effet, il faut faire un compromis entre la dimension du vecteur et le contenu des informations. Un vecteur de taille très petite conduirait à de mauvaises performances du réseau de neurones.

À la fin de cette phase, nous obtenons un tableau de nombres à deux entrées : les données et les attributs les caractérisant (tableau 2.1).

		Les données					
		X_1	X_2	...	X_d	...	X_D
Les attributs	1	X_{11}	X_{12}	...	X_{1d}	...	X_{1D}
	2	X_{21}	X_{22}	...	X_{2d}	...	X_{2D}

	n	X_{n1}	X_{n2}	...	X_{nd}	...	X_{nD}

	N	X_{N1}	X_{N2}	...	X_{Nd}	...	X_{ND}

Tableau 2.1: Représentation d'un tableau à deux entrées: les attributs et les données

À la fin de ces étapes, nous obtenons une base de connaissances qui sert comme une base d'apprentissage.

L'apprentissage consiste à présenter séquentiellement les exemples et à modifier les poids synaptiques selon une équation dite équation d'apprentissage.

➔ Décision : Pour que le système soit performant, il faut qu'il se comporte bien sur une base de données autre que sa base d'apprentissage, cette base s'appelle la base de généralisation.

Comme nous l'avons précédemment dit, l'apprentissage est l'étape essentielle dans un réseau de neurones. Lors de l'apprentissage, nous cherchons à minimiser une erreur, sur la base d'apprentissage, entre la sortie fournie et la sortie désirée. Nous espérons ainsi minimiser une erreur sur un ensemble de généralisation, c'est à dire un ensemble de données inconnues du réseau.

Nous pouvons arrêter l'apprentissage :

- quand l'erreur entre la sortie du réseau et la sortie désirée ne diminue plus ou diminue d'une façon non significative,
- après un certain nombre d'itérations, fixé à priori,
- pour un problème de classification lorsque le réseau classe correctement toutes les données.

Le problème de ces méthodes est le risque de sur-apprentissage ou d'apprentissage "par cœur". Malheureusement, les données sont souvent entachées de bruit, un apprentissage par cœur ne permet pas une bonne généralisation. Il est donc préférable de tester les capacités du réseau pendant la phase d'apprentissage et d'interrompre celle-ci lorsque l'erreur en généralisation se dégrade [B10].

2.4.3.3 Types d'apprentissage

Comme nous avons déjà énuméré dans la partie d'apprentissage automatique, les types d'apprentissage qui peuvent être classés en trois catégories :

- Apprentissage supervisé : Dans ce type d'apprentissage, nous cherchons à imposer au réseau un fonctionnement donné en forçant à partir des entrées qui lui sont présentées, les sorties du réseau à prendre des valeurs données en modifiant les poids synaptiques. Le réseau se comporte alors comme un filtre dont les paramètres de transfert sont ajustés à partir des couples entrée/sortie présentés.
- Apprentissage non supervisé : Dans ce cas, des exemples ou "prototypes" ou "patrons" sont présentés au réseau que nous laissons s'auto-organiser et se stabiliser

au moyen de lois locales qui régissent l'évolution des poids synaptiques. Ce mode d'apprentissage est aussi appelé "apprentissage par compétition" [B8].

- Apprentissage par renforcement : Il correspond au cas où l'algorithme apprend un comportement étant donné une observation. L'action de l'algorithme sur l'environnement produit une valeur de retour qui guide l'algorithme d'apprentissage.

Comme nous l'avons préalablement souligné, nous avons utilisé l'apprentissage supervisé. Par la suite, nous allons étudier le Perceptron multicouche.

2.4.4 Perceptron Multicouche

Il y a de très nombreuses sortes de réseaux de neurones actuellement. Cependant nous intéressons au Perceptron multicouche car notre réseau de neurones sera de ce type.

2.4.4.1 Perceptron simple

Afin de définir la structure collective d'un ensemble de neurones, il est important de définir le perceptron monocouche (SLP en anglais pour dire Simple Layer Perceptron). C'est le premier modèle opérationnel de réseau de neurones, conçu en 1958 par Rosenblatt. Il est inspiré du système visuel. Extrêmement simple, ce modèle ne comportait alors qu'un seul neurone. Il permet de classifier correctement des objets appartenant à deux classes linéairement séparables. Il consiste en un seul neurone qui possède un seuil ainsi qu'un vecteur de poids synaptiques ajustables. La structure ainsi que les divers composants d'un Perceptron simple sont illustrés dans la figure 2.12 [B11].

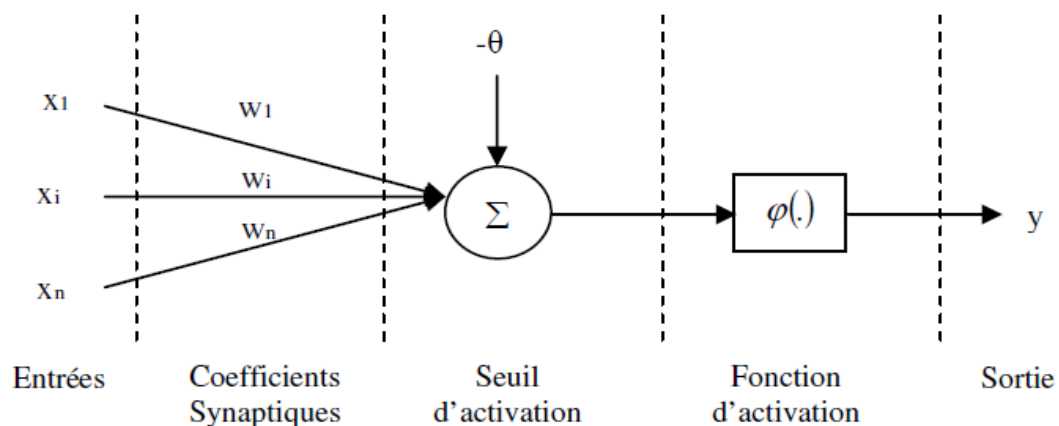


Figure 2.12: Le Perceptron simple

Les limites du perceptron monocouche ont été montrées. Pour cela, nous avons eu recours à un réseau plus complexe et qui s'appelle le Perceptron multicouche que nous allons dévoiler à ce niveau.

2.4.4.2 Structure du Perceptron Multicouche

Les Perceptrons Multicouches (PMC), en anglais Multi Layer Perceptrons (MLP), sont les réseaux de neurones les plus courants et les plus simples. Ils sont très largement utilisés en classification, en reconnaissance de formes, traitement des images, et en problèmes d'approximation et de prédictions, notamment pour leurs bonnes performances et leur simplicité.

Le PMC est une extension du précédent perceptron, avec une ou plusieurs couches cachées entre l'entrée et la sortie, donc un PMC possède trois types de couches: une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie. Chaque neurone d'une couche est connecté à tous les neurones de la couche qui le précède, ce qui donne un réseau complètement connecté. La figure 2.13 représente un PMC à trois couches. La couche d'entrée comporte deux neurones. La couche cachée contient cinq neurones et enfin la couche de sortie possède un seul neurone [B11].

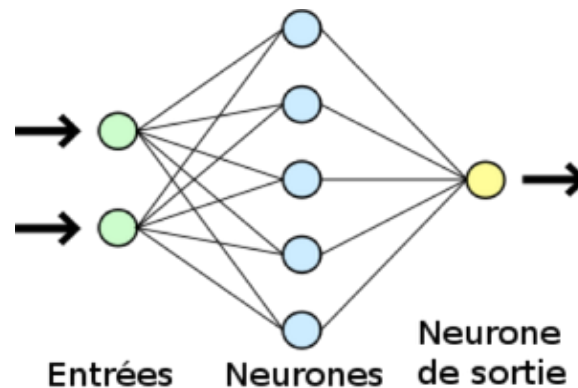


Figure 2.13: Architecture d'un Perceptron Multicouche à une seule couche cachée

Le choix du nombre de couches cachées dépend généralement de la complexité du problème à résoudre, en théorie une seule couche cachée peut être suffisante pour résoudre un problème donné mais il se peut que le fait de disposer de plusieurs couches cachées permette de résoudre plus facilement un problème complexe [B12].

Le choix de la fonction de transfert pour les couches cachées réside dans sa dérivabilité et sa simplicité de calcul pour l'apprentissage comme par exemple la fonction sigmoïde ou la fonction tangente hyperbolique [B11].

Ce type de réseau (le PMC) se situe dans la famille générale des réseaux à propagation vers l'avant, c.à.d. qu'en mode normal d'utilisation, l'information se propage dans un sens unique, des entrées vers les sorties sans aucune rétroaction. Son apprentissage est de type supervisé, par correction des erreurs. Dans ce cas uniquement, le signal d'erreur est rétro propagé vers

les entrées pour mettre à jour les poids des neurones. Donc, cela nous mène à introduire l'algorithme d'apprentissage du perceptron multicouche qui consiste en algorithme de rétro-propagation du gradient.

2.4.4.3 Algorithme d'apprentissage de PMC : Règle de rétro-propagation du gradient

La règle du gradient de l'erreur (delta rule) est l'une des règles les plus utilisées pour l'apprentissage de réseaux de neurones. Cette règle, initialement développée pour résoudre des problèmes de traitements adaptatifs du signal, a ensuite été exploitée pour obtenir le très populaire algorithme de rétro propagation du gradient de l'erreur (retro propagation) pour réseaux de neurones multicouches [B9].

Dans un cadre général, l'apprentissage consiste en un entraînement du réseau. Nous présentons au réseau des entrées et nous lui demandons de modifier sa pondération de telle sorte que nous retrouvons la sortie correspondante de ce réseau. L'algorithme consiste dans un premier temps à propager vers l'avant les entrées jusqu'à obtenir une sortie calculée par le réseau. La seconde étape est de comparer la sortie calculée à la sortie réelle connue ou dite la sortie désirée ([B12] et [16]). L'objectif est de minimiser une fonction de coût "E". L'équation suivante exprime cette fonction de coût, à partir de l'erreur quadratique, pour un couple entrée-sortie, avec d_k la sortie désirée pour le neurone d'indice k et s_k la sortie obtenue par le réseau : $E = \frac{1}{2} \sum_i (d_k - s_k)^2$ [B9]. Nous rappelons que la sortie désirée joue en quelque sorte le rôle du professeur et sert à superviser l'entraînement [B11].

Les poids sont donc modifiés de telle sorte qu'à la prochaine itération, l'erreur commise entre la sortie calculée et la sortie connue soit minimisée. Malgré tout, il ne faut pas oublier que nous avons des couches cachées. L'erreur est rétro propagée vers l'arrière jusqu'à la couche d'entrée tout en modifiant la pondération. Le processus est répété sur tous les exemples jusqu'au temps où nous obtenions une erreur de sortie considérée comme négligeable ([B12] et [16]).

Quand nous cherchons à minimiser l'erreur en trouvant le minimum de la fonction du coût, le piège est de tomber dans des minimas locaux. L'astuce est d'utiliser deux paramètres différents :

- ➔ Le taux d'apprentissage (Learning rate) : est un paramètre qui contrôle la vitesse de descente (la pente) à laquelle les pondérations sont ajustées ([17] et [18]). Plus nous descendons vite, plus il est probable que nous loupons le minimum [17]. Donc, un taux d'apprentissage plus grand, augmente la vitesse d'apprentissage mais diminue la

précision et un taux d'apprentissage plus petit prend plus de temps mais augmente la précision du réseau de neurones [19].

- Le moment ou dit aussi le coefficient de hasard (momentum) : est un terme d'inertie que l'on ajoute à la formule de rétro propagation pour aider l'algorithme du gradient à sortir des minimas locaux ([17] et [20]).

Pour conclure la partie des réseaux de neurones, nous allons citer quelques exemples généraux de leurs applications.

2.4.5 Exemples généraux d'application des RN

Se trouvant à l'intersection de différents domaines (informatique, électronique, science cognitive, neurobiologie et même philosophie), l'étude des réseaux de neurones est une voie prometteuse de l'intelligence artificielle, qui a des applications dans de nombreux domaines et secteurs très variés ([B8] et [B13]):

- ★ Industrie : contrôle qualité, diagnostic des pannes, corrélations entre les données fournies par différents capteurs, commandes de processus, robotique, simulation boîte noire, analyse de signature ou d'écriture manuscrite, tournées de véhicules, régulation de trafic.
- ★ Finance : prévision et modélisation du marché (cours de monnaie,...), sélection d'investissements, attribution de crédit.
- ★ Télécommunication et informatique : analyse du signal, élimination du bruit, reconnaissances de formes (bruits, images, paroles), compression de données, cryptographie, classification.
- ★ Environnement : analyse chimique, évaluation des risques, prévisions et modélisation météorologiques, gestion et allocation des ressources.

Conclusion

Dans ce chapitre, nous avons essayé de présenter les notions de base que nous avons dues savoir pour mieux comprendre le travail demandé. En premier lieu, nous avons défini l'apprentissage automatique, ses domaines d'application et ses types. En second lieu, nous avons introduit la linguistique informatique et le traitement automatique de langue. En troisième lieu, nous avons présenté quelques applications existantes reliant l'apprentissage et le TAL. Et, en quatrième lieu, nous nous sommes intéressés par les concepts de base des réseaux de neurones. Dans le chapitre suivant, nous allons décrire la solution que nous proposons pour la problématique posée.

Chapitre 3

Approche et solution proposées

Introduction

Dans ce chapitre, nous commençons par l'analyse et spécification des besoins auxquels doit répondre notre plateforme. Ensuite, nous procédons à la conception globale et détaillée de notre application. Puis, nous décrivons la procédure de développement d'un réseau de neurones. Une modélisation s'avère indispensable à ce stade pour pouvoir présenter les objectifs réels de notre application afin d'arriver à lister ses différentes exigences. Cette modélisation peut être formalisée par des diagrammes de cas d'utilisation pour la partie d'analyse et spécification des besoins et des diagrammes de séquences et classes pour la partie conception. La méthodologie UML est alors utilisée pour cette tâche.

I. Analyse et spécification des besoins

I.1 Spécification des besoins

La spécification des besoins représente une phase importante dans un cycle de développement du projet, c'est au cours de laquelle que les besoins d'utilisateur sont identifiés et précisés.

I.1.1 Spécification des besoins fonctionnels

Un besoin fonctionnel est un besoin spécifiant une action qu'un système doit être capable d'effectuer. L'application doit obligatoirement répondre à l'ensemble des besoins fonctionnels suivants :

- ✓ Permettre à l'utilisateur de saisir une phrase, ses mots et leurs propriétés,
- ✓ Mettre à jour des phrases et/ou leurs mots et/ou leurs propriétés soit en les ajoutant, les modifiant ou les supprimant,
- ✓ Afficher des statistiques des mots d'une phrase saisie sous forme de graphes,
- ✓ Afficher les phrases du chat relié à l'application, les statistiques des mots qui composent ces phrases et des graphes correspondant à ces statistiques,
- ✓ Générer automatiquement des phrases selon un dictionnaire bien précis en affichant quelques statistiques permettant d'alimenter la BD et tester le réseau de neurones.

I.1.2 Spécification des besoins non fonctionnels

Les besoins non fonctionnels décrivent toutes les contraintes auxquelles est soumis le système pour son développement et son fonctionnement. Notre solution doit répondre aux critères suivants :

- ✓ **Ergonomie des interfaces et Facilité d'utilisation** : Fournir une interface simple et élégante pour l'utilisateur afin de garantir une facilité d'exploitation et une rapidité de service.
- ✓ **Portabilité** : L'application doit fonctionner sur n'importe quel environnement d'exécution.
- ✓ **Maintenabilité** : Dans le développement des différents modules de cette plateforme, il faut tenir compte de la possibilité de son extension par l'ajout de nouvelles fonctionnalités. Ces modules doivent être aussi bien documentés afin de faciliter la maintenabilité de l'application.
- ✓ **Rapidité** : Le système doit être rapide car il est connecté à un chat; il doit récupérer les phrases en temps réel.

I.2 Analyse des besoins

I.2.1 Identification des acteurs

Un acteur est une personne ou un composant (système ou interface) externe au système qui agit comme un déclencheur ou un intermédiaire pour réaliser une action avec le système. En réponse à l'action d'un acteur, le système fournit un service qui correspond à son besoin.

Comme notre projet s'intègre dans le cadre de développement d'un assistant virtuel, nous n'avons qu'un seul acteur qui est un utilisateur qualifié qui a des connaissances en apprentissage automatique et en réseaux neuronaux en informatique.

I.2.2 Langage de modélisation

Pour la modélisation objet, nous avons choisi le langage commun UML " Unified Modeling Language " [B1]. En effet, UML est un langage de modélisation formel et normalisé, né de la fusion de plusieurs méthodes existantes. Il permet de modéliser informatiquement un ensemble d'une partie du monde réel en un ensemble d'entités informatiques. Ces entités informatiques sont appelées objets. Ces objets sont décrits par des vues statiques et dynamiques, incluant un ensemble de diagrammes, qui collaborent pour représenter diverses projections d'une même représentation d'un système d'objets.

I.2.3 Diagrammes des cas d'utilisation

Le diagramme de cas d'utilisation est un diagramme UML utilisé pour donner une vision globale du comportement fonctionnel d'un système logiciel. Ce diagramme décrit les différentes actions qui peuvent être effectuées par un acteur dans un système.

I.2.3.1 Diagramme des cas d'utilisation général

Afin de présenter les principales fonctionnalités de notre application, le diagramme de cas d'utilisation général du système entier est illustré par la figure 3.1 suivante.

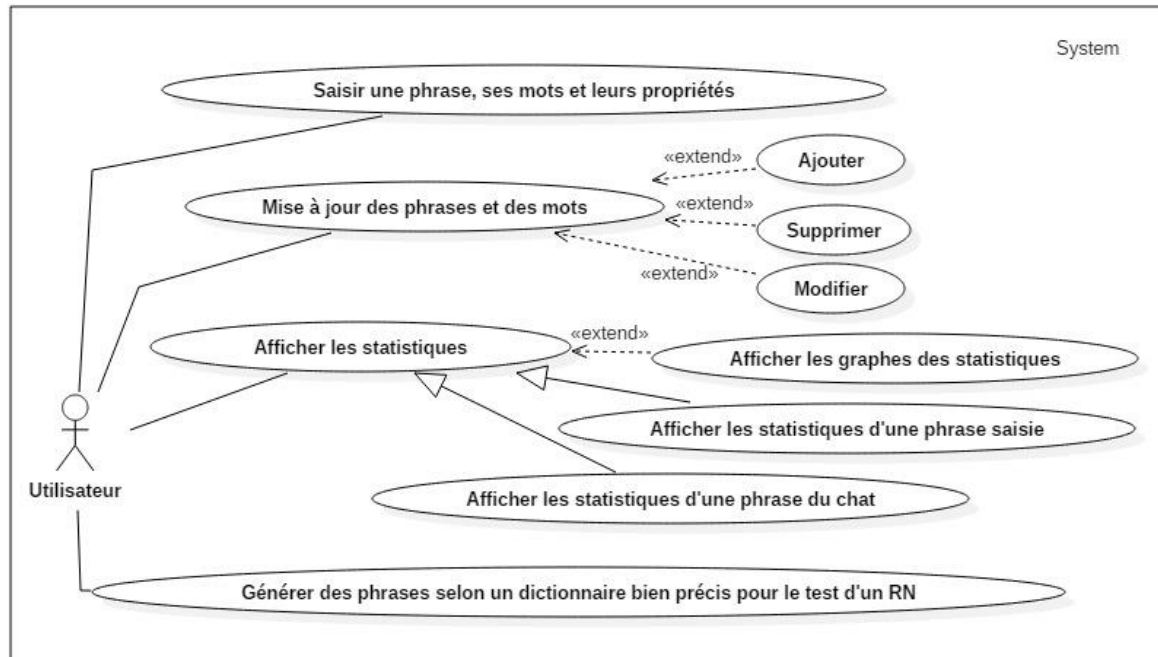


Figure 3.1: Diagramme de cas d'utilisation du système

I.2.3.2 Description détaillée des cas d'utilisation

Nous allons maintenant détailler les différents cas d'utilisation en élaborant les scénarios nominaux et alternatifs associés.

- Saisir une phrase, ses mots et leurs propriétés

L'utilisateur saisit une phrase et les mots qui composent cette phrase et leurs propriétés : le pourcentage d'importance du mot, la fonction du mot dans la phrase et la position du mot. Le tableau 3.1 décrit une description textuelle de ce cas d'utilisation.

Titre	Saisir une phrase, ses mots et leurs propriétés
But	Saisir une phrase, ses mots et leurs propriétés
Acteur	Utilisateur
Pré conditions	Pas de pré conditions
Définition des enchaînements	Scénario nominal <ol style="list-style-type: none"> 1. L'utilisateur saisit une phrase. 2. Le système lui affiche le nombre de mots qui composent la phrase. 3. L'utilisateur ajoute la phrase et son nombre de mots à la BD en cliquant sur le bouton correspondant.

	<ol style="list-style-type: none"> Le système lui affiche dans une table la phrase saisie et le nombre de ses mots qui sont déjà ajoutés à la BD. En plus, il lui affiche les différents mots. L'utilisateur sélectionne à chaque fois le mot et fait entrer ses propriétés : son pourcentage d'importance qui est un nombre aléatoire et qui varie dans l'intervalle] 0,100[(à condition que la somme des pourcentages de tous les mots soit égale à 100 %), sa fonction (il choisit à partir d'une liste déroulante la fonction correspondante) et sa position dans la phrase (il suffit de cliquer dans le champ et le système le lui affiche) et les ajoute à la BD. Le système ajoute le mot et ses propriétés à la BD et il affiche dans un tableau le mot et ses propriétés. <p>Scénario alternatif</p> <ul style="list-style-type: none"> - Quand le système affiche le nombre de mots, il questionne l'utilisateur s'il est correct ou non et il lui donne la main pour le corriger s'il est incorrect. - Si la somme des pourcentages d'importance de tous les mots n'est pas égale à 100%, le système lui affiche une erreur et lui donne la possibilité de corriger. - Si la fonction du mot n'existe pas dans la liste, le système donne à l'utilisateur la possibilité d'ajouter une nouvelle fonction. - Si la position donnée par le système est incorrecte, il est possible aussi de la corriger.
Post conditions	La phrase saisie, le nombre de ses mots, ses différents mots et leurs propriétés sont ajoutés à la BD et sont affichés à l'utilisateur.

Tableau 3.1: Tableau descriptif du cas d'utilisation "Saisir une phrase, ses mots et leurs propriétés"

- Mise à jour d'une phrase et/ou ses mots et/ou leurs propriétés

L'utilisateur peut mettre à jour la phrase saisie et/ou les mots qui composent cette phrase et/ou leurs propriétés soit en modifiant les valeurs soit en les supprimant. Le tableau 3.2 décrit une description textuelle de ce cas d'utilisation.

Titre	Mise à jour d'une phrase et/ou de ses mots et/ou leurs propriétés
But	Modifier et/ou supprimer des phrases et/ou leurs mots et/ou leurs propriétés
Acteur	Utilisateur
Pré conditions	La phrase, ses mots et leurs propriétés sont ajoutés à la BD.
Définition des enchaînements	<p>Scénario nominal</p> <ol style="list-style-type: none"> Pour modifier une phrase, l'utilisateur la sélectionne du tableau puis la modifie tout en modifiant son nombre de mots si nécessaire. De même pour les mots, à condition que la somme des pourcentages d'importance de tous les mots d'une phrase soit égale à 100%. Pour supprimer une phrase ou un mot, il sélectionne l'objet à supprimer. L'utilisateur clique sur le bouton "Modifier" ou "Supprimer" selon le cas. Le système lui exécute la tâche de modification ou de suppression.

	Scénario alternatif - Si la somme des pourcentages d'importance des mots n'est pas égale à 100, le système avertit l'utilisateur et lui permet de corriger surtout dans le cas de modification.
Post conditions	Ce qui à modifier a été modifié et ce qui à supprimer a été supprimé.

Tableau 3.2: Tableau descriptif du cas d'utilisation "Mise à jour d'une phrase et/ou ses mots et/ou leurs propriétés"

- Afficher les statistiques

L'utilisateur peut afficher les statistiques des mots d'une phrase saisie. Le tableau 3.3 décrit une description textuelle du cas d'utilisation "Afficher les statistiques".

Titre	Afficher les statistiques
But	Afficher les statistiques
Acteur	Utilisateur
Pré conditions	Pas de pré conditions
Définition des enchaînements	Scénario nominal <ol style="list-style-type: none"> 1. L'utilisateur saisit une phrase et clique sur le bouton "Afficher statistiques". 2. Le système lui affiche alors dans un tableau les différents mots qui composent cette phrase et les statistiques suivantes pour chaque mot : le nombre d'apparitions du mot dans la BD (le nombre d'occurrences de ce mot dans la BD) et son taux d'apprentissage qui est calculé selon une formule que nous détaillerons plus loin. En plus, le système affiche les mots dans une liste. 3. L'utilisateur peut sélectionner un mot de la liste et par conséquent le système lui affiche dans une liste déroulante les différentes fonctions que prennent ce mot. 4. Le client choisit une fonction et l'application lui affiche le nombre d'apparitions de cette fonction dans la BD. 5. L'utilisateur peut afficher des graphes de statistiques par un simple clic sur les boutons correspondants : <ul style="list-style-type: none"> • S'il clique sur le bouton "Histogramme de nombre d'apparitions des mots", le système lui affiche un histogramme de nombre d'apparitions de chaque mot composant la phrase. • S'il clique sur le bouton "Histogramme de taux d'apprentissage des mots", la plateforme lui affiche un histogramme de taux d'apprentissage de chaque mot. • S'il clique sur le bouton "Camembert de nombre d'apparitions des fonctions", le système affiche un graphe de camembert où il y a les différents nombres d'apparitions de ces fonctions. Scénario alternatif - Si un mot est nouveau (c.à.d. qui n'existe pas dans la BD), connu par le fait

	<p>que le nombre d'apparitions de ce mot est égal à zéro, et donc il n'a pas de taux d'apprentissage, alors l'utilisateur peut cliquer sur ce mot dans la liste et par la suite le système lui ouvre un panneau pour ajouter ce mot et ses propriétés. Le client remplit les champs et clique sur le bouton "afficher les statistiques" et ainsi les valeurs du nombre d'apparitions et de taux d'apprentissage se modifient car ce mot est ajouté à la BD.</p> <p>- Il faut que tous les mots existent dans la BD pour afficher le graphe d'histogramme de taux d'apprentissage des mots.</p>
Post conditions	Les statistiques sont affichées.

Tableau 3.3: Tableau descriptif du cas d'utilisation "Afficher les statistiques"

Comme nous l'avons déjà mentionné, notre plateforme permet aussi à l'utilisateur d'afficher les phrases écrites dans la messagerie instantanée (le chat qui est reliée à notre application) et permet de lui afficher les statistiques. Cela se fait de la même manière que le cas d'utilisation "afficher les statistiques". Ce sont les mêmes scénarios mais juste au lieu de saisir une phrase, le client choisit une phrase dite dans le chat pour découvrir ses statistiques. Il peut aussi cliquer sur un bouton pour rafraîchir la liste des phrases écrites dans le chat pour qu'il puisse récupérer les phrases en temps réel.

- Générer des phrases selon un dictionnaire bien précis pour le test d'un RN

Le système peut générer des phrases aléatoires à partir d'un dictionnaire de mots bien précis car nous avons besoin de ces phrases pour le test du réseau de neurones. En outre, le système peut fournir à l'utilisateur quelques statistiques. Nous allons détailler postérieurement cette approche reliée au RN. Le tableau 3.4 décrit une description textuelle du cas d'utilisation "Générer des phrases selon un dictionnaire bien précis pour le test d'un RN".

Titre	Générer des phrases selon un dictionnaire bien précis pour le test d'un RN
But	Générer des phrases pour tester le RN
Acteur	Utilisateur
Pré conditions	Pas de pré conditions
Définition des enchaînements	<p>Scénario nominal</p> <ol style="list-style-type: none"> 1. L'utilisateur clique sur le bouton "Générer automatiquement". 2. Le système lui génère une phrase. 3. Si la phrase est bien structurée et correcte et si l'utilisateur veut afficher ses statistiques, il lui suffit d'appuyer sur le bouton "Affichage et Ajout". 4. Le système lui affiche alors les statistiques : le nombre d'apparitions de chaque mot dans la phrase et son taux d'apprentissage. Il ajoute aussi à la BD deux enregistrements sous une forme bien déterminée que nous préciserons ultérieurement.

Post conditions	Une phrase est générée aléatoirement, ses statistiques sont affichées et il y a ajout à la BD selon une forme précise.
------------------------	--

Tableau 3.4: Tableau descriptif du cas d'utilisation " Générer des phrases selon un dictionnaire bien précis pour le test d'un RN "

Dans cette première section de ce chapitre, nous avons énuméré les besoins fonctionnels et non fonctionnels de la solution proposée. Nous avons fourni une analyse plus détaillée de ces besoins. Ces besoins seront la base sur laquelle nous allons réaliser la conception de cette solution. Cette conception fera l'objet de la deuxième section.

II. Conception

La présente section sera consacrée à la présentation de la phase de conception de notre système. Cette étape est primordiale dans le déroulement du projet et a pour but de détailler les tâches à entreprendre pour préparer le terrain à l'étape de réalisation. Nous présentons d'abord l'architecture générale du système. Nous détaillons ensuite à travers les diagrammes de séquences et de classes cette conception. Finalement, nous décrivons la procédure de développement d'un réseau de neurones.

II.1 Conception générale

II.1.1 Architecture globale de la solution

A partir des interfaces de notre système qui est connecté à un chat, nous collectons les phrases saisies ou extraites du chat. Nous faisons des statistiques sur ces données qui nous servent à construire le réseau de neurones. Notre système possède une base de données pour stocker toutes les informations nécessaires à la création et l'apprentissage du réseau de neurones. La figure 3.2 montre les principaux composants de notre solution :

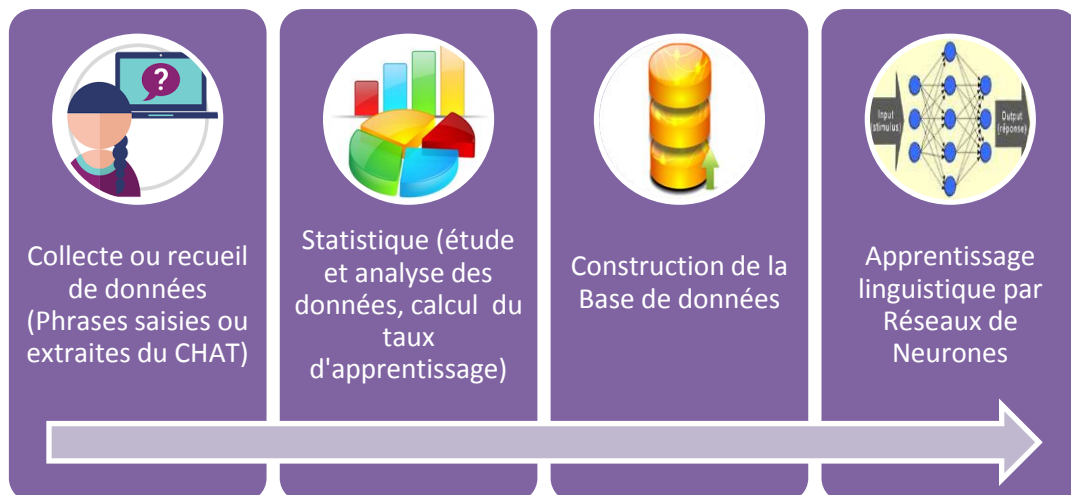


Figure 3.2: Architecture globale du système

II.1.2 Choix de l'approche Orientée Objet

La programmation orientée objet est définie comme un paradigme de programmation informatique basé sur l'interaction de briques appelées objets, qui sont eux-mêmes des abstractions d'objets réels, via leurs relations et les propriétés qui leur sont accordées. Elle a l'avantage de découper les logiciels en pièces modulaires, réutilisables et facilement adaptables, en fonction de l'environnement. Par ailleurs, cette approche offre plusieurs atouts notamment :

- La modularité : Les communications entre objets sont réalisées par le biais d'opérations d'interface.
- La sécurité : Le code de chaque méthode ne s'applique que sur les données de l'objet. Certaines parties de l'objet ne sont pas accessibles pour certains.
- La bonne conception : Les données et les méthodes sont spécifiées en même temps. La spécification et l'implantation peuvent être séparées.
- La lisibilité : Les données et les méthodes sont décrites au même endroit.
- La portabilité : Les parties masquées pourront être modifiées sans que l'utilisateur n'ait à changer son code puisqu'il n'a pas directement accès à celles-ci.
- La Réutilisabilité : Exploiter des composants déjà implémentés.
- L'optimalité : Réduire le code produit.

II.2 Conception détaillée

Après avoir présenté l'architecture globale de notre application, nous exposons dans cette partie la conception détaillée. Nous commençons par décrire la conception de la base de données puis présenter quelques scénarios détaillés de quelques cas d'utilisation.

II.2.1 Conception de la base de données

II.2.1.1 Modèle relationnel de la base de données

La base de données de notre application comporte les tables suivantes :

- Phrase (Id phrase, Phrase),
- Mots (Id mot, Mot, Pourcentage, Fonction, Position, Nombre_de_mots, #numéro_phrase),
- Fonctions (Id fonction, Fonction),
- Ofmessagearchive (messageID, conversationID, fromJID, fromJIDRessource, toJID, toJIDRessource, sentDate, stanza, body),
- Représentation (Id, Représentation, Fonction),
- Dictionnaire (Id mot dic, mot, taux_apprentissage, Fonction),
- Inputs outputs (Id_phrase_test, {tous les mots du dictionnaire}),

- Data outputs (Id_output, {tous les mots du dictionnaire}).

II.2.1.2 Description des tables

Ci-après la description du contenu de chaque table.

- La table Phrase regroupe les phrases saisies par l'utilisateur. Elle contient l'identifiant de la phrase et la phrase elle-même.
- La table Mots regroupe tous les mots que contient la phrase saisie et que l'utilisateur les a ajoutés. Elle contient aussi toutes les informations relatives à ces mots, tels que les pourcentages d'importance dans la phrase, leurs fonctions, leurs positions, le nombre total de mots dans la phrase et le numéro de la phrase dans la table Phrase.
- La table fonctions regroupe tous les fonctions que peuvent prendre un mot.
- La table Ofmessagearchive rassemble tous les phrases écrites dans le chat qui est relié à notre plateforme. C'est lorsque nous avons relié notre BD au serveur du chat à l'aide d'un plugin que plusieurs tables relatifs au chat sont apparues et cette table est la plus importante. Elle est composée de l'identifiant du message, l'identifiant de la conversation, le champ fromJID qui indique le premier membre du chat dans une discussion à deux, le champ fromJIDRessource qui prend toujours le nom du chat "Spark", le champ toJIDRessource qui indique le deuxième membre du chat dans une discussion à deux, le champ sentDate contient la date d'envoi du message mais sous forme d'un grand entier, le champ stanza contient toutes ces informations dans des balises, et le champ body qui est le plus important et qui renferme le contenu du message envoyé lors du chat.
- La table représentation contient des enregistrements sous forme de l'identifiant de cette représentation, la représentation qui signifie les différents émoticônes et smileys qui existent dans le chat et que l'utilisateur peut les taper, et enfin la fonction de ces émoticônes par exemple l'émoticône :) a comme fonction smiley sourire et l'émoticône :D a comme fonction émoticône rire, heureux.
- Ces différents tables sont utilisées dans notre application développée tandis que les tables que nous allons présenter maintenant sont utilisées pour le réseau de neurones et son test puisque l'essentiel de notre projet est de faire un apprentissage linguistique à l'aide des réseaux de neurones.
- La table dictionnaire stocke les mots que contiennent un ensemble de phrases que nous avons choisi sur lesquels nous allons faire l'entraînement de notre réseau de

neurones. Les colonnes de cette table sont : l'identifiant du mot dans le dictionnaire, le taux d'apprentissage du mot et la fonction de ce mot.

- La table InputsOutputs n'est pas une table comme les tables auxquelles nous sommes habituées. Nous avons eu recours à cette table pour faciliter la tâche de test de notre réseau de neurones. Elle est composée de l'identifiant de la phrase que nous allons tester dans le RN et les mots du dictionnaire. Les mots qui composent une phrase saisie ou générée par l'application prennent la valeur de leurs nombres d'apparition dans cette phrase sinon les autres mots du dictionnaire prennent une valeur nulle par défaut. Ce sont les entrées de notre RN.
- Comme la table InputsOutputs, la table DataOutputs sert pour le test du RN. Les attributs sont l'identifiant de l'output c'-à-d l'identifiant de la sortie du RN, et les mots qui composent notre dictionnaire. Comme les entrées, les sorties sont sous forme de vecteurs et nous avons mis ces vecteurs dans des enregistrements. Les mots qui composent la phrase saisie ou générée par l'application cette fois-ci prennent les valeurs de leurs taux d'apprentissage sinon les autres mots du dictionnaire prennent une valeur nulle par défaut.

II.2.1.3 Diagramme de classes

Le diagramme de classes constitue un élément très important de la modélisation : il permet de définir quelles seront les composantes du système final et d'en déduire la base de données.

Néanmoins, nous constatons souvent qu'un diagramme de classes proprement réalisé permet de structurer le travail de développement de manière très efficace. Il permet aussi dans le cas de travaux réalisés en groupe, de séparer les composantes de manière à pouvoir répartir le travail de développement entre les membres du groupe. Enfin, il permet de construire le système de manière correcte. Le diagramme de classes de la figure 3.3 fournit une description bien détaillée des différentes classes de l'application ainsi que les différentes associations qui les relient :

- La classe Phrase représente les phrases tapées par l'utilisateur.
- La classe Mots représente les différentes informations des mots qui composent les phrases.
- La classe Fonction illustre les différentes fonctions que peuvent prendre les mots.
- La classe OfMessageArchive regroupe les messages et donc les phrases écrites dans le chat.
- La classe Représentation décrit les représentations qui correspondent aux émoticônes du chat et leurs fonctions.

- La classe Dictionnaire rassemble les mots qui composent l'ensemble d'entraînement ou d'apprentissage pour le réseau de neurones.
- Les deux classes InputsOutputs et DataOutputs sont utilisées pour le test de RN.

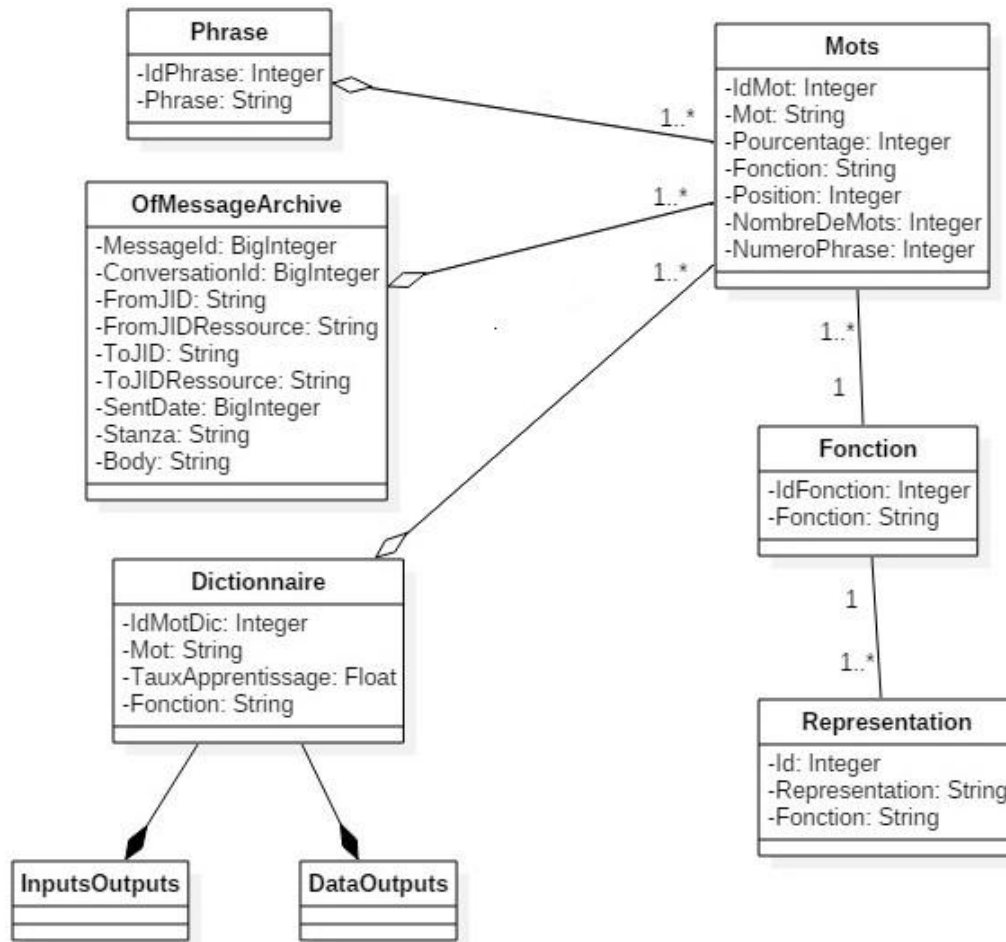


Figure 3.3: Diagramme de classes

II.2.2 Diagrammes de séquences

Les diagrammes de séquences permettent de représenter les vues dynamiques du système. En effet, ils montrent les collaborations entre les objets selon un point de vue temporel en mettant l'accent sur la chronologie des envois des messages. Les diagrammes de séquences relatifs à quelques scénarios de notre application seront représentés dans ce qui suit.

II.2.2.1 Diagramme de séquence de "Saisir une phrase, ses mots et leurs propriétés"

L'utilisateur fait entrer des phrases, leurs mots et leurs propriétés à la base de données pour former l'ensemble ou la base d'apprentissage. La figure 3.4 décrit le scénario de "Saisir une phrase, ses mots et leurs propriétés".

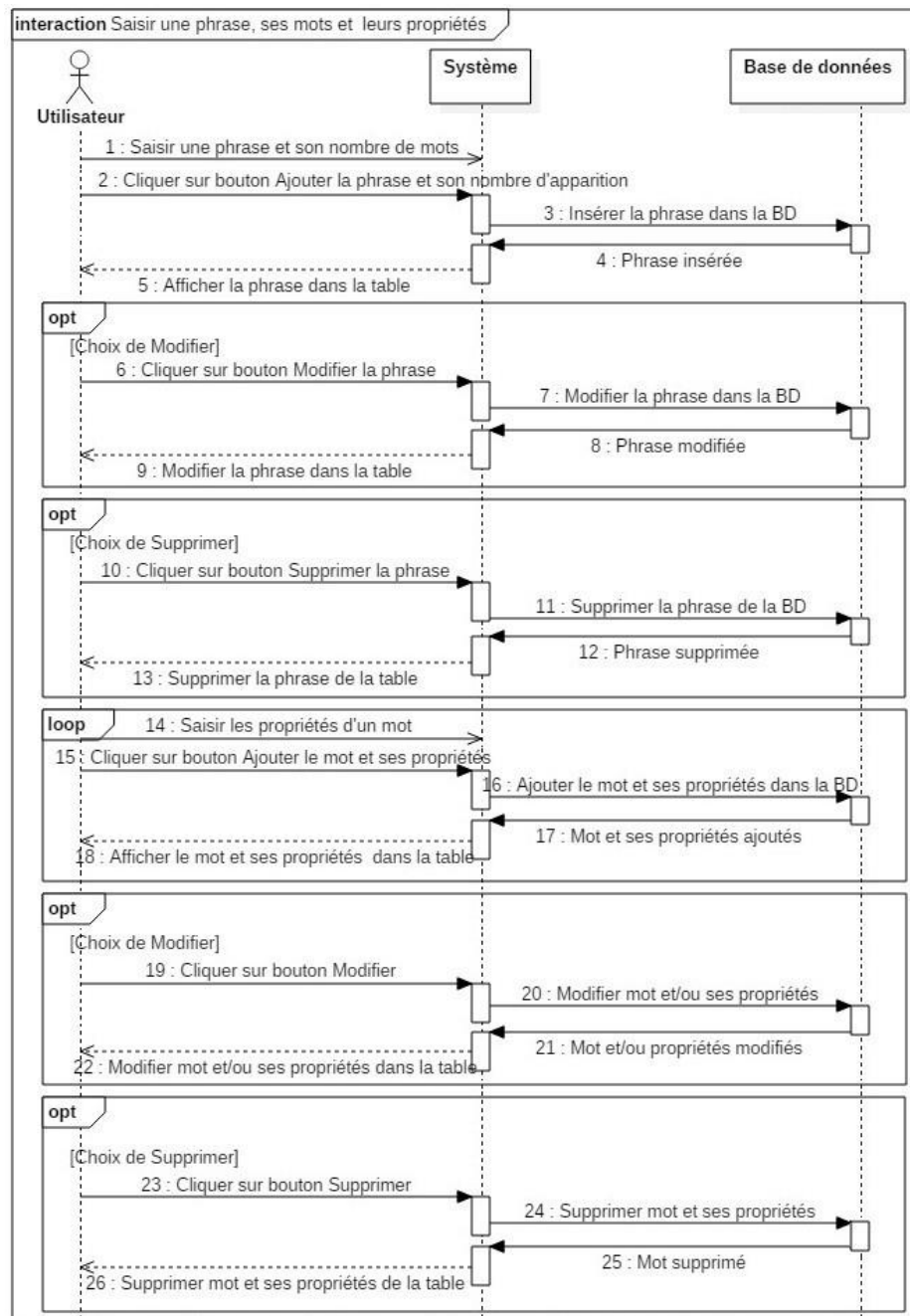


Figure 3.4: Diagramme de séquence de Saisir une phrase, ses mots et leurs propriétés

Comme le montre la figure 3.5, l'utilisateur doit écrire une phrase pour que le système lui affiche les statistiques. L'application permet aussi à l'utilisateur d'ajouter les mots à la BD s'ils n'existent pas déjà. Elle donne la possibilité à l'utilisateur d'afficher des graphes de statistiques comme des histogrammes et des camemberts.

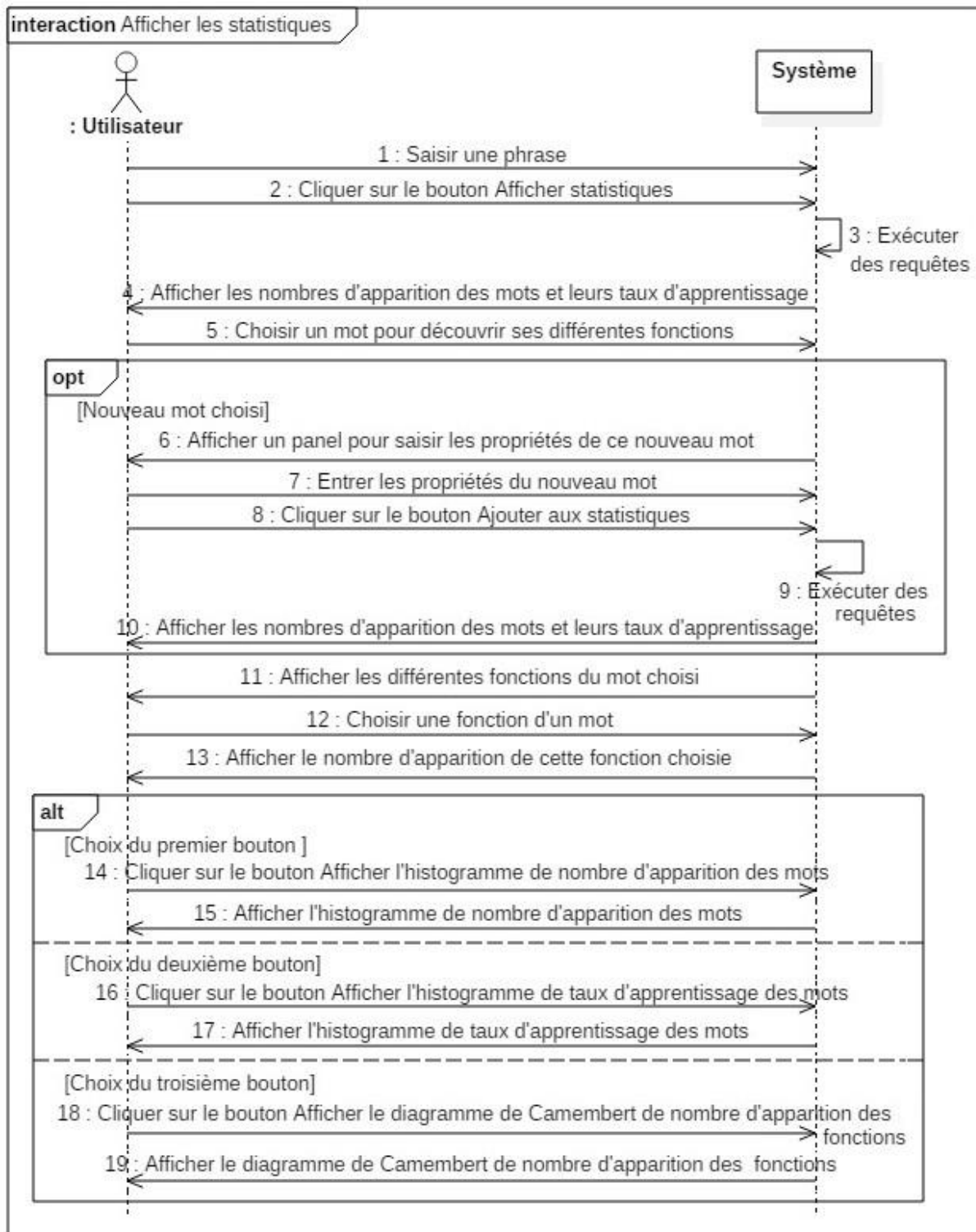


Figure 3.5: Diagramme de séquence d' Afficher les statistiques

II.2.2.3 Diagramme de séquence d' "Afficher les statistiques du chat"

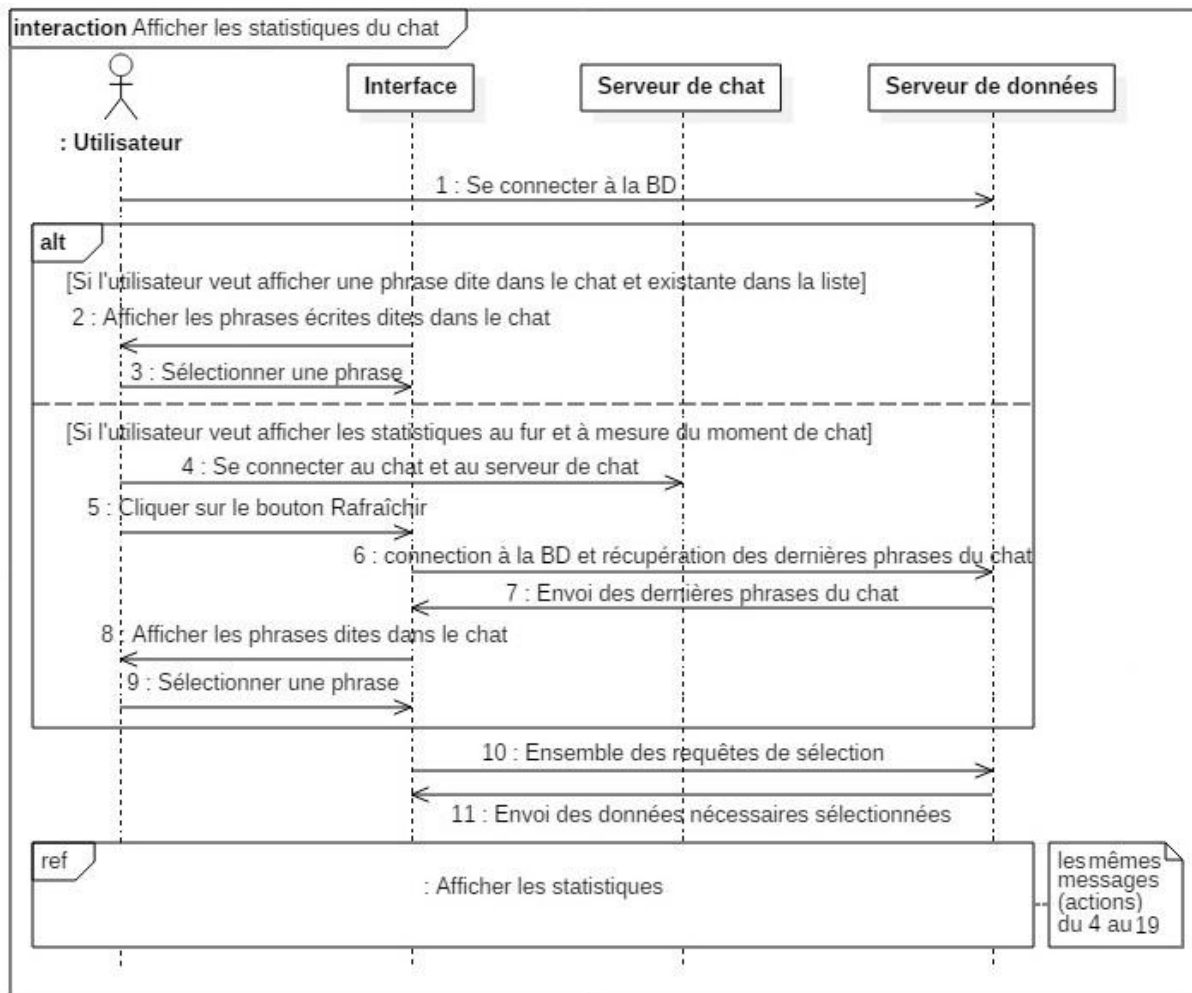


Figure 3.6: Diagramme de séquence d'Afficher les statistiques du chat

La figure 3.6 décrit le cas d'utilisation " Afficher les statistiques du chat ". Il suffit que l'utilisateur choisisse une phrase écrite dans le chat pour afficher les statistiques de ses mots qui sont le nombre d'apparitions de ces mots dans la BD et leurs taux d'apprentissage. Pareil au cas d'utilisation " Afficher les statistiques ", l'utilisateur peut ajouter un mot à la BD et afficher des graphes de statistiques. Après avoir présenté la conception globale et détaillée de notre application, nous passons au développement du réseau de neurones.

III. Procédure de développement d'un réseau de neurones

Le cycle classique de développement d'un réseau de neurones peut être séparé en sept étapes :

1. La collecte des données,
2. L'analyse des données,
3. La séparation des bases de données,
4. Le choix d'un réseau de neurones,
5. La mise en forme des données,

6. L'apprentissage,
7. La validation.

Nous éclaircissons ces étapes dans ce qui suit.

1. Collecte des données

L'objectif de cette étape est de recueillir des données, à la fois pour développer le réseau de neurones et pour le tester. Dans le cas d'applications sur des données réelles, l'objectif est de rassembler un nombre de données suffisant pour constituer une base représentative des données susceptibles d'intervenir en phase d'utilisation du système neuronal.

2. Analyse des données

Il est souvent préférable d'effectuer une analyse des données de manière à déterminer les caractéristiques discriminantes pour détecter ou différencier ces données. Ces caractéristiques constituent l'entrée du réseau de neurones. Il est généralement nécessaire de présenter des caractéristiques représentatives.

Cette détermination des caractéristiques a des conséquences à la fois sur la taille du réseau (et donc le temps de simulation), sur les performances du système (pouvoir de séparation lors d'un problème de classification, taux de détection), et sur le temps de développement (temps d'apprentissage).

Une étude statistique sur les données pourrait écarter celles qui sont aberrantes et redondantes. Par exemple, dans le cas d'un problème de classification, il appartient à l'expérimentateur de déterminer le nombre de classes auxquelles ses données appartiennent et de déterminer pour chaque donnée la classe à laquelle elle appartient.

3. Séparation des bases de données

Afin de développer une application à base de réseaux de neurones, il est nécessaire de disposer de deux bases de données : une base pour effectuer l'apprentissage et une autre pour tester le réseau obtenu et de déterminer ses performances. Afin de contrôler la phase d'apprentissage, il est souvent préférable de posséder une troisième base de données appelée base de validation croisée.

Il n'y a pas de règle pour déterminer ce partage de manière quantitative. Il résulte souvent d'un compromis tenant compte du nombre de données dont nous disposons et du temps imparti pour effectuer l'apprentissage.

4. Choix d'un réseau de neurones

Il existe un grand nombre de types de réseaux de neurones, avec pour chacun des avantages et des inconvénients. Le choix d'un réseau peut dépendre:

- de la tâche à effectuer (classification, association, contrôle de processus, séparation aveugle de sources...),
- de la nature des données,
- d'éventuelles contraintes d'utilisation temps-réel (certains types de réseaux de neurones, tel que la "machine de Boltzmann", nécessitant des tirages aléatoires et un nombre de cycles de calculs indéfini avant stabilisation du résultat en sortie, présentent plus de contraintes que d'autres réseaux pour une utilisation temps-réel),
- des différents types de réseaux de neurones disponibles dans le logiciel de simulation que nous utilisons (à moins de les programmer).

Ce choix est aussi en fonction de la maîtrise ou de la connaissance que nous avons de certains réseaux, ou encore du temps dont nous disposons pour tester une architecture prétendue plus performante.

5. Mise en forme des données pour un réseau de neurones

De manière générale, les bases de données doivent subir un prétraitement afin d'être adaptées aux entrées et sorties du réseau de neurones. Un prétraitement courant consiste à effectuer une normalisation appropriée, qui tienne compte de l'amplitude des valeurs acceptées par le réseau.

6. Apprentissage du réseau de neurones

Tous les modèles de réseaux de neurones requièrent un apprentissage. Plusieurs types d'apprentissage peuvent être adaptés à un même type de réseau de neurones. Les critères de choix sont souvent la rapidité de convergence ou les performances de généralisation.

Le critère d'arrêt de l'apprentissage est souvent calculé à partir d'une fonction de coût, caractérisant l'écart entre les valeurs de sortie et les valeurs de références (réponses souhaitées pour chaque exemple présenté).

Certains algorithmes d'apprentissage se chargent de la détermination des paramètres architecturaux des réseaux de neurones. Si nous n'utilisons pas ces techniques, l'obtention des paramètres architecturaux optimaux se fera par comparaison des performances obtenues pour différentes architectures de réseaux de neurones. Des contraintes dues à l'éventuelle réalisation matérielle du réseau peuvent être introduites lors de l'apprentissage.

7. Validation

Une fois le réseau de neurones entraîné (après apprentissage), il est nécessaire de le tester sur une base de données différente de celle utilisée pour l'apprentissage. Ce test permet à la fois d'apprécier les performances du système neuronal et de détecter le type de données qui pose

le problème. Si les performances ne sont pas satisfaisantes, il faudra soit modifier l'architecture du réseau, soit modifier la base d'apprentissage (caractéristiques discriminantes ou représentation des données).

Conclusion

Nous avons consacré ce chapitre à la description de notre solution et approche proposées. Nous avons dégagé les exigences de notre système, analysé ses besoins afin d'appréhender rapidement le fonctionnement général et de comprendre les détails de chaque fonctionnalité. Puis, nous avons présenté la conception de notre plateforme. Nous avons exposé dans un premier temps, la conception générale et dans un second temps nous avons expliqué la conception détaillée à travers une description des diagrammes de classes et de séquences. Enfin, nous avons clôturé ce chapitre par l'énumération des étapes nécessaires au développement du notre réseau de neurones. Dans le chapitre suivant, nous mettons en œuvre l'approche proposée et nous interprétons les résultats obtenus.

Chapitre 4

Réalisation

Introduction

Ce chapitre décrit les différentes tâches de la réalisation de la solution proposée. L'environnement matériel et logiciel, utilisé pour le développement, ainsi que les technologies adoptées sont l'objet de la première partie suivie par une présentation du travail effectué étayée par des imprimés écrans.

4.1 Environnement de travail

Nous dédions cette section à la présentation de l'environnement matériel et logiciel utilisé pour la réalisation du projet.

4.1.1 Environnement matériel

Pendant les différentes phases de ce projet, nous avons disposé d'un PC ayant les caractéristiques suivantes :

- ❖ PC portable : Dell,
- ❖ RAM : 6 Go,
- ❖ Disque dur : 322 Go,
- ❖ Processeur : Intel(R) Core(TM) i5-2410M CPU @ 2.30 GHz 2.30 GHz,
- ❖ Système d'exploitation : Microsoft Windows 7.

4.1.2 Environnement logiciel

Nous nous sommes servis tout le long de la phase de développement de l'environnement logiciel suivant :

- ❖ Netbeans comme environnement de développement JAVA,
- ❖ Navigateur Web Google Chrome,
- ❖ Interface phpMyAdmin de WampServer,
- ❖ MySQL comme S.G.B.D,
- ❖ Serveur Openfire du chat,
- ❖ Messagerie instantanée Spark,
- ❖ NeurophStudio pour le réseau de neurones,
- ❖ Star UML pour la modélisation UML,
- ❖ Microsoft Office.

4.2 Choix techniques

4.2.1 Choix du langage de programmation Java

Java est un langage de programmation purement orienté objet. Java est un langage de programmation puissant conçu pour être sûr, portable et international. C'est un environnement de développement qui est continuellement étendu pour fournir de nouvelles

caractéristiques et bibliothèques permettant de gérer de manière élégante des problèmes complexes. Du point de vue du programmeur, Java permet de réduire le temps de développement d'une application grâce à la réutilisation du code développé. Toutes ces caractéristiques nous justifient le choix de ce langage.

4.2.2 Choix de l'environnement de développement NetBeans

NetBeans est un environnement de développement intégré (EDI) open source. En plus de Java, Netbeans permet également de supporter différents autres langages. Il comprend toutes les caractéristiques d'un IDE moderne. Il est disponible sous Windows, Linux, Solaris, Mac OS X. Il constitue par ailleurs une plate-forme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)) [21]. Nous avons utilisé la bibliothèque libre JFreeChart qui permet de représenter des données statistiques sous forme graphiques. Ces graphiques reposent sur les différents modes de représentations typiques tels que les courbes, les histogrammes et les camemberts.

4.2.3 Choix de la base de données MySQL

Nous avons choisi la base de données MySQL qui est open source. Sa simplicité, sa bonne documentation et ses performances honorables en ont rapidement fait un système de base de données très populaire. Il fonctionne sur beaucoup de plateformes différentes incluant Linux, Windows, etc. Il est aussi accessible en utilisant des langages de programmation en C, C++, C#, Java, Perl, PHP, Python, etc. [22].

4.2.4 Choix du chat Spark

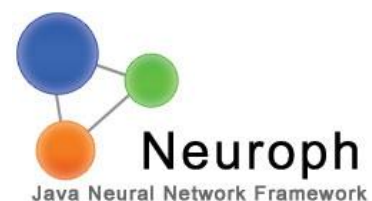
Spark est un client de messagerie instantanée pour le réseau standard ouvert Jabber (XMPP). C'est un logiciel libre, multiplateforme. Cet open source est optimisé pour une utilisation professionnelle. C'est un



moyen sécurisé de discuter en VoIP, ou par chat, sans devoir passer par des serveurs publics. Il propose une interface personnalisable et jolie, plusieurs packs de smileys, la gestion multi-protocoles et des groupes sur un réseau privé. Combiné à Openfire, un serveur XMPP privé et open source, qui peut utiliser une base utilisateur LDAP ou ActiveDirectory, nous offre la possibilité de l'ajouter un plugin (Monitoring Plugin) afin de récupérer les messages de chat et c'est ce que nous cherchons pour notre projet.

4.2.5 Choix de Neuroph Studio

Neuroph Studio est un environnement de développement des réseaux de neurones en Java pour développer des architectures



de réseaux de neurones. Il simplifie le développement des RN en fournissant un outil GUI qui prend en charge la création, la formation et l'enregistrement de ces réseaux. Il est recommandé pour les débutants. Nous pouvons expérimenter avec les architectures des RN dans ce Studio puis utiliser la bibliothèque Neuroph pour bénéficier de ces réseaux dans nos programmes Java. Ce cadre est bien documenté, facile à utiliser et très flexible.

4.3 Phases d'implémentation

Nous décrivons dans cette section l'ensemble des étapes qui ont servi à la réalisation de notre plateforme d'apprentissage linguistique de textes français par réseaux de neurones. Nous commençons par dévoiler les interfaces graphiques qui constituent un élément indispensable dans la réussite d'un système informatique. Ces IHM (Interfaces Homme-Machine) doivent obéir à des chartes graphiques et aux règles d'ergonomie pour que l'utilisateur s'adapte facilement à la logique de l'application. L'application débute par le lancement de l'interface de la figure 4.1.

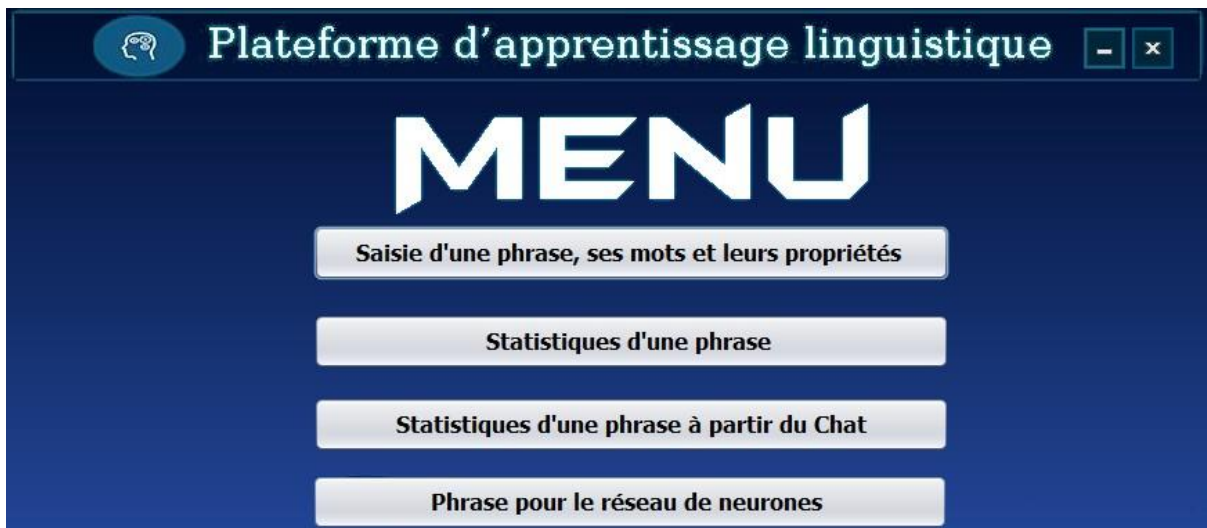


Figure 4.1: Le menu de notre plateforme d'apprentissage linguistique

4.3.1 Collecte et recueil de données

Dès que l'utilisateur clique sur le bouton "Saisie d'une phrase, ses mots et leurs propriétés", l'interface correspondante s'affiche. D'abord, il lui faut entrer une phrase. Puis, le système lui calcule le nombre de mots dans cette phrase et vérifie avec lui si ce nombre est correct ou non. Si c'est incorrect, il lui donne la possibilité de corriger. Voici un exemple illustratif.



Figure 4.2: La boîte de dialogue pour vérifier le nombre de mots d'une phrase

Une fois la phrase et son nombre de mots saisis, l'utilisateur clique sur le bouton "Ajouter" pour l'ajouter à la base de données. Le système lui affiche dans la première table la phrase saisie et le nombre de ses mots. Par la suite, l'utilisateur doit entrer chaque mot composant cette phrase et ses propriétés: le pourcentage d'importance du mot (nombre entier] 0,100[, sa fonction correspondante et sa position. S'il n'a pas trouvé la fonction adéquate, le système lui permet d'ajouter une nouvelle fonction. La plateforme lui affiche alors dans la deuxième table ce mot et ses propriétés (voir figure 4.3).

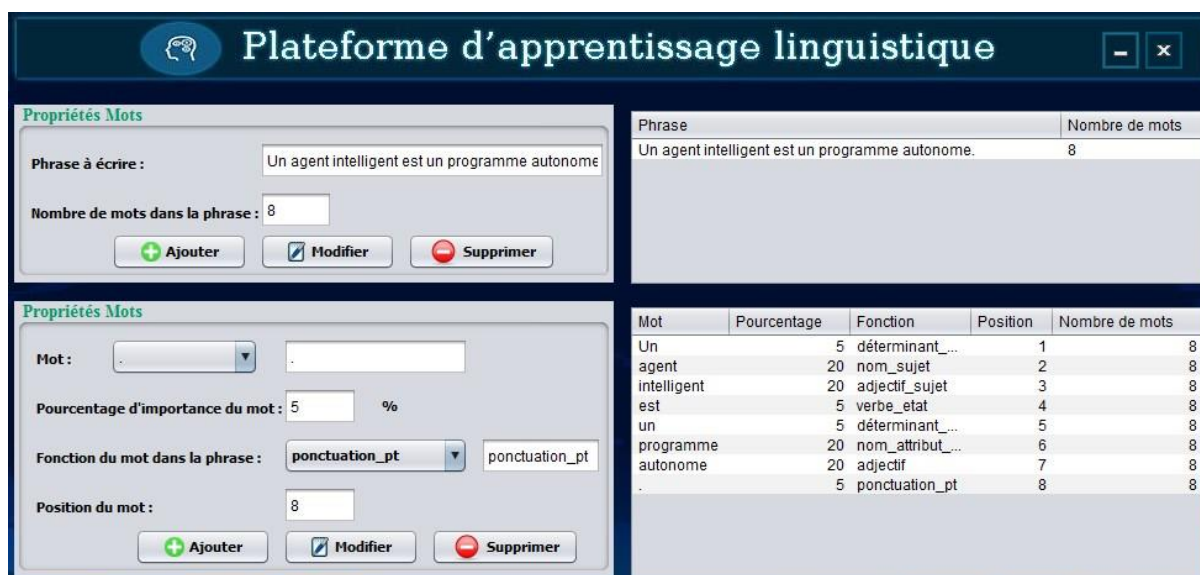


Figure 4.3: Un exemple décrivant le cas d'utilisation "Saisir une phrase, ses mots et leurs propriétés"

Le résultat dans la base de données est décrit consécutivement dans les deux figures 4.4 et 4.5 qui montrent l'ajout de cette phrase à la table "phrase" et l'ajout de tous ses mots et leurs propriétés dans la table "mots".

	Id_phrase	Phrase
<input type="checkbox"/>	66	Va-t'en.
<input type="checkbox"/>	68	C'est la loi n'est-ce pas.
<input type="checkbox"/>	69	L'été est chaud.
<input type="checkbox"/>	70	C'est lundi.
<input type="checkbox"/>	71	Révolution tunisienne.
<input type="checkbox"/>	72	:-)
<input type="checkbox"/>	74	Welcome April!
<input type="checkbox"/>	75	Bill Gates est un informaticien et un entrepreneur...
<input type="checkbox"/>	76	Aïe!
<input type="checkbox"/>	77	j'attends Monsieur
<input type="checkbox"/>	78	Le soleil brille.
<input type="checkbox"/>	79	Facebook est le réseau social le plus populaire.
<input type="checkbox"/>	80	Google est très populaire.
<input type="checkbox"/>	81	Bravo!
<input type="checkbox"/>	82	Réseaux de neurones!
<input type="checkbox"/>	83	C'est lundi.
<input type="checkbox"/>	84	Les réseaux de neurones artificiels sont construit...
<input type="checkbox"/>	85	Un agent intelligent est un programme autonome.

Figure 4.4: La phrase saisie est ajoutée à la table "phrase" de la BD

	Id_mot	Mot	Pourcentage	Fonction	Position	Nombre_de_mots	Numéro_phrase
<input type="checkbox"/>	214	.	1	punctuation_pt	4	4	83
<input type="checkbox"/>	215	Les	4	déterminant_sujet	1	12	84
<input type="checkbox"/>	216	réseaux	10	nom_sujet	2	12	84
<input type="checkbox"/>	217	de	4	préposition	3	12	84
<input type="checkbox"/>	218	neurones	15	nom_sujet	4	12	84
<input type="checkbox"/>	219	artificiels	15	adjectif_sujet	5	12	84
<input type="checkbox"/>	220	sont	10	verbe_etat	6	12	84
<input type="checkbox"/>	221	construits	10	verbe	7	12	84
<input type="checkbox"/>	222	sur	4	préposition_attribut_sujet	8	12	84
<input type="checkbox"/>	223	un	4	déterminant_attribut_sujet	9	12	84
<input type="checkbox"/>	224	paradigme	10	nom_attribut_sujet	10	12	84
<input type="checkbox"/>	225	biologique	10	adjectif_COD	11	12	84
<input type="checkbox"/>	226	.	4	punctuation_pt	12	12	84
<input type="checkbox"/>	227	heureuse	64.4564	adjectif	3	0	0
<input type="checkbox"/>	228	s'	4.38095	pronom réfléchi	3	0	0
<input type="checkbox"/>	229	appellent	17.5238	verbe	4	0	0
<input type="checkbox"/>	230	Un	5	déterminant_sujet	1	8	85
<input type="checkbox"/>	231	agent	20	nom_sujet	2	8	85
<input type="checkbox"/>	232	intelligent	20	adjectif_sujet	3	8	85
<input type="checkbox"/>	233	est	5	verbe_etat	4	8	85
<input type="checkbox"/>	234	un	5	déterminant_attribut_sujet	5	8	85
<input type="checkbox"/>	235	programme	20	nom_attribut_sujet	6	8	85
<input type="checkbox"/>	236	autonome	20	adjectif	7	8	85
<input type="checkbox"/>	237	.	5	punctuation_pt	8	8	85

Figure 4.5 : Les mots de la phrase sont ajoutés à la table "mots" de la BD

Notons que la somme des pourcentages d'importance de tout les mots composant la phrase doit égale à 100 %. Sinon, le système affiche un avertissement. La figure suivante montre, un

exemple de phrase : "Nous testons la première interface." où nous avons ajouté un mot avec un pourcentage incorrect.

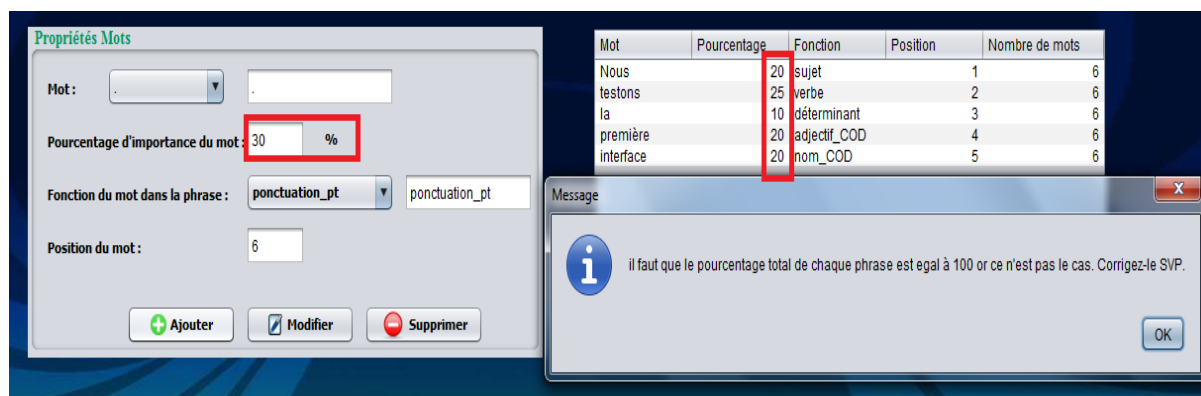


Figure 4.6 : Ajout d'un mot avec un pourcentage incorrect

L'utilisateur peut aussi modifier la phrase, ou les mots ou leurs propriétés. Ci-après, par exemple, nous avons changé la fonction du mot "est" de "verbe_etat" au "verbe".

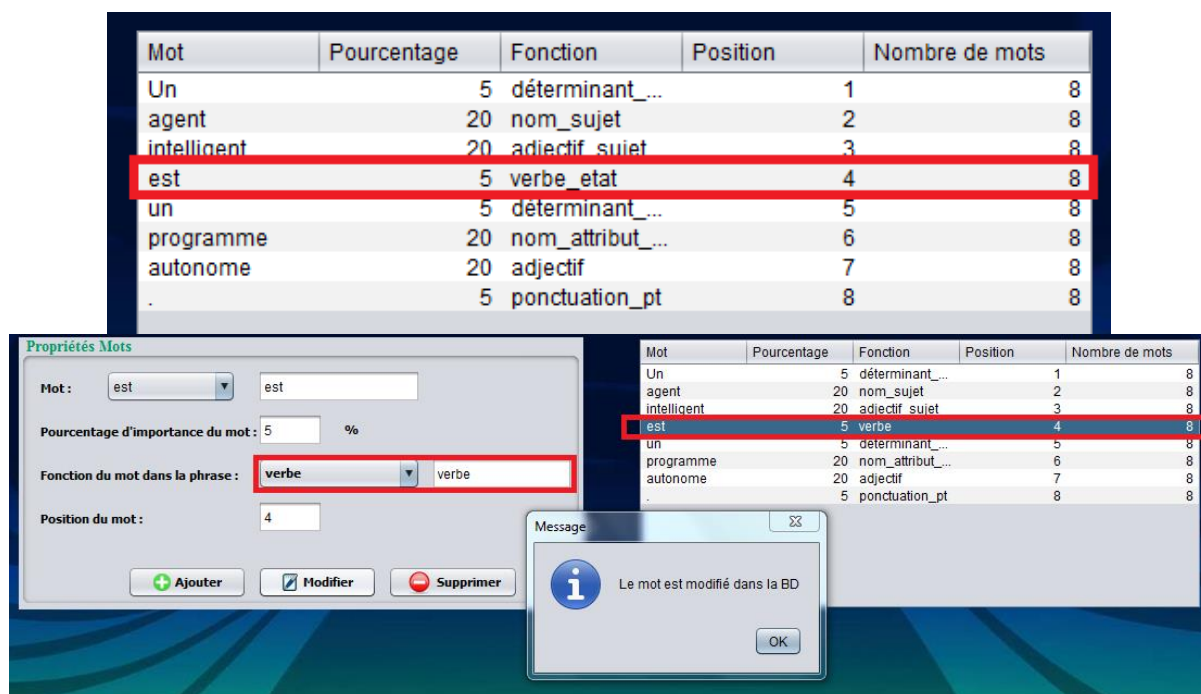


Figure 4.7: Exemple de modification d'une propriété (la fonction) d'un mot

Voici la modification dans la BD.

<input type="checkbox"/>			230	Un	5	déterminant_sujet	1	8	85
<input type="checkbox"/>			231	agent	20	nom_sujet	2	8	85
<input type="checkbox"/>			232	intelligent	20	adjectif_sujet	3	8	85
<input type="checkbox"/>			233	est	5	verbe	4	8	85
<input type="checkbox"/>			234	un	5	déterminant_attribut_sujet	5	8	85
<input type="checkbox"/>			235	programme	20	nom_attribut_sujet	6	8	85
<input type="checkbox"/>			236	autonome	20	adjectif	7	8	85
<input type="checkbox"/>			237	.	5	ponctuation_pt	8	8	85

Figure 4.8: La modification de la fonction du mot dans la BD

En outre, l'utilisateur peut supprimer un mot ou une phrase (voir figure4.9):

Mot	Pourcentage	Fonction	Position	Nombre de mots
Nous	20	sujet	1	6
testons	25	verbe	2	6
la	10	déterminant	3	6
première	20	adjectif COD	4	6
interface	20	nom COD	5	6
.	5	punctuation_pt	6	6

Figure 4.9: La suppression d'un mot dans la table de l'interface

Le résultat de la suppression dans la BD est présenté par la figure 4.10 :

<input type="checkbox"/>			238	Nous	20	sujet	1	6	86
<input type="checkbox"/>			239	testons	25	verbe	2	6	86
<input type="checkbox"/>			240	la	10	déterminant	3	6	86
<input type="checkbox"/>			241	première	20	adjectif_COD	4	6	86
<input type="checkbox"/>			243	.	5	punctuation_pt	6	6	86

Figure 4.10: La suppression d'un mot dans la BD

4.3.2 Statistiques et calcul de taux d'apprentissage

Pour afficher les statistiques, l'utilisateur doit taper une phrase et cliquer sur le bouton "Afficher les statistiques". Les statistiques sont le nombre d'apparitions de chaque mot dans la BD et le taux d'apprentissage affichés à l'utilisateur dans une table (voir Figure 4.11).

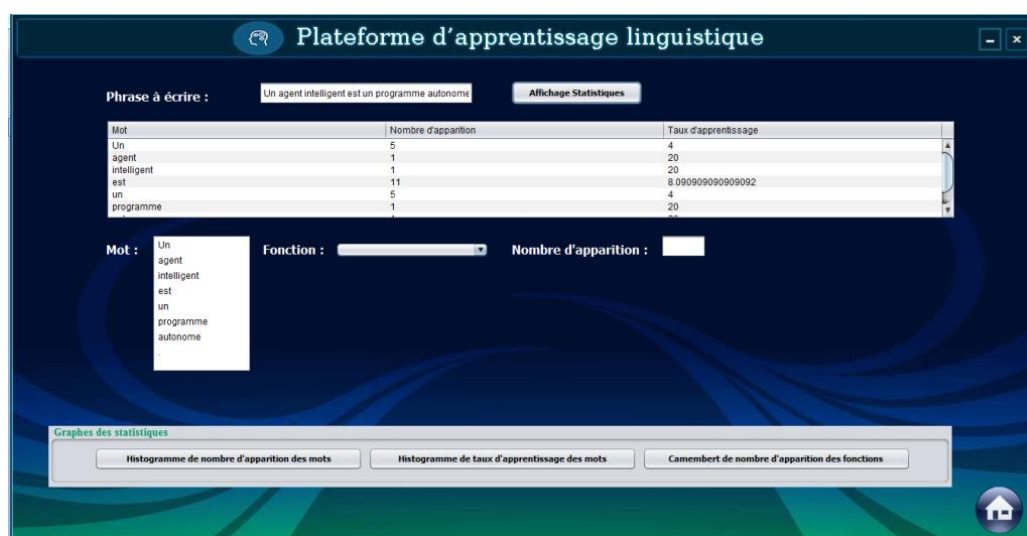


Figure 4.11: L'interface du cas d'utilisation "Afficher les statistiques"

A l'examen de cette interface, nous constatons que:

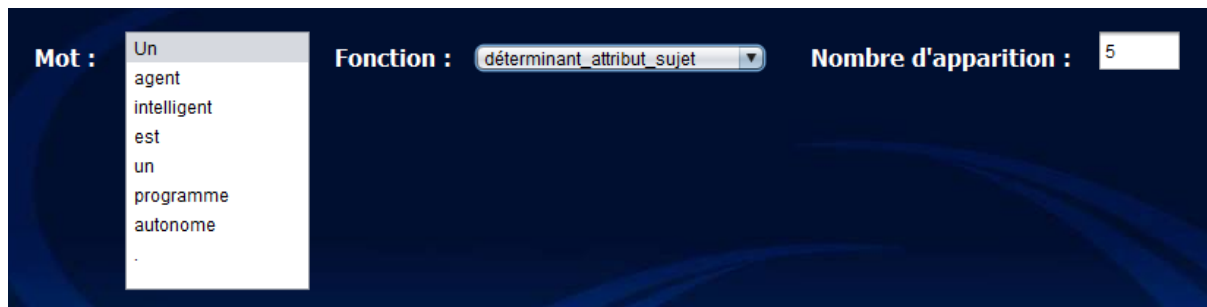


Mot	Nombre d'apparition	Taux d'apprentissage
Un	5	4
agent	1	20
intelligent	1	20
est	11	8.090909090909092
un	5	4
programme	1	20

Figure 4.12: Les statistiques des mots d'une phrase

Le nombre d'apparitions concerne les mots de la BD. Le taux d'apprentissage est un nombre réel qui représente la moyenne des différentes valeurs de pourcentage que prend le mot. Par exemple, les mots "agent", "intelligent" et "programme" de la phrase que nous avons déjà saisie, existent une seule fois dans la BD, et donc chacun de ces mots a comme taux d'apprentissage la valeur du pourcentage entrée lors de l'ajout. Prenons l'exemple du point de ponctuation ".", ce dernier existe plusieurs fois dans la BD. Notons ce nombre par N , avec des différents pourcentages : p_1, p_2, \dots, p_N , le taux d'apprentissage de ce mot est égal à $\sum_{i=1}^N p_i = \frac{p_1 + p_2 + \dots + p_N}{N}$.

Cette interface permet aussi d'afficher les différentes fonctions que prend le mot et le nombre d'apparitions de cette fonction dans la BD (voir Figure 4.13).



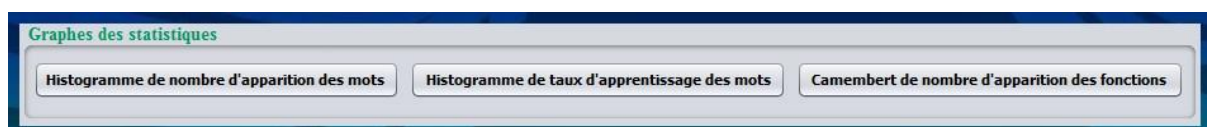
Mot : Un, agent, intelligent, est, un, programme, autonome, .

Fonction : déterminant_attribut_sujet

Nombre d'apparition : 5

Figure 4.13: Une fonction que prend un mot et son nombre d'apparitions

Cette interface affiche aussi des graphes de statistiques (voir Figure 4.14).



Graphes des statistiques

Histogramme de nombre d'apparition des mots Histogramme de taux d'apprentissage des mots Camembert de nombre d'apparition des fonctions

Figure 4.14: Les boutons des graphes de statistiques

Il s'agit de trois boutons relatifs à l'histogramme du nombre d'apparitions des mots, l'histogramme de taux d'apprentissage des mots et le camembert du nombre d'apparitions des fonctions. En cliquant sur le premier bouton, le système affiche le graphe suivant:

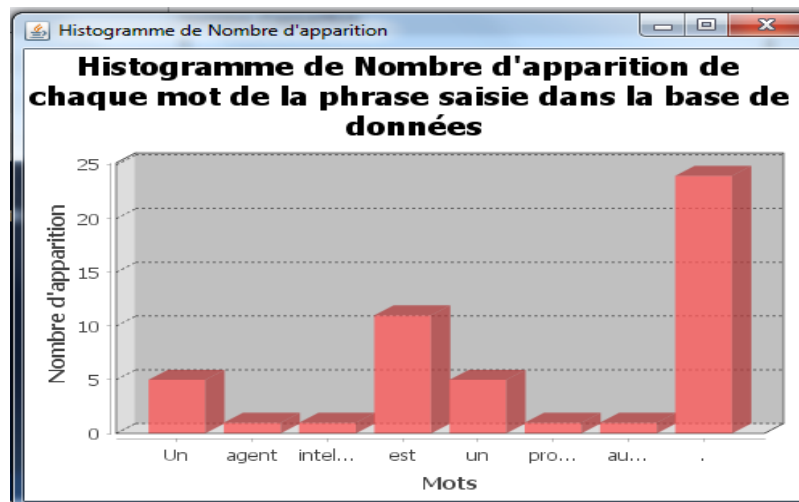


Figure 4.15: Histogramme de nombre d'apparitions des mots

Pour la phrase "Un agent intelligent est un programme autonome.", l'application affiche pour chaque mot, son nombre d'apparitions dans la BD. Le deuxième bouton correspond à l'histogramme de taux d'apprentissage des différents mots:

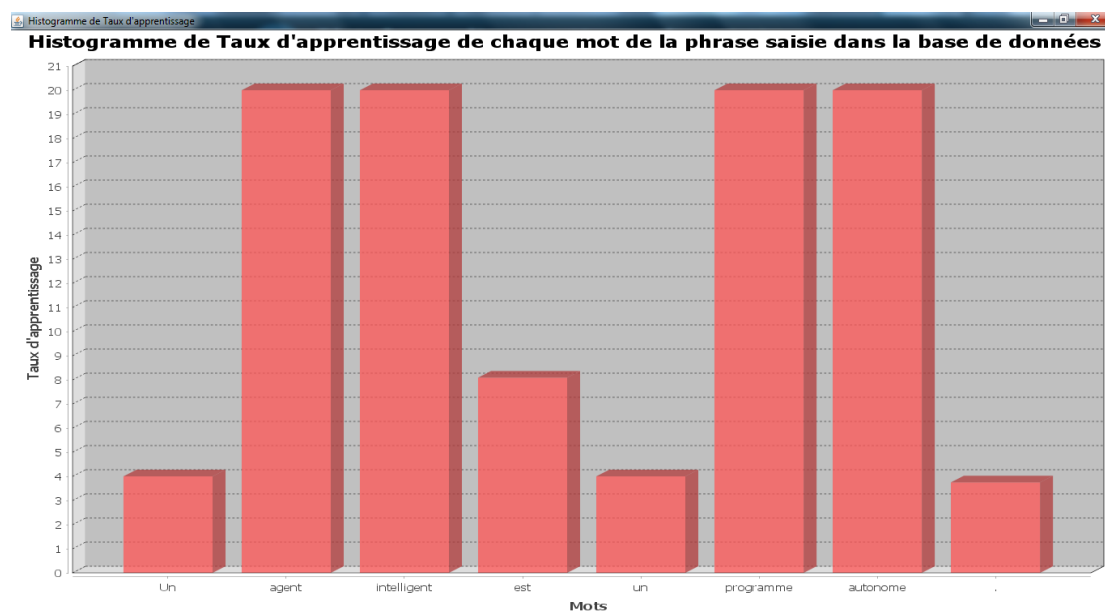


Figure 4.16: Histogramme de taux d'apprentissage des mots

Ici, par exemple, le taux d'apprentissage du mot "un" est égal à 4 et du mot "autonome" est égal à 20. Le troisième bouton "Camembert de nombre d'apparitions des fonctions" donne une idée sur les fonctions qui se répètent beaucoup dans la BD. Comme le montre la figure 4.17, le point de ponctuation "." existe presque toujours dans les phrases et donc nous voyons que sa portion est grande.

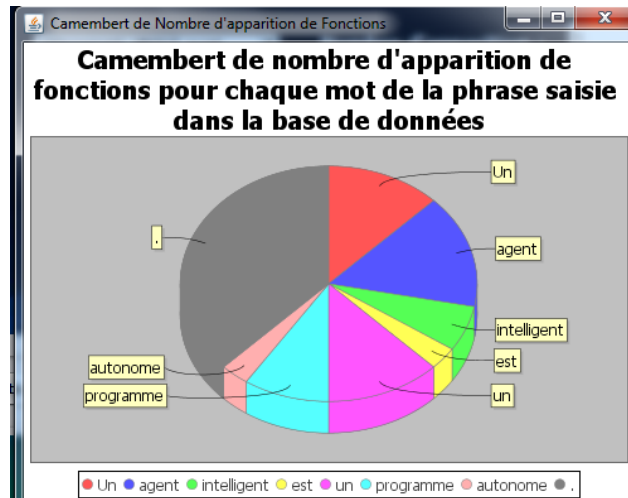


Figure 4.17: Camembert de nombre d'apparitions de fonctions pour chaque mot

Quand l'utilisateur saisit une phrase pour découvrir ses statistiques, il se peut que quelques mots de cette phrase n'existent pas dans la BD. Ce sont des nouveaux mots qu'il faut ajouter à la BD en tapant leurs propriétés et en cliquant sur le bouton "Ajouter aux statistiques". Ces mots seront ainsi ajoutés dans la BD et leurs statistiques s'affichent dans la table. Cette capture d'écran est prise pendant que l'utilisateur ajoute un nouveau mot :

Phrase à écrire : Cortana est un assistant virtuel de Microsoft. Affichage Statistiques

Mot	Nombre d'apparition	Taux d'apprentissage
Cortana	0	
est	11	8.090909090909092
un	5	4
assistant	0	
virtuel	0	
de	2	4

Mot : Cortana Fonction : Nombre d'apparition :

est
un
assistant
virtuel
de
Microsoft

Mot : Cortana

Pourcentage d'importance du mot : Ajouter aux Statistiques

Fonction du mot dans la phrase : Ajouter une fonction

Position du mot :

Figure 4.18: Imprime-écran pour l'ajout d'un nouveau mot

Une fois ajouté, le nouveau mot aura 1 comme nombre d'apparitions. Mais, il faut déterminer son taux d'apprentissage. Le calcul de ce taux diffère un peu de ce que nous avons fait précédemment. Prenons par exemple, la phrase "Cortana est un assistant virtuel de Microsoft.". Comme le montre la figure 4.18 et le tableau ci-dessous, nous avons :

Mot	Nombre d'apparitions	Taux d'apprentissage
Cortana	0	—
est	11	8.0909
un	5	4
assistant	0	—
virtuel	0	—
de	2	4
Microsoft	0	—
.	24	3.75

Tableau 4.1: Table de statistiques

Calculons la somme des taux d'apprentissage des mots composant la phrase et existant dans la BD : $8.0909 + 4 + 4 + 3.75 = 19.8409$

Donc ces mots représentent cette somme de la totalité de la BD et d'où nous avons :

Dans BD : $19.8409 \Rightarrow 100\%$ (1)

Le taux d'apprentissage de chaque mot connu par le système dans cette phrase correspond au quotient de son taux d'apprentissage affiché par son nombre d'apparitions dans la BD :

- pour le mot "est" : $8.0909 / 11 = 0.7355$
- pour le mot "un" : $4 / 5 = 0.8$
- pour le mot "de" : $4 / 2 = 2$
- pour le mot "." : $3.75 / 24 = 0.1562$

La somme de ces taux d'apprentissage est: $0.7355 + 0.8 + 2 + 0.1562 = 3.6917$

Elle représente la somme des taux d'apprentissage dans la phrase, d'où nous retenons cette équation :

Dans phrase : $3.6917 \Rightarrow 100\%$ (2)

Nous proposons les pourcentages d'importance des mots à ajouter notés x_i (pour tout l'exemple i est le nombre de mots à ajouter) de cette façon :

- pour le mot "Cortana" : soit $x_1 = 30\%$,
- pour le mot "assistant" : soit $x_2 = 15\%$,
- pour le mot "virtuel" : soit $x_3 = 15\%$,
- pour le mot "Microsoft" : soit $x_4 = 20\%$.

Maintenant, nous introduisons la notion du "**poids**". Rappelons que dans un réseau de neurones artificiel, chaque entrée est associée à un poids qui représente la force de connexion avec le neurone suivant. Ces poids dans les RNA prennent des valeurs. Ici, nous présentons le calcul théorique de ces poids. Les nouveaux mots c.à.d. ceux que nous allons ajouter

possèdent des poids. En reliant avec notre exemple que nous sommes en train de traiter, soient ces poids notés y_i de cette manière :

- Soit le poids du mot "Cortana" y_1 que nous voulons le calculer :

En utilisant (2) et "la règle de trois", nous avons :

$$3.6917 \Rightarrow 100$$

$$y_1 = ? \Rightarrow x_1 = 30$$

$$\Rightarrow y_1 = \frac{30 \times 3.6917}{100} = 1.10751$$

- Soit le poids du mot "assistant" y_2 :

$$3.6917 \Rightarrow 100$$

$$y_2 = ? \Rightarrow x_2 = 15$$

$$\Rightarrow y_2 = \frac{15 \times 3.6917}{100} = 0.5537$$

- Soit le poids du mot "virtuel" y_3 :

$$3.6917 \Rightarrow 100$$

$$y_3 = ? \Rightarrow x_3 = 15$$

$$\Rightarrow y_3 = \frac{15 \times 3.6917}{100} = 0.5537$$

- Soit le poids du mot "Microsoft" y_4 :

$$3.6917 \Rightarrow 100$$

$$y_4 = ? \Rightarrow x_4 = 20$$

$$\Rightarrow y_4 = \frac{20 \times 3.6917}{100} = 0.7383$$

Dans ce qui suit, le calcul des taux d'apprentissage de chacun des quatre mots, notés TA_i :

À l'aide de (1) et de la règle de trois, nous obtenons :

- Pour le mot "Cortana" :

$$19.8409 \Rightarrow 100$$

$$x_1 + y_1 = 30 + 1.10751 = 31.10751 \Rightarrow ?$$

$$\Rightarrow TA_1 = \frac{31.10751 \times 100}{19.8409} = 156.784792$$

Et voici le résultat affiché à l'utilisateur par notre système :

Mot	Nombre d'apparition	Taux d'apprentissage
Cortana	1	156.78484
est	11	8.090909090909092
un	5	4
assistant	0	4
virtuel	0	4
de	2	4

Figure 4.19: Calcul du taux d'apprentissage d'un nouveau mot

- Pour le mot "assistant" :

$$19.8409 \Rightarrow 100$$

$$x_2 + y_2 = 15 + 0.5537 = 15.5537 \Rightarrow ?$$

$$\Rightarrow TA_2 = \frac{15.5537 \times 100}{19.8409} = 78.392$$

Cet imprime-écran nous montre l'ajout du mot "assistant" et la saisie du pourcentage d'importance qui est égal à 15.

Phrase à écrire : Cortana est un assistant virtuel de Microsoft. Affichage Statistiques

Mot	Nombre d'apparition	Taux d'apprentissage
Cortana	1	156.78484
est	11	8.090909090909092
un	5	4
assistant	0	
virtuel	0	
de	2	4

Mot : Cortana
est
un
assistant
virtuel
de
Microsoft

Fonction : Nombre d'apparition :

Mot : assistant

Pourcentage d'importance du mot : 15 Ajouter aux Statistiques

Fonction du mot dans la phrase : nom_COD nom_COD

Position du mot : 4

Et notre plateforme nous donne comme résultat :

Phrase à écrire : Cortana est un assistant virtuel de Microsoft. Affichage Statistiques

Mot	Nombre d'apparition	Taux d'apprentissage
Cortana	1	156.78484
est	11	8.090909090909092
un	5	4
assistant	1	78.39242
virtuel	0	
de	2	4

Figure 4.20: Taux d'apprentissage d'un nouveau mot

- Pour le mot "virtuel" :

$$19.8409 \rightarrow 100$$

$$x_3 + y_3 = 15 + 0.5537 = 15.5537 \rightarrow ?$$

$$\Rightarrow TA_3 = \frac{15.5537 \times 100}{19.8409} = 78.392$$

- Pour le mot "Microsoft" :

$$19.8409 \rightarrow 100$$

$$x_4 + y_4 = 20 + 0.73834 = 20.73834 \rightarrow ?$$

$$\Rightarrow TA_4 = \frac{20.73834 \times 100}{19.8409} = 104.523$$

Ces valeurs sont illustrées par la figure suivante :

Mot	Nombre d'apparition	Taux d'apprentissage
est	11	8.090909090909092
un	5	4
assistant	1	78.39242
virtuel	1	78.39242
de	2	4
Microsoft	1	104.523224
.	24	3.75

Figure 4.21: Les taux d'apprentissage de notre exemple

Rappelons que notre projet entre le cadre du TALN. Dans ce travail, nous nous sommes intéressés à la langue française qui se caractérise par la présence des apostrophes et des traits

d'union pour séparer les mots. Comme nous traitons des phrases, il a fallu tenir compte de certains cas particuliers dont voici des exemples.

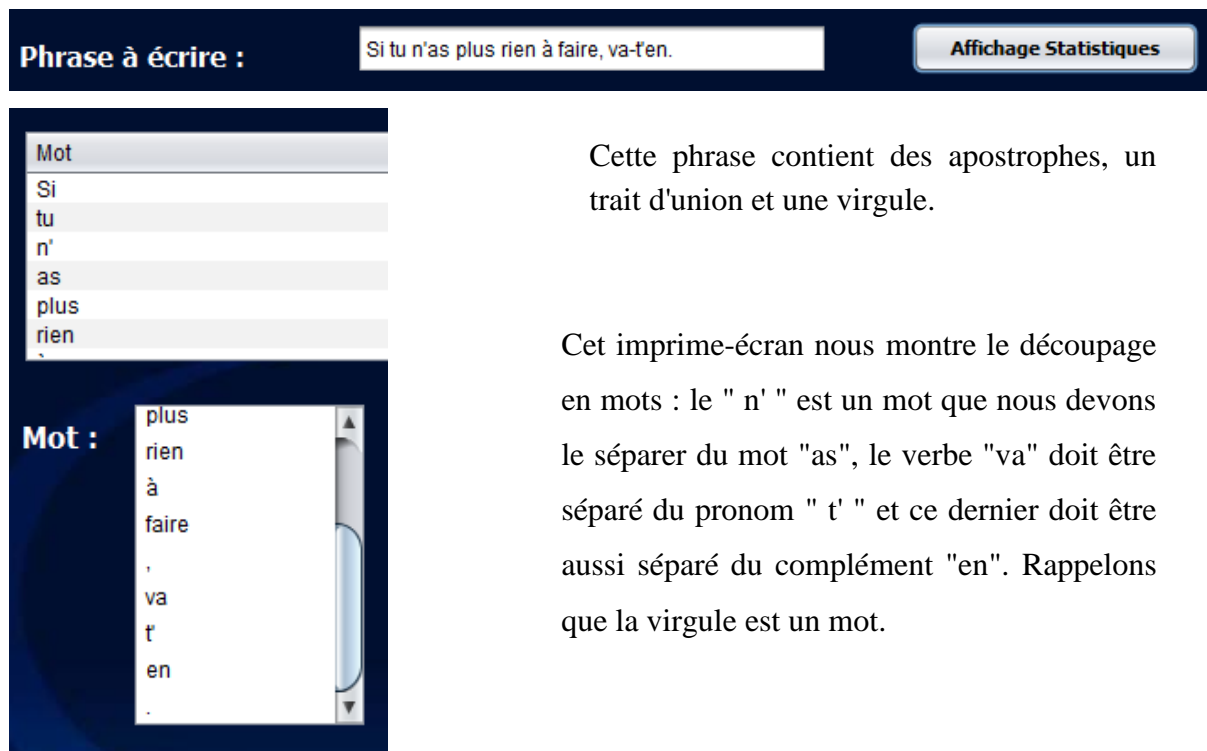


Figure 4.22: Exemple 1 de découpage en mots

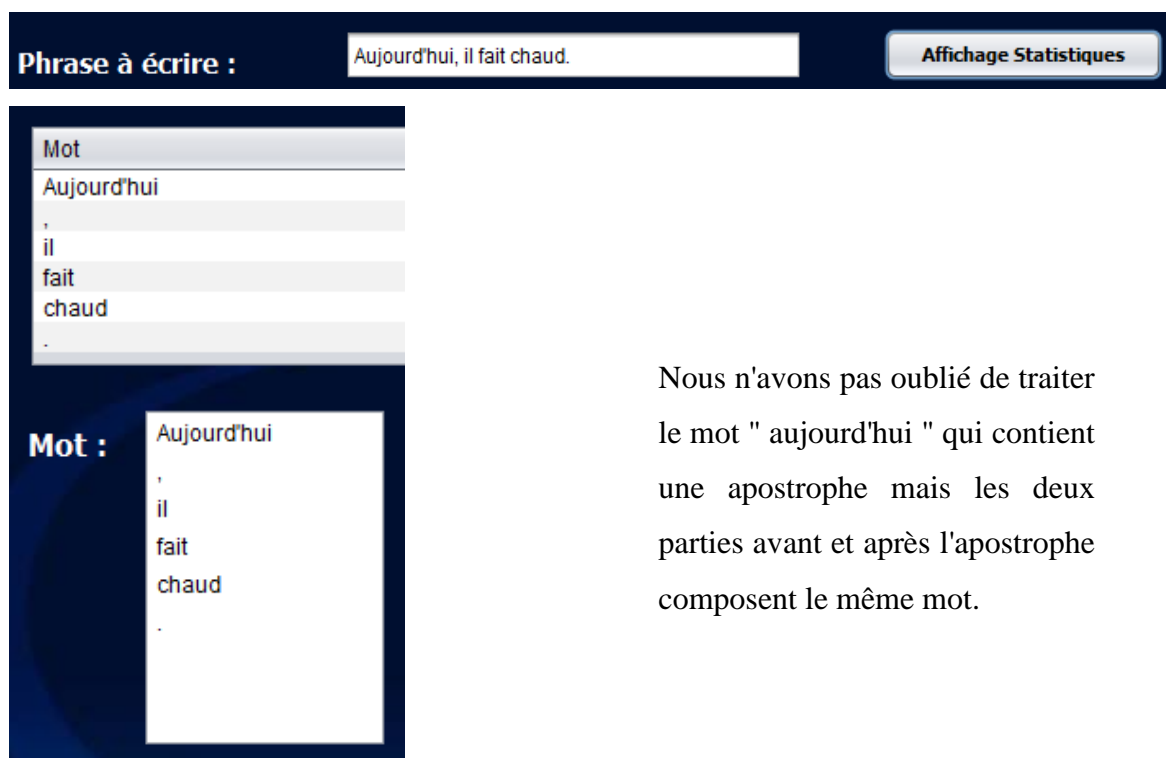


Figure 4.23: Exemple 2 de découpage en mots

Notre système offre à l'utilisateur non seulement la possibilité d'afficher des statistiques des phrases saisies mais aussi des phrases récupérées du chat lié à notre plateforme. L'interface

dédiée à cette fonctionnalité est presque la même que l'interface consacrée à l'affichage des statistiques des phrases saisies, mais juste l'utilisateur doit choisir une phrase écrite pendant une discussion instantanée pour découvrir ses statistiques. Pour rafraîchir les messages et les récupérer en temps-réel, il y a un bouton "Rafraîchir" (voir figure 4.24).

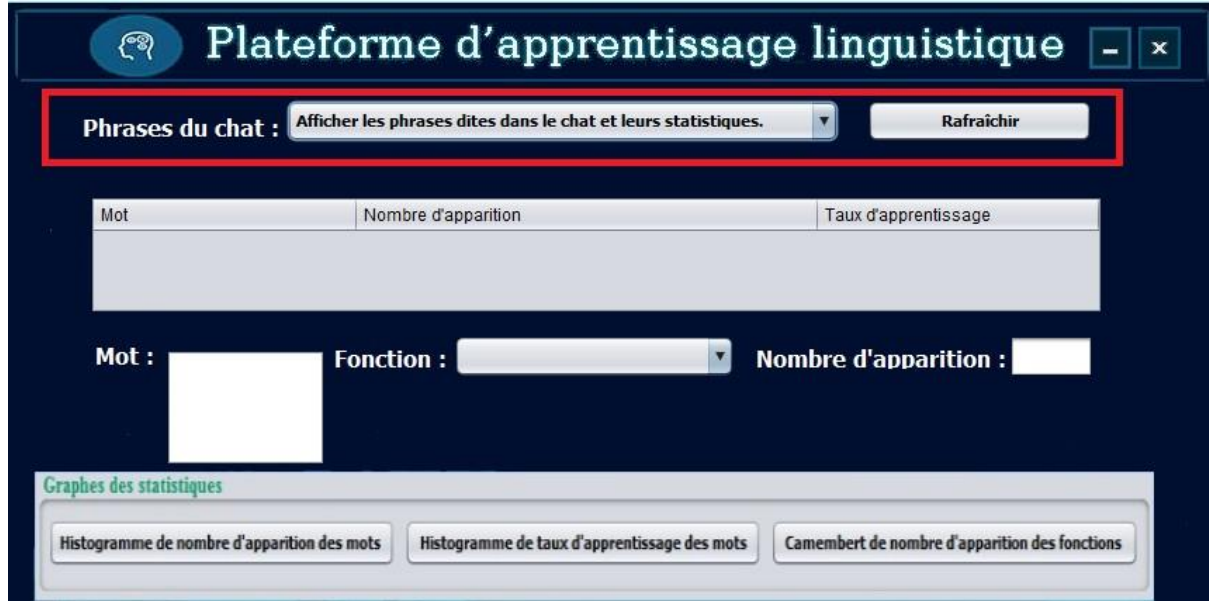


Figure 4.24: Interface pour afficher les statistiques d'une phrase écrite en chat

Nous avons choisi le chat Spark relié à son serveur Openfire parce qu'il offre plusieurs fonctionnalités. Nous avons ajouté à ce serveur le plugin "Monitoring Plugin" pour pouvoir archiver les messages du chat. La première interface pour ouvrir le chat est indiquée par la capture d'écran suivante :

Cette interface permet de créer des nouveaux comptes et de paramétrer les connexions et bien sûr de se connecter.



Figure 4.25: Ouverture du chat

Lorsque nous connectons au chat, l'interface illustrée par la figure 4.26 apparaît. Nous pouvons aussi changer de situation pendant le chat soit: connecté, absent, invisible, au téléphone, ne pas déranger.

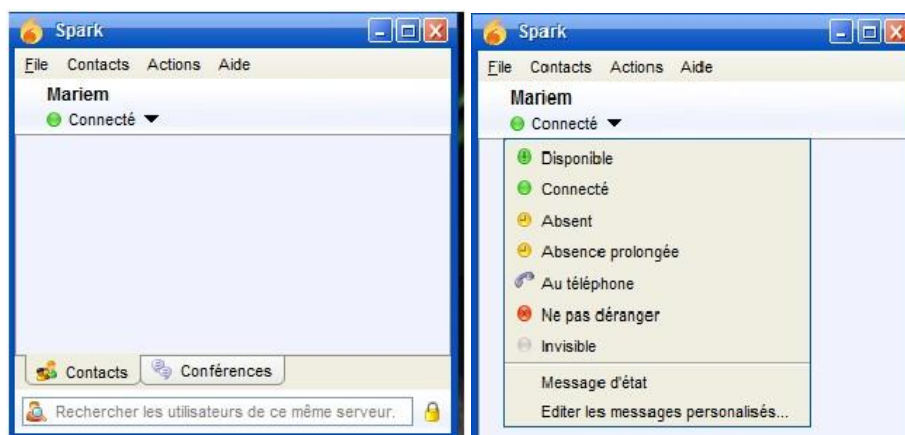


Figure 4.26: Connexion au chat et les différentes situations d'un utilisateur du chat

L'interface du chat est conviviale. Voici une figure qui montre une discussion.



Figure 4.27: Le chat

Les icônes à droite en haut sont des services : voir les informations à propos de l'utilisateur, inviter à une conférence, envoyer des fichiers à l'utilisateur, envoyer un screenshot, voir l'historique de conversation et chiffrement du message. La figure suivante nous confirme l'archivage dans la BD des phrases écrites dans cette conversation:

<input type="checkbox"/>			13	7	makram@user-pc	Spark	mariem@user-pc	NULL	1459327317817	<message id="kEG75-35" to="mariem@user-pc" from="m...	salut
<input type="checkbox"/>			14	7	mariem@user-pc	Spark	makram@user-pc	Spark	1459327329146	<message id="iQDCF-36" to="makram@user-pc/Spark" f...	Bonjour!
<input type="checkbox"/>			15	7	makram@user-pc	Spark	mariem@user-pc	Spark	1459327342484	<message id="kEG75-39" to="mariem@user-pc/Spark" f...	est ce que il est entrain de sauvegarder dans la b...
<input type="checkbox"/>			16	7	makram@user-pc	Spark	mariem@user-pc	Spark	1459327417050	<message id="kEG75-48" to="mariem@user-pc/Spark" f...	ok behi :)

Figure 4.28: Les phrases dans la BD

Les colonnes indiquent consécutivement l'id du message, l'id de la conversation, l'émetteur, la ressource (Spark), le récepteur, toJIDResource, la date d'envoi, stanza et enfin le corps du message.

Voilà un screenshot d'une autre conversation et la preuve de l'existence de ses phrases dans la BD.

	stanza	body
(20:23) Mariem: Bonsoir	<message id="95mEa-544" to="maryouma@user-pc" from=...	Bonsoir
(20:24) maryouma: bonsoir	<message id="1Avls-35" to="mariem@user-pc/Spark" f...	bonsoir
(20:28) maryouma: Comment ça va?	<message id="1Avls-50" to="mariem@user-pc/Spark" f...	Comment ça va?
(20:28) Mariem: Bien 😊	<message id="95mEa-569" to="maryouma@user-pc/Spark..."	Bien :-)
(20:28) maryouma: Où est-tu?	<message id="1Avls-58" to="mariem@user-pc/Spark" f...	Où est-tu?
(20:35) Mariem: À la maison 🏠 Je suis entrain d'écrire mon rapport de PFE 🧐	<message id="95mEa-601" to="maryouma@user-pc/Spark..."	À la maison :]) Je suis entrain d'écrire mon rappo...
(20:38) maryouma: Est-ce que tu as avancé? 😊	<message id="1Avls-98" to="mariem@user-pc/Spark" f...	Est-ce que tu as avancé? :-p
(20:39) Mariem: Oui j'avance 😊	<message id="95mEa-620" to="maryouma@user-pc/Spark..."	Oui j'avance :-)
La notification d'alerte a bien été envoyée à l'utilisateur	<message id="1Avls-126" to="mariem@user-pc/Spark" ...	In challah mention très bien
(20:41) maryouma: In challah mention très bien	<message id="1Avls-133" to="mariem@user-pc/Spark" ...	:x :-)
(20:42) maryouma: 🤔🤔		
(20:42) Mariem: Merci 🍀		

Nous remarquons que Spark utilise les couleurs pour les noms d'utilisateurs. En plus, il offre les émoticônes (Lorsque nous cliquons sur l'icône où il y a le smiley, une liste de smileys est proposée à l'utilisateur.).

Figure 4.29: Autre discussion et ses phrases dans la BD

Cette fenêtre indique la liste des smileys :



Figure 4.30: Liste de smileys offerte par Spark

La figure 4.31 affiche les phrases échangées au cours du chat.

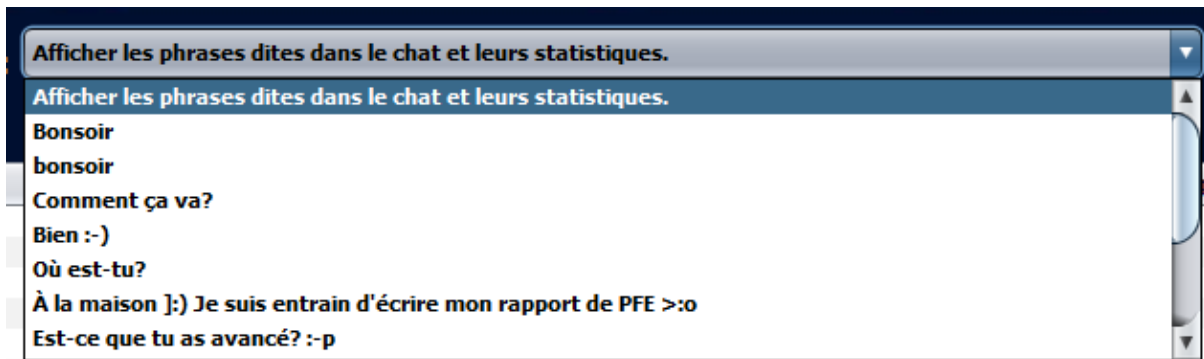


Figure 4.31: Les phrases dans la plateforme

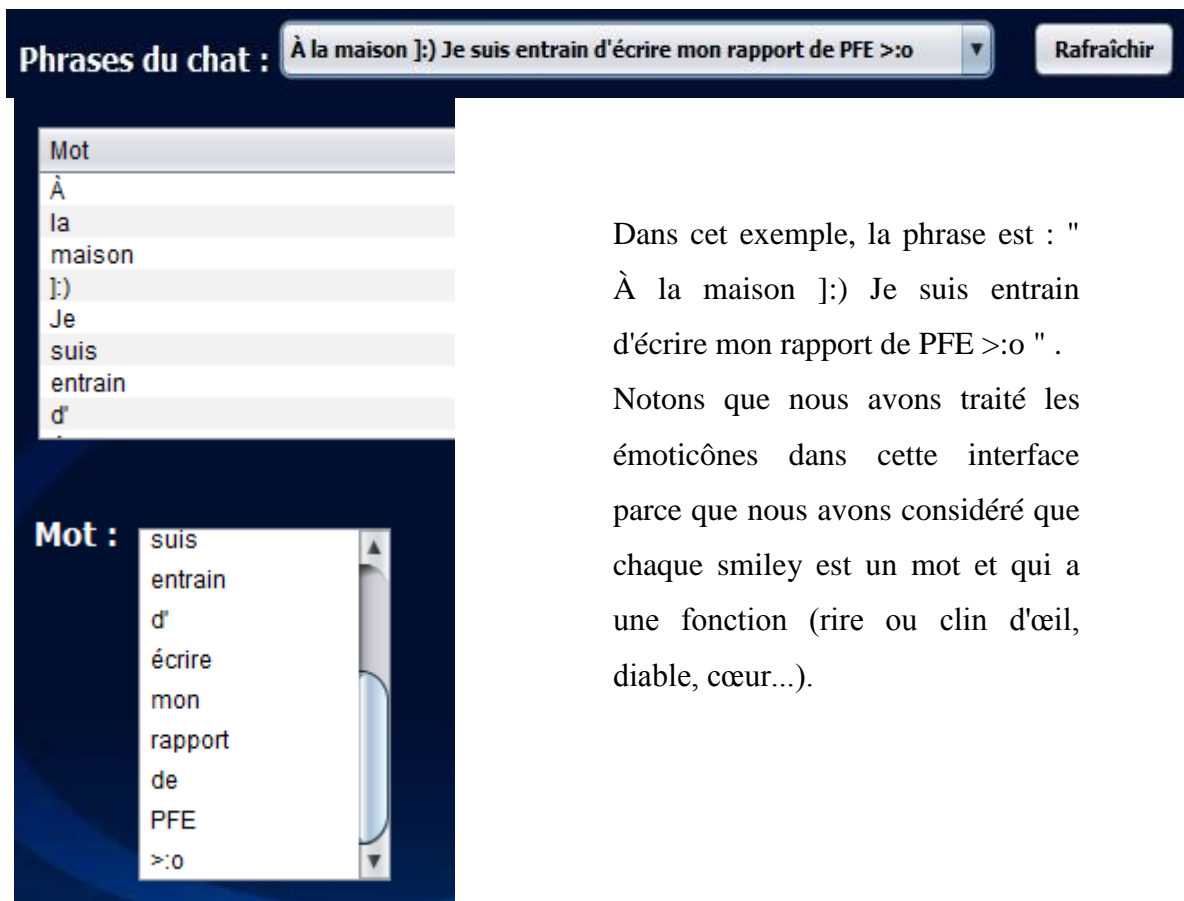


Figure 4.32: Exemple qui traite les smileys

L'étape d'affichage des statistiques se base sur une méthode statistique comme nous l'avons définie précédemment et permet de recueillir et acquérir plus de données (les mots et leurs propriétés ajoutés).

4.3.3 Préparation des données, construction de la base d'apprentissage et choix du RNA

Après la collecte des données (les phrases et les mots) dans le but de constituer une base sur laquelle nous fondons notre système neuronal, nous allons effectuer une analyse de ces données pour déterminer les entrées du réseau de neurones.

Nous devons faire un prétraitement sur ces données, choisir les caractéristiques et les attributs qui les définissent et les convertir en une forme acceptable par le réseau de neurones. Le choix des données et les attributs les caractérisant est très délicat et crucial. En effet, nous avons besoin de transformer les valeurs du monde réel à des formes numériques utilisées par les réseaux de neurones. Dans cette perspective, nous nous sommes basés sur des exemples cités dans ces deux références [23] et [24].

Nous avons eu recours à la représentation par sac de mots (bag-of-words, en anglais) qui est une description d'un document (texte, image,...). Dans une représentation de ce genre, un texte (dans notre cas une phrase) est réduit à une liste d'occurrences de ses mots indépendamment de leur ordre d'apparition dans le texte.

Nous avons choisi notre échantillon de 22 phrases de la BD sur lesquelles nous allons travailler pour définir les entrées du réseau de neurones (voir le tableau 4.2).

Numéro de la phrase	Phrase
1	Hello!
2	Je suis heureux.
3	Je suis malheureux.
4	Je deviendrai ingénieur.
5	Marie joue de la guitare.
6	Elle arrose ses fleurs.
7	La POO est importante pour un développeur.
8	C'est les vacances.
9	L'intelligence artificielle est le nom donné à l'intelligence des machines et des logiciels.
10	La BI est un moyen d'aide à la décision pour une entreprise.
11	Elle participe à la réunion.
12	Il est midi.
13	Je vais à l'école.
14	Les Big data s'appellent aussi les données massives.
15	Elle mange des oranges.
16	Il est malade.
17	J'aime les gâteaux.
18	Quelle heure est-il?
19	Il fait beau.
20	L'été est chaud.
21	C'est lundi.
22	Les réseaux de neurones artificiels sont construits sur un paradigme biologique.

Tableau 4.2: Base d'apprentissage pour le réseau de neurones

Cet échantillon constitue la base d'apprentissage servant à l'apprentissage du réseau de neurones. Cette base se compose de 80 mots distincts c'est pourquoi nous définissons ci-dessous le dictionnaire qui contient ces mots. Nous avons besoin aussi de leurs taux d'apprentissage ultérieurement car les valeurs de taux d'apprentissage constituent les sorties du réseau de neurones. Les figures 4.33, 4.34, 4.35 sont des captures d'écran de la table dictionnaire que nous avons dans la BD:

id_mot_dic	mot	taux_apprentissage	Fonction
1	Hello	0.9	locution_phrase
2	!	0.08727	ponctuation_pt_exclamation
3	Je	0.18	sujet
4	suis	0.125	verbe
5	heureux	0.73	adjectif_s_p
6	malheureux	0.73	adjectif_s_p
7	.	0.03636	ponctuation_pt
8	deviendrai	0.1	verbe
9	ingénieur	0.73	nom
10	Marie	0.3	sujet
11	joue	0.2	verbe
12	de	0.04	préposition
13	la	0.02833	déterminant
14	guitare	0.4	nom
15	Elle	0.16666	sujet
16	arrose	0.3	verbe
17	ses	0.05	déterminant_p
18	fleurs	0.5	nom_p
19	POO	0.35	nom
20	est	0.105	verbe
21	importante	0.2	adjectif
22	pour	0.055	préposition
23	un	0.03333	déterminant
24	développeur	0.2	nom
25	C'	0.15	sujet
26	les	0.046	déterminant_p
27	vacances	0.5	nom_p
28	L'	0.025	déterminant
29	intelligence	0.19	nom
30	artificielle	0.2	adjectif

Figure 4.33: Notre dictionnaire (partie 1)

id_mot_dic	mot	taux_apprentissage	Fonction
31	le	0.01	déterminant
32	nom	0.01	nom
33	donné	0.02	adjectif
34	à	0.035	préposition
35	des	0.02333	déterminant_p
36	machines	0.15	nom_p
37	et	0.01	conjonction de coordination
38	logiciels	0.15	nom_p
39	BI	0.2	nom
40	moyen	0.15	nom
41	d'	0.01	préposition
42	aide	0.15	nom
44	décision	0.15	nom
46	une	0.02	déterminant
47	entreprise	0.15	nom
48	participe	0.4	verbe
49	réunion	0.4	nom
50	Il	0.125	sujet
51	midi	0.8	nom
52	vais	0.3	verbe
53	école	0.3	nom
54	Big	0.2	adjectif_angl
55	Data	0.2	nom_angl
56	s'	0.0438	pronom_réflechi
57	appellent	0.17523	verbe_p
58	aussi	0.04	adverbe
59	données	0.2	nom_p
60	massives	0.2	adjectif_p
61	mange	0.3	verbe
62	oranges	0.3	nom_p

Figure 4.34: Notre dictionnaire (partie 2)

Comme la couche d'entrée, le nombre de neurones de la couche de sortie du réseau de neurones est égal à 80 puisque chaque mot a son propre taux d'apprentissage. De même, le vecteur de la couche de sortie correspondant à la dernière phrase de notre base d'apprentissage "Les réseaux de neurones artificiels sont construits sur un paradigme biologique." est: [0,0,0,0,0,0, 0.03636 , 0,0,0,0, 0.04 ,0,0,0,0,0,0,0,0,0,0, 0.03333 ,0,0, 0.046 ,0,0,0,0,0,0,0,0 ,0, 0.1, 0.15 , 0.15 , 0.1, 0.1 , 0.04 , 0.1 , 0.1].

4.3.4 Création du RNA et apprentissage

Après le choix des représentations de la couche d'entrée et de sortie du réseau de neurones, nous passons à l'étape de l'apprentissage du réseau. Pour créer le réseau de neurones, nous utilisons Neuroph Studio. Le type du réseau de neurones qui va être utilisé pendant nos expériences est le Perceptron Multicouche avec l'algorithme d'apprentissage : la règle de rétro-propagation du gradient. Pour former un réseau de neurones, il faut:

1. Préparer les données,
2. Créer un projet Neuroph,
3. Créer le réseau de neurones,
4. Créer l'ensemble de données pour l'apprentissage,
5. Faire l'apprentissage du réseau,
6. Tester le réseau pour être sûr qu'il a été bien formé.

Après la création du projet dans Neuroph Studio, pour créer le réseau de neurones, nous entrons le nom du réseau et nous choisissons son type, dans notre cas c'est le perceptron multicouche.

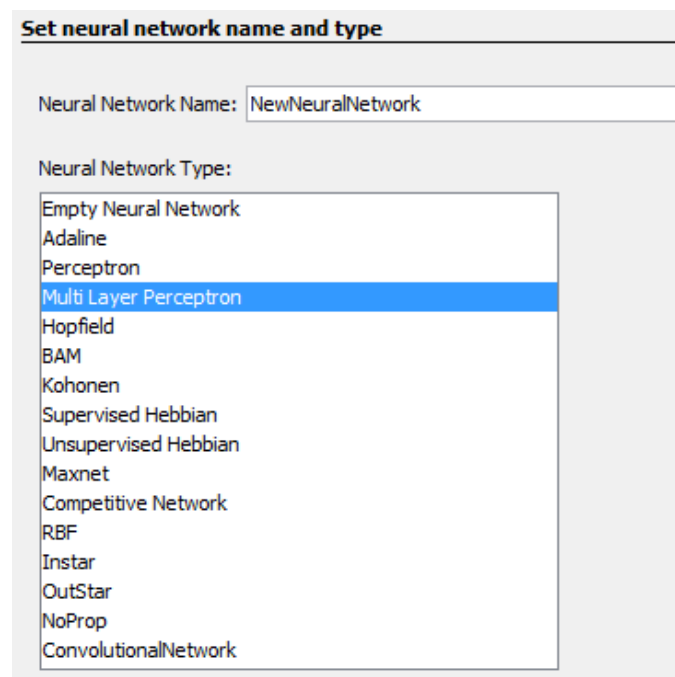


Figure 4.36: Création du réseau de neurones (1): choix de son nom et son type

Dans la fenêtre de dialogue suivante, nous entrons le nombre de neurones : 80 en entrée et en sortie. La question qui se pose à présent est : quel est le nombre de couches cachées et dans chaque couche combien de neurones nous devrions nous avoir? Notons qu'il n'y a pas de règle pour choisir le nombre de couches cachées et le nombre de ses neurones. Pour simplifier, nous décidons d'avoir une seule couche cachée. Mais le nombre de neurones de cette couche doit être optimal car trop de neurones cachés aboutissent à des données complexes tandis que quelques-uns donnent lieu à des réseaux assez longs en apprentissage avec un grand nombre d'itérations. Nous allons jouer sur ce nombre et nous allons essayer des réseaux de neurones avec différents nombres de neurones cachés. Ici par exemple, nous prenons 40 neurones cachés pour ce réseau.

Nous cochons aussi l'option "Utiliser neurone biais" parce que les neurones biais sont ajoutés aux réseaux de neurones pour les aider à apprendre. Nous choisissons la fonction de transfert "Sigmoid" pour tous les tests. Pour la règle d'apprentissage nous choisissons dans ce réseau "rétro-propagation avec momentum". Le momentum est ajouté pour accélérer le processus d'apprentissage et améliorer l'efficacité de l'algorithme. La figure 4.37 résume ces choix :

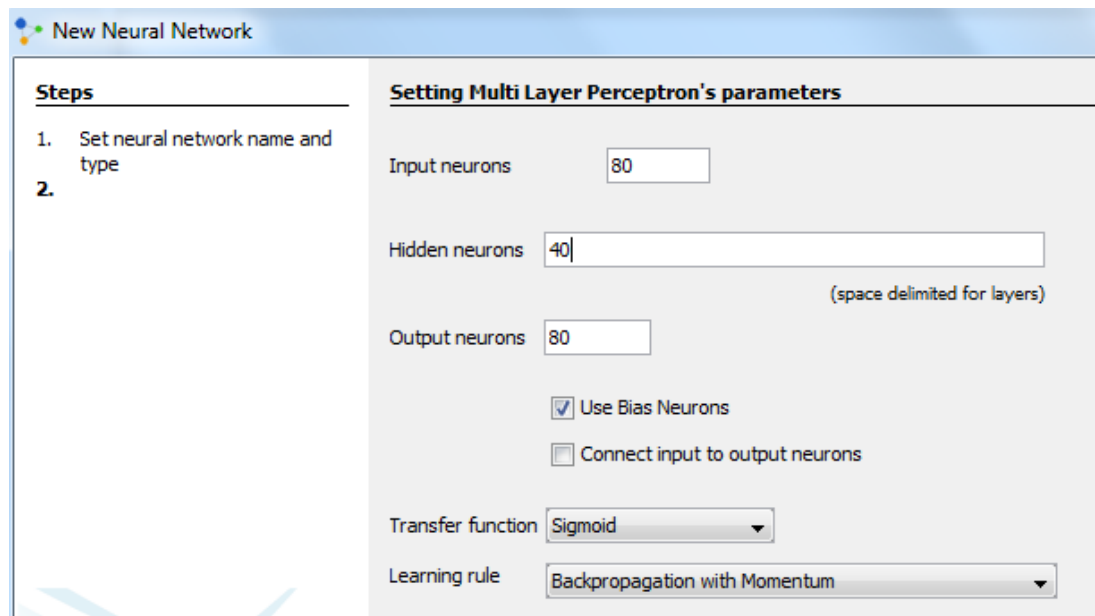


Figure 4.37: Création du réseau de neurones (2): choix du nombre de neurones dans chaque couche, la fonction de transfert et la règle d'apprentissage

Puisque la taille du réseau est très grande, voici une capture d'écran qui le résume. La figure 4.38 montre les trois couches, les entrées, les sorties, leurs numéros, le nombre de connexions qui existent entre la première couche et la deuxième couche, et entre la deuxième couche et la troisième couche et les neurones biais qui sont les neurones en rouge de valeur 1 et qui existent dans la couche d'entrée et la couche cachée.

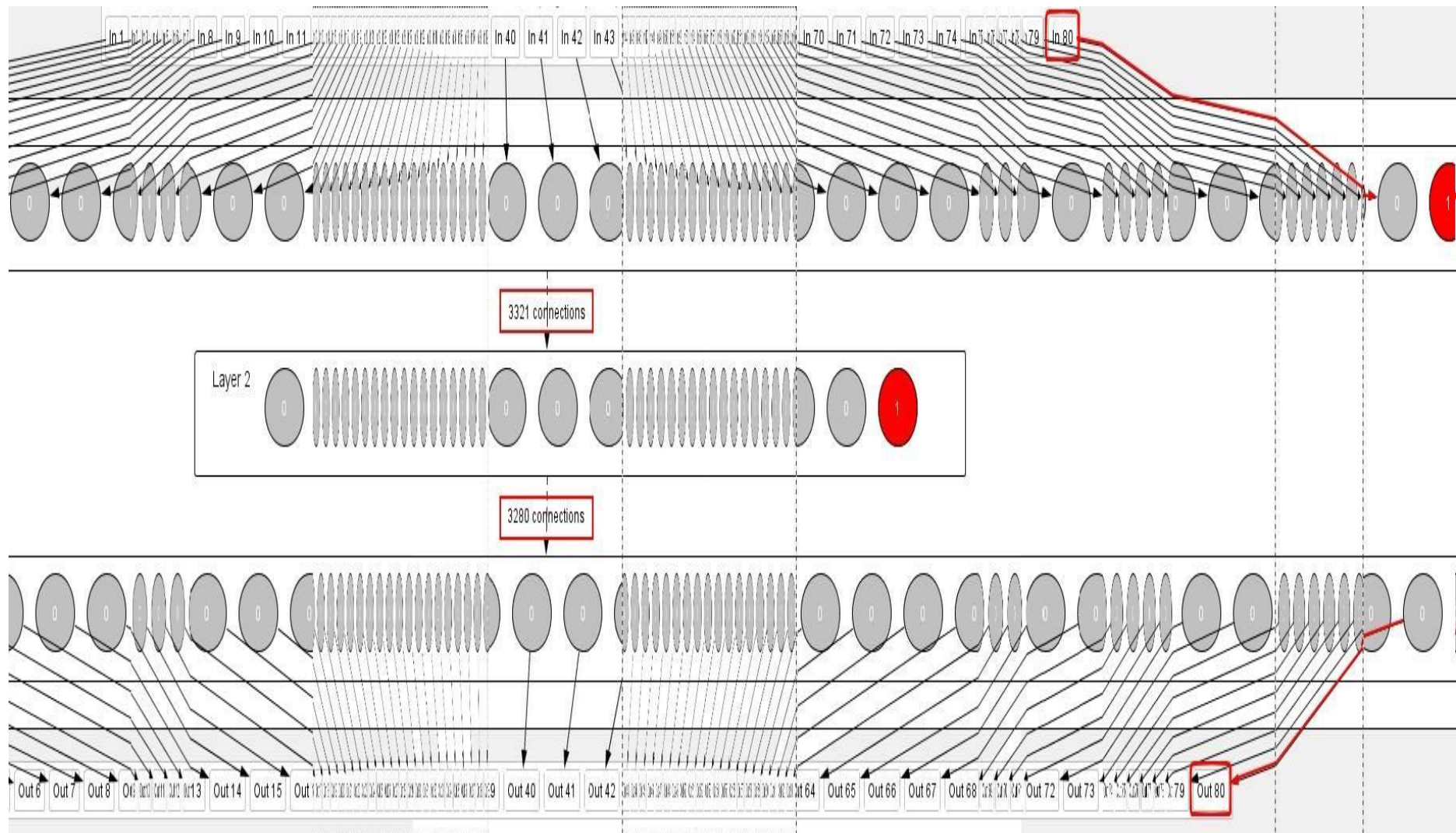


Figure 4.38: Imprime-écran du réseau de neurones

Maintenant, nous arrivons à l'étape de création de l'ensemble de données d'apprentissage. Étant donné que nous avons un nombre assez important d'instances de données, il est plus facile de charger toutes les données directement à partir de certains fichiers que nous avons préparés en avance. Nous devons alors sélectionner le fichier dans lequel nous avons sauvegardé l'ensemble de données normalisé. Les valeurs dans le fichier sont sous la forme d'un tableau mais nous pouvons insérer des valeurs avec comme séparateurs des virgules, des points-virgules ou des espaces vides.

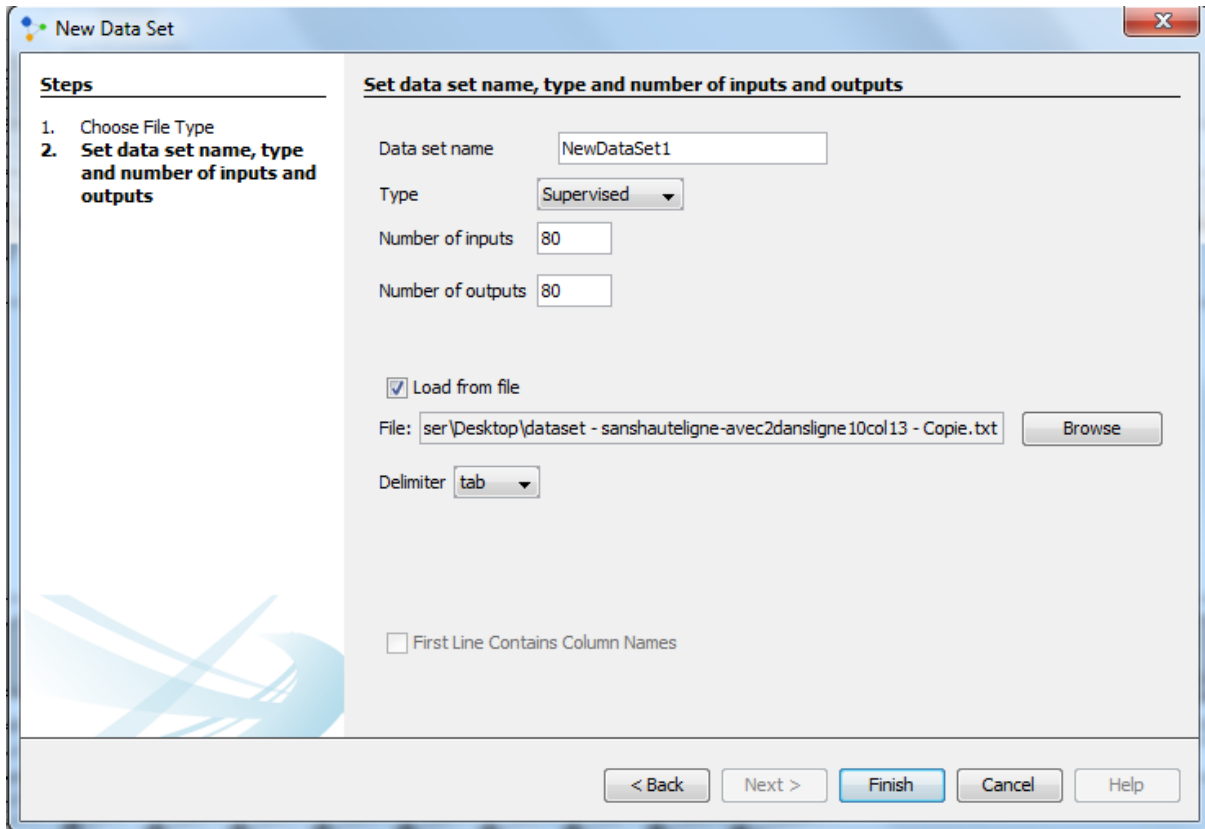
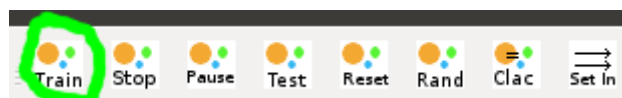


Figure 4.39: Création de l'ensemble d'apprentissage

Il faut choisir aussi le type d'apprentissage. Dans notre cas, il s'agit d'apprentissage supervisé car nous allons imposer au réseau un fonctionnement donné en forçant à partir des entrées qui lui sont présentées, à prendre des valeurs bien précises de ses sorties. Le résultat de cette étape est que toutes les données sont chargées dans la table.

Pour faire l'apprentissage, nous devons glisser et déposer (drag and drop) notre ensemble d'apprentissage dans le champ correspondant dans la fenêtre du réseau et nous cliquons sur le bouton "train" qui deviendra activé dans la barre d'outils.



Une fenêtre de dialogue s'ouvre pour paramétrer l'apprentissage:

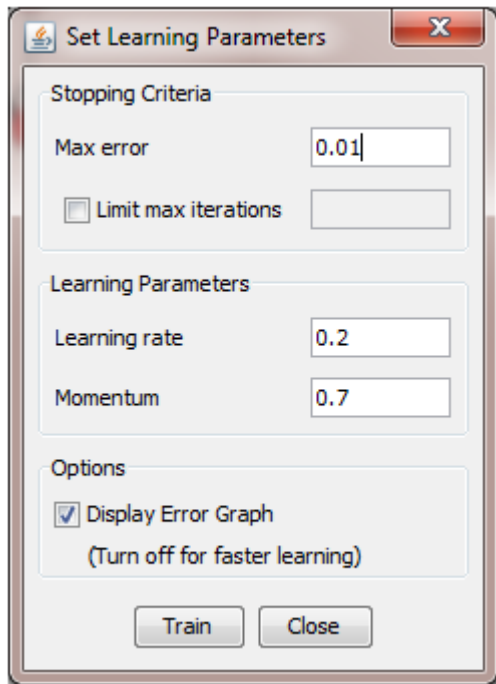


Figure 4.40: Paramètres d'apprentissage

L'erreur maximale ou "max error" est une valeur qui a une limite entre 0.01 et 0.05. Lorsque la valeur nette d'erreur totale descend au-dessous de l'erreur maximale, l'apprentissage est terminé. Plus la valeur de l'erreur est petite, plus nous obtenons une meilleure approximation. Le taux d'apprentissage est une constante dans l'algorithme d'un réseau de neurones qui affecte la vitesse d'apprentissage. Plus le taux est grand, plus le réseau apprend vite. Sa limite est entre 0.01 et 0.9. Le momentum permet au réseau de sauter à travers les minimas locaux. Sa limite est aussi entre 0.01 et 0.9 [25].

Nous choisissons le taux d'apprentissage égal à 0.2, l'erreur maximale égale à 0.01 et le momentum égal à 0.7. Après l'apprentissage, nous obtenons la figure ci-dessous qui correspond au graphe de l'erreur totale du réseau de neurones. Nous constatons que le réseau a convergé puisque la courbe a descendu jusqu'à 0.00 en faisant presque 48 itérations et elle est constante sur cette valeur dès la trentième itération.

Total Network Error Graph

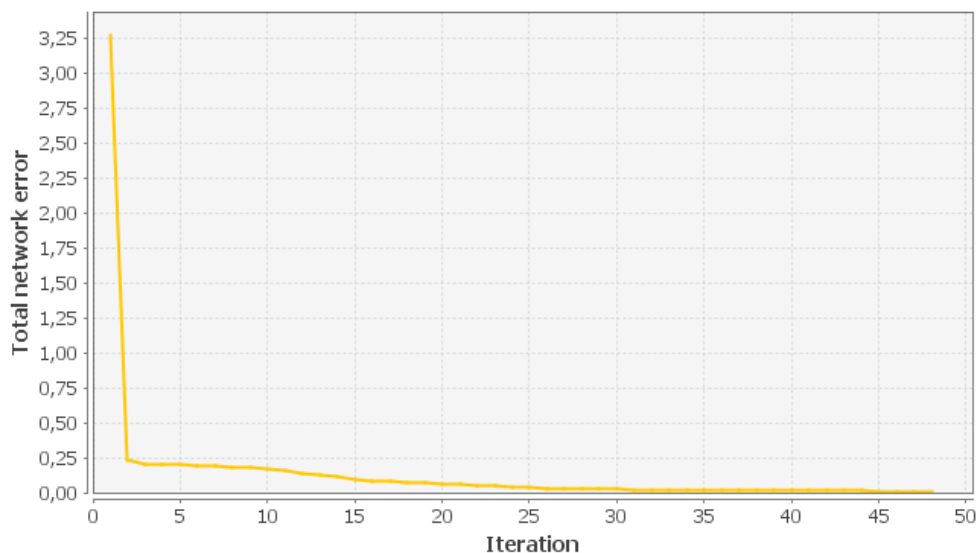


Figure 4.41: Graphe d'erreur totale de notre réseau

Nous pouvons récupérer l'erreur quadratique moyenne totale après l'apprentissage. La figure 4.42 montre le résultat obtenu :

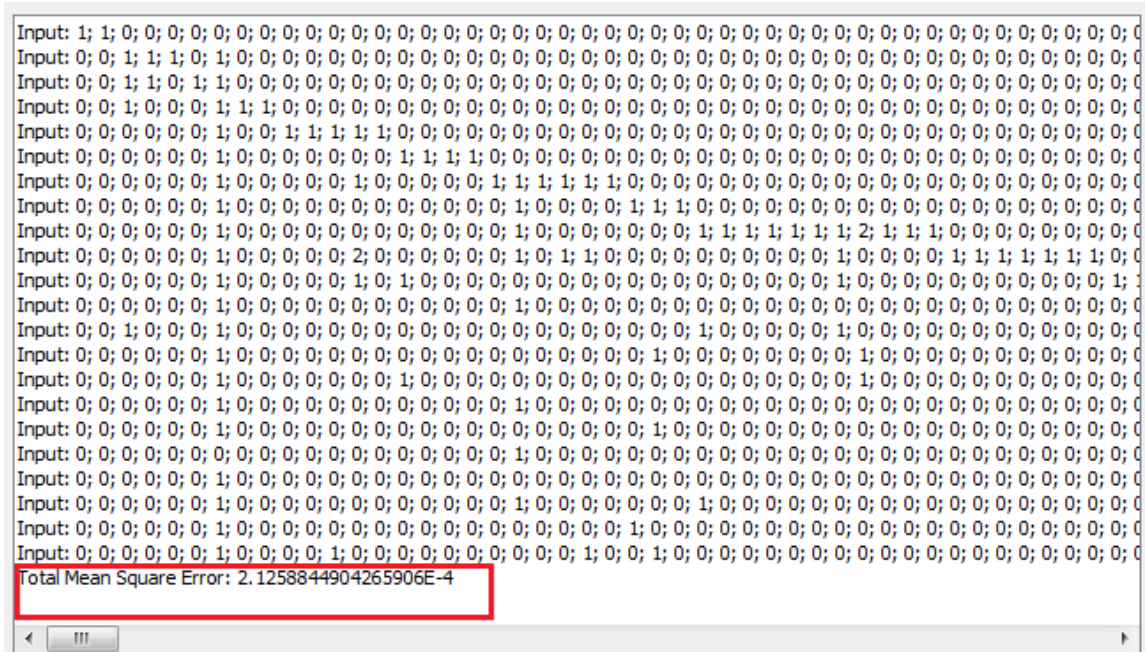


Figure 4.42: L'erreur quadratique moyenne totale du réseau après apprentissage

Nous observons dans cette capture d'écran les entrées (pour les 22 phrases) du réseau. L'erreur quadratique moyenne totale est égale à $2.1258 \cdot 10^{-4}$ qui est très proche de zéro. Une seule couche cachée s'avère ainsi suffisante.

4.3.5 Tests et interprétation

Nous avons réalisé plusieurs tests et apprentissages avec différentes valeurs des paramètres avec la même fonction d'activation sigmoïde. Nous résumons ce que nous avons retiré pour quelques-uns dans le tableau en regard :

Numéro du réseau	Nombre de neurones	Taux d'apprentissage	Momentum	Nombre d'itérations	Erreur maximale	Erreur quadratique moyenne totale
1	20	0.5	0.7	160	0.01	$2.1968 \cdot 10^{-4}$
2	40	0.5	0.7	55	0.04	$6.9744 \cdot 10^{-4}$
3	40	0.5	0.7	80	0.01	$1.9686 \cdot 10^{-4}$
4	40	0.5	0	85	0.01	$1.8256 \cdot 10^{-4}$
5	40	0.8	0.7	50	0.01	$1.9134 \cdot 10^{-4}$
6	40	0.2	0.7	48	0.01	$2.1258 \cdot 10^{-4}$
7	40	0.2	0	185	0.01	$2.2530 \cdot 10^{-4}$
8	80	0.5	0.7	43	0.01	$1.8271 \cdot 10^{-4}$
9	100	0.5	0.7	35	0.01	$1.7676 \cdot 10^{-4}$
10	100	0.5	0	37	0.01	$1.7708 \cdot 10^{-4}$

Tableau 4.3: Résultats de différents apprentissages de quelques réseaux de neurones

D'après ce tableau, nous interprétons que pour le même taux d'apprentissage, le même momentum et la même erreur maximale (0.5, 0.7 et 0.01), en augmentant le nombre de neurones dans la couche cachée, le nombre d'itérations diminue et l'erreur quadratique moyenne totale diminue aussi.

En outre, nous concluons que pour le même réseau (c.à.d. avec le même nombre de neurones cachés, dans ces tests, il est égal à 40) et pour les mêmes valeurs du taux d'apprentissage et de momentum (0.5 et 0.7) mais pour différentes erreurs maximales : le nombre d'itérations est plus petit pour l'erreur maximale la plus grande. En effet, nous avons 55 itérations pour le deuxième réseau de neurones avec une erreur maximale égale à 0.04 alors que pour le troisième réseau qui possède une erreur maximale 0.01, nous avons 80 itérations.

D'un autre côté, pour les mêmes valeurs des paramètres sauf l'erreur maximale, le réseau qui a la plus grande erreur maximale a l'erreur quadratique moyenne totale la plus grande. Quand nous comparons le 2^{ème} le 3^{ème} réseau, l'erreur maximale du deuxième est égale à 0.04 tandis que celle du 3^{ème} réseau est égale à 0.01. L'erreur quadratique moyenne du 2^{ème} réseau est $6.9744 \cdot 10^{-4}$ alors que celle du 3^{ème} réseau est $1.9686 \cdot 10^{-4}$. Mais les deux valeurs de ces erreurs sont moins que leurs erreurs maximales, donc les deux apprentissages ont réussi.

Pour les mêmes valeurs des paramètres sauf celui du momentum et pour le même nombre de neurones cachés, le réseau qui possède lors de son apprentissage un momentum non nul, a moins d'itérations et donc plus rapide que celui qui dispose d'un momentum nul; c'est le cas du 3^{ème} et 4^{ème} réseau. L'apprentissage du 3^{ème} réseau se termine en 80 itérations par contre l'apprentissage du 4^{ème} réseau se termine en 85 itérations. De même, l'apprentissage du 6^{ème} réseau se termine en 48 itérations mais celui du septième réseau se termine en 185 itérations et donc ce dernier est plus lent. En revanche, les erreurs quadratiques moyennes totales sont presque proches.

4.3.6 Construction de la base de test

Après l'étape d'apprentissage du réseau, il nous reste qu'à faire la validation qui représente la dernière étape de développement d'un réseau de neurones. Comme mentionné auparavant, la validation consiste à tester le réseau de neurones avec une base de données différente de celle d'apprentissage et que l'on appelle base de validation ou de généralisation afin d'apprécier les performances du système neuronal. Si le réseau est performant, il doit se comporter bien avec cette base sinon il faut modifier soit l'architecture du réseau soit les représentations des données.

Dans notre cas, la base de test doit être construite seulement à partir des mots composant le dictionnaire. Ainsi, nous avons quelques restrictions et limites pour composer de nouvelles

phrases syntaxiquement correctes. Le tableau suivant présente les phrases qui existent dans la base de généralisation.

Numéro de la phrase	Phrase
1	Marie aime les oranges.
2	Il est un développeur.
3	Je suis malade.
4	L'entreprise fait des réseaux.
5	Elle participe à la réunion et aide à la décision.
6	L'intelligence artificielle BI Big Data données massives et les logiciels sont importante.
7	Je suis un beau ingénieur développeur joue de la guitare aime les fleurs réseaux vacances et l'été.

Tableau 4.4: Base de test pour le réseau de neurones

Nous avons essayé de construire des phrases longues pour tester le réseau même si nous avons été limités par les mots du dictionnaire. Par exemple, dans la dernière phrase, nous remarquons qu'il nous manque la virgule pour séparer quelques mots. Parfois, nous ne pouvons pas faire l'accord avec le sujet en nombre ou en genre car nous possédons que ces mots à savoir l'adjectif "importante" dans la 6^{ème} phrase qui normalement doit être au pluriel.

Nous avons dédié une interface pour saisir ces phrases de la base de généralisation. Au cas où l'utilisateur ne veut pas saisir des phrases sous prétexte qu'il ne veut pas chercher dans le dictionnaire pour les composer, notre système lui facilite la tâche en générant automatiquement des phrases simples à partir du dictionnaire.

Dans cette interface, il y a un champ où l'utilisateur peut saisir ses phrases, un bouton pour lui proposer des phrases et un autre bouton pour afficher dans une table les mots qui composent ces phrases, le nombre d'apparitions de chaque mot dans sa phrase et leurs taux d'apprentissage. Le bouton "Affichage et ajout" permet aussi d'ajouter automatiquement ces phrases à la base de données sous une forme bien déterminée afin de les utiliser dans le test du réseau de neurones (voir la figure ci-dessous).

Mots	Nombre d'apparition de ce mot dans la phrase	Taux d'apprentissage
Marie	1	30
aime	1	15
les	1	4.6
oranges	1	30
.	1	3.75

Figure 4.43: Interface pour tester le réseau de neurones

Ces phrases sont ajoutées dans deux tables de la BD. L'une pour les entrées du réseau et qui s'appelle "inputs outputs" et l'autre pour ses sorties et qui porte le nom de "data outputs". Jetons un clin d'œil sur ces deux tables grâce aux deux figures 4.46 et 4.47 :

Id_phrase_test	Hello	pt_exclamation	Je	suis	heureux	malheureux	point
1	0	0	0	0	0	0	1
2	0	0	0	0	0	0	1
3	0	0	1	1	0	0	1
4	0	0	0	0	0	0	1
5	0	0	0	0	0	0	1
6	0	0	0	0	0	0	1
7	0	0	1	1	0	0	0

deviendrai	ingénieur	Marie	joue	de	la	guitare	Elle	arrose	ses	fle
0	0	1	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	2	0	1	0	0	
0	0	0	0	0	0	0	0	0	0	
0	1	0	1	1	1	1	0	1	0	

Figure 4.44: Table des entrées de la base de test

Pour les sept phrases de la base de généralisation, si le mot existe dans la phrase, nous mettons son nombre d'apparitions dans la case correspondante. Le nombre de colonnes de cette table est bien sûr égal à 80 : le nombre de mots que contient le dictionnaire. D'où, cette capture d'écran est un extrait de la table. Pour la table des sorties, nous mettons le taux d'apprentissage des mots composant la phrase à étudier.

id_output	Hello	pt_exclamation	Je	suis	heureux	malheureux	point
1	0	0	0	0	0	0	0.03636
2	0	0	0	0	0	0	0.03636
3	0	0	0.18	0.125	0	0	0.03636
4	0	0	0	0	0	0	0.03636
5	0	0	0	0	0	0	0.03636
6	0	0	0	0	0	0	0.03636
7	0	0	0.18	0.125	0	0	0.03636

deviendrai	ingénieur	Marie	joue	de	la	guitare	Elle	arrose
0	0	0.3	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0.02833	0	0.16666	
0	0	0	0	0	0	0	0	
0	0.73	0	0.2	0.04	0.02833	0.4	0	0.03636

Figure 4.45: Table des sorties de la base de test

Phrase 1	Mots	Marie	oranges	aime	les	.				
	Sortie trouvée (ST)	0.007	0.014	0.011	0.007	0.035				
	Sortie désirée (SD)	0.3	0.3	0.15	0.046	0.036				
	Différence entre les deux sorties (SD-ST)	0.293	0.286	0.139	0.039	0.001				
Phrase 2	Mots	développeur	Il	un	.	est				
	ST	0.012	0.037	0.03	0.05	0.132				
	SD	0.2	0.125	0.03	0.036	0.105				
	SD - ST	0.188	0.088	0	-0.014	-0.027				
Phrase 3	Mots	malade	Je	.	suis					
	ST	0.108	0.107	0.037	0.207					
	SD	0.75	0.18	0.036	0.125					
	SD - ST	0.642	0.073	-0.001	-0.082					
Phrase 4	Mots	fait	entreprise	réseaux	.	des	L'			
	ST	0.024	0.04	0.04	0.02	0.01	0.02			
	SD	0.15	0.15	0.1	0.036	0.023	0.025			
	SD-ST	0.126	0.11	0.06	0.016	0.013	0.005			
Phrase 5	Mots	aide	décision	Elle	à	participe	et	.	la	réunion
	ST	0.006	0.019	0.091	0.001	0.379	0.01	0.052	0.027	0.555
	SD	0.15	0.15	0.166	0.035	0.4	0.01	0.036	0.028	0.4
	SD-ST	0.144	0.131	0.075	0.034	0.021	0	-0.016	0.001	-0.155

Phrase 6	Mots	artificielle	massives	BI	données	Big	Data	importante	intelligence	logiciels	sont	.	et
	ST	0.002	0.003	0.017	0.023	0.063	0.028	0.028	0.026	0.01	0.027	0.036	0.01
	SD	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.19	0.15	0.1	0.036	0.01
	SD-ST	0.198	0.197	0.183	0.177	0.173	0.172	0.172	0.164	0.14	0.073	0	0
	Mots	les	L'										
	ST	0.048	0.05										
	SD	0.046	0.025										
	SD-ST	-0.002	-0.025										
Phrase 7	Mots	ingénieur	beau	vacances	fleurs	été	guitare	la	développeur	Je	aime	joue	
	ST	0.04	0.018	0.004	0.019	0.01	0.036	0.006	0.004	0.005	0.014	0.075	
	SD	0.73	0.65	0.5	0.5	0.4	0.4	0.283	0.2	0.18	0.15	0.2	
	SD-ST	0.69	0.632	0.496	0.481	0.39	0.364	0.277	0.196	0.175	0.136	0.125	
	Mots	réseaux	de	un	et	L'	les	arrose					
	ST	0.007	0.003	0.007	0.01	0.02	0.181	0.52					
	SD	0.1	0.04	0.033	0.01	0.025	0.046	0.3					
	SD-ST	0.093	0.037	0.026	0	0.005	- 0.135	- 0.22					

Tableau 4.5: Comparaison entre les sorties trouvées et les sorties désirées pour le RN avec 100 neurones cachés, LR= 0.5 et momentum=0.7

Nous illustrons à présent les résultats dans le tableau 4.5. Pour chaque phrase, nous listons ses mots, les sorties trouvées lors du test avec le réseau de neurones considéré. Elles sont notées par ST, les sorties souhaitées ou désirées c.à.d. que nous voulons que le réseau nous retourne et ce sont celles que nous avons imposées dans l'étape précédente. Ces sorties sont désignées par SD. L'écart (ou la différence) entre les deux sorties souhaitées et trouvées est noté par SD-ST, son importance est de comparer si ces sorties sont proches ou non. Pour cette raison, nous avons trié cette différence par ordre décroissant.

4.3.7 Interprétation des résultats

D'après ce tableau, il y a des différences entre les sorties désirées et trouvées qui sont très proches de zéro et même nulles parfois. Par exemple, dans la 6^{ème} phrase, nous avons les mots "et" et le point "." qui ont un écart nul. Nous constatons aussi que les mots les mieux appris dans ce réseaux sont les mots de liaison à savoir le mot "et" qui a toujours une différence nulle, les déterminants comme "les", "un", "la", "L' ", et le point ".". Le fait que ces mots se répètent beaucoup dans les phrases, ils sont donc les mieux appris. Ce réseau de neurones avec 100 neurones cachés, décrit par ce tableau conduit à plusieurs différences presque nulles, ce qui est l'idéal.

Pour évaluer la performance de ce réseau de neurones, nous avons calculé un taux moyen d'erreur, qui indique dans quelle mesure les prévisions du réseau sont proches des valeurs cibles de cette façon: $\frac{\sum_{i=1}^{\text{nombre de mots dans la base de test}} (S_D - S_T)^2}{\text{nombre de mots dans la base de test}}$. Nous l'avons trouvé égale à 0.0473 qui est une valeur relativement petite et prouve la performance du réseau proposé.

Nous déduisons ainsi que le choix du nombre de neurones cachés est très important pour l'efficacité d'un réseau de neurones. Sans oublier que la performance des réseaux est très sensible aux paramètres choisis dans l'étape d'apprentissage.

Conclusion

À travers ce chapitre, nous avons décrit l'environnement de développement logiciel et matériel qui nous a servi de moyen incontournable pour la mise au point de ce projet. Ensuite, nous avons justifié nos choix techniques. Puis, nous avons expérimenté notre application et interprété les résultats obtenus. Cette expérimentation nous a permis de tirer quelques conclusions quant aux caractéristiques des réseaux de neurones et de valider l'approche de résolution que nous avons proposée.

Conclusion générale

Ce projet, effectué au sein de l'entreprise *Cube 3D Technology*, s'inscrit dans le cadre du projet de fin d'études d'ingénieur en informatique à l' *École Nationale Supérieure d'Ingénieurs de Tunis*. À travers ce rapport, nous avons procuré le bilan complet de notre travail qui consiste à réaliser une plateforme d'apprentissage linguistique de textes français par réseaux de neurones.

Nous avons entamé ce rapport par une étude du contexte général du sujet, ce qui nous a permis de dégager la problématique ayant engendré le besoin d'une telle application. Puis, nous avons effectué une étude théorique des concepts de base liés à notre projet à savoir l'apprentissage, le traitement automatique du langage naturel et les réseaux de neurones. Nous avons aussi cité et traité quelques exemples qui s'appuient sur ces notions.

Ensuite, nous avons dévoilé notre approche proposée en identifiant en premier lieu les besoins fonctionnels et non fonctionnels auxquels devra répondre notre application. En second lieu, nous avons présenté une conception générale et une conception détaillée de notre plateforme. Par la suite, nous avons décrit la procédure de développement du réseau de neurones que nous avons proposé en réponse à la problématique posée.

Enfin, nous avons montré les choix techniques et nous avons exposé le travail effectué à travers des captures d'écran. Nous avons détaillé toutes les étapes nécessaires pour faire l'apprentissage des réseaux de neurones en expliquant l'expérimentation effectuée et les résultats des tests obtenus pour valider notre solution proposée.

Tout au long de ce projet, une étude bibliographique exhaustive était menée pour clarifier les concepts entrant en jeu dans notre approche proposée. En effet, nous avons justifié notre démarche par différentes recherches réalisées.

Au terme de ce projet de fin d'études, nous estimons que nous avons pu atteindre les objectifs que nous avons fixés. En fait, ce travail fut d'un apport considérable, vu qu'il nous a permis de nous familiariser avec les concepts des réseaux de neurones et de découvrir et manipuler différents environnements.

Cependant, nous avons rencontré plusieurs difficultés durant la phase d'étude des notions associées aux réseaux de neurones qui nous a pris beaucoup de temps afin de s'adapter avec ses concepts, dans la concrétisation de ces réseaux associée à notre problème et le choix du

chat lié à notre plateforme le plus adéquat pour développer notre solution ainsi que les paramètres spécifiques de ces réseaux.

Ce travail pourrait certainement être amélioré par faire un apprentissage sur une base de données plus large. Nous envisageons aussi de faire des tests sur d'autres types des réseaux de neurones qui pourraient être plus performants. En plus, il est possible d'analyser syntaxiquement les mots pour pouvoir prédire les fonctions des mots suivants. Cela va servir à l'analyse sémantique permettant une compréhension intelligente par un avatar virtuel 3D capable d'interagir avec un interlocuteur afin d'effectuer plusieurs tâches. Et nous pouvons essayer de compléter le développement de cet assistant virtuel.

Pour conclure, l'apport de ce travail a été d'une importance énorme, dans la mesure où il a été une occasion pour enrichir largement nos connaissances et a confirmé nos futures ambitions d'exercer dans le domaine de la recherche en informatique.

Il a été aussi une occasion pour s'intégrer dans la vie professionnelle dans un cadre de respect mutuel et de collaboration pour atteindre les objectifs fixés.

Bibliographie

- [B1] Roques Pascal, UML 2, par la pratique, Paris : Edition EYROLLES, pp. 309, 2004.
- [B2] S. Tolar, “ Techniques d'apprentissage automatique” *Apprentissage automatique et réduction du nombre de dimensions*, pp. 1, 2008.
- [B3] F. Tshirhart, “Réseaux de neurones formels appliqués à l’Intelligence Artificielle et au jeu”, ACADEMIA, mémoire de recherche, pp. 21,22, 2009.
- [B4] M. Benjamin, S. Giner, M. Gilles, A. Bouragba, “Rapport des neurones pour le Go”, pp. 11.
- [B5] “TAL: Traitement Automatique des Langues”, pp. 27-35, 2012.
- [B6] E. Hervet, “Réseaux de neurones”, pp. 1-2, 2014.
- [B7] C. Touzet, “Les réseaux de neurones artificiels introduction au connexionnisme”, pp. 8, 1992.
- [B8] P. Borne, M. Benrejeb, J. Haggège, “Les réseaux de neurones Présentation et applications”, pp. 1.
- [B9] “Techniques neuronales adaptées aux données spatio-temporelles”, pp. 31.
- [B10] W. Al Falou, “DEA Modélisation et ingénierie du logiciel scientifique : Reconnaissance de caractères manuscrits par réseau de neurones”, pp. 17-19, 1998.
- [B11] L. Chergui, “Combinaison de classifieurs pour la reconnaissance de mots arabes manuscrits”, pp. 57-58, 2013.
- [B12] E.M. Brakni, “Réseaux de neurones artificiels appliqués à la méthode électromagnétique transitoire InfiniTEM”, pp. 11, 2011.
- [B13] T. Cour, G. Giraud, A. Kods, T. Luong, R. Lauransson, C. Marcovici, K. Sadeghi, “Reconnaissance de formes par réseau de neurones”, pp. 47, 2002.

Nétographie

- [1] Y. LeCun, “Les enjeux de la recherche en intelligence artificielle” disponible: https://interstices.info/jcms/p_89081/les-enjeux-de-la-recherche-en-intelligence-artificielle [consulté le 20/07/2016].
- [2] T. Lucas, “ Learning, big data, réseaux de neurones... pourquoi l’intelligence artificielle revient- elle maintenant?” disponible: <http://www.usine-digitale.fr/editorial/deep-learning-big-data-reseaux-de-neurones-pourquoi-l-intelligence-artificielle-revient-elle-maintenant.N327758> [consulté le 20/07/2016].
- [3] www.cube3dtech.com [consulté le 20/07/2016].
- [4] M. Tual, ”Apprentissage : l’intelligence artificielle, une élève de plus en plus douée” disponible: http://www.lemonde.fr/pixels/article/2015/12/22/apprentissage-l-intelligence-artificielle-une-eleve-de-plus-en-plus-douee_4836339_4408996.html [consulté le 21/07/2016].
- [5] P. Lemberger, “Le machine Learning - quand les données remplacent les algorithmes” disponible: <http://www.journaldunet.com/solutions/expert/56923/le---machine-Learning-----quand-les-donnees-remplacent-les-algorithmes.shtml> [consulté le 21/07/2016].
- [6] P. Beraud, “Apprentissage automatique (Machine Learning)” article de Microsoft, disponible: https://blogs.msdn.microsoft.com/big_data_france/2013/04/24/apprentissage-automatique-machine-learning/ [consulté le 21/07/2016].
- [7] C. Rouveirol, “Apprentissage automatique”, cours en ligne, disponible: http://www-lipn.univ-paris13.fr/~rouveirol/enseigne/MICR_AS/Arbre-decision-1-2x2.pdf [consulté le 21/07/2016].
- [8] S. Tollari, “Thèse” disponible: <http://wwwia.lip6.fr/~tollaris/ARTICLES/ THESE/node7.html> [consulté le 21/07/2016].
- [9] Wikipédia, disponible: https://fr.wikipedia.org/wiki/Apprentissage_automatique [consulté le 21/07/2016].
- [10] Psychomédia, disponible: <http://www.psychomedia.qc.ca/lexique/definition/apprentissage-profond> [consulté le 21/07/2016].
- [11] “Cursus de linguistique informatique”, Université Paris, disponible: <http://li.linguist.univ-paris-diderot.fr/> [consulté le 22/07/2016].
- [12] “Linguistique informatique”, Dictionnaire l’internaute, disponible:

<http://www.linternaute.com/dictionnaire/fr/definition/linguistique-informatique/> [consulté le 22/07/2016].

[13] SMG, “Linguistique informatique”, disponible: <http://www.smglanguages.com/fr/smg-fra/smgscitech-fr/linguistique-informatique> [consulté le 22/07/2016].

[14] M. Quintana, “Qu’est-ce que le TAL?”, disponible: <https://www.inbenta.com/fr/blog/entree/qu-est-ce-que-le-tal> [consulté le 22/07/2016].

[15] S. Nissen, “Création d’un réseau de neurones - c’est facile”, Bibliothèque FANN, disponible: http://fann.sourceforge.net/fann_fr.pdf [consulté le 23/07/2016].

[16] W. Arrouy, “ Les réseaux de neurones, La rétro-propagation du gradient”, disponible: <http://william.arrouy.free.fr/neural/neu3.html> [consulté le 27/07/2016].

[17] J.C. Risch, “Les réseaux de neurones”, blog scientifique des outils du Big Data aux algorithmes de Machine Learning, 2015, disponible: <https://jcrisch.wordpress.com/2015/04/02/les-reseaux-de-neurones/> [consulté le 27/07/2016].

[18] Statistica, disponible: <http://www.statsoft.fr/concepts-statistiques/glossaire/t/taux-apprentissage.html> [consulté le 27/07/2016].

[19] L. Guira, “Stock market prediction using Neuroph neural networks”, Technobium, 2015, disponible: <http://technobium.com/stock-market-prediction-using-neuroph-neural-networks/> [consulté le 27/07/2016].

[20] Wikipédia, disponible: https://fr.wikipedia.org/wiki/R%C3%A9tropropagation_du_gradient [consulté le 27/07/2016].

[21] Wikipédia, disponible : <https://fr.wikipedia.org/wiki/NetBeans> [consulté le 02/08/2016].

[22] Wikipédia, disponible : <https://fr.wikipedia.org/wiki/MySQL> [consulté le 02/08/2016].

[23] T. Mikolov, “Using Neural Networks for Modelling and Representing Natural Languages”, Facebook AI Research, 2014, pp11-12 disponible: <http://www.coling-2014.org/COLING%202014%20Tutorial-fix%20-%20Tomas%20Mikolov.pdf> [consulté le 06/08/2016].

[24] J. Heaton, “Non-Mathematical Introduction to Using Neural Networks”, Heaton Research, 2016, disponible: <http://www.heatonresearch.com/content/non-mathematical-introduction-using-neural-networks> [consulté le 06/08/2016].

[25] I. Petković, “Teaching assistant evaluation with Neural Networks”, Université de Belgrade, disponible : <http://neuroph.sourceforge.net/tutorials/TeachingAssistant/TeachingAssistantEvaluation.html> [consulté le 07/08/2016].

Résumé

Dans le cadre du projet de fin d'études, nous avons réalisé une plateforme d'apprentissage linguistique de textes français par réseaux de neurones. Elle permet à l'utilisateur d'afficher des statistiques des mots composant des phrases, saisies ou récupérées à partir d'un chat lié à notre application, dont les plus importants sont leurs taux d'apprentissage. Ces mots forment la base d'apprentissage et servent comme entrées au réseau de neurones et les taux représentent les sorties de ce dernier. Nous avons adopté un apprentissage supervisé et des réseaux de neurones de type Perceptron multicouche.

Mots clés: Apprentissage linguistique, Réseaux de neurones, Mots, Taux d'apprentissage, Apprentissage supervisé, Perceptron multicouches.

Abstract

As part of the end of studies project, we achieved a language learning platform for French texts by neural networks. It allows user to display statistics of words composing sentences entered or retrieved from a chat linked to our application, which the most important are learning rates. These words constitute the learning base and serve as inputs to the neural network and the rates represent its outputs. We adopted a supervised learning and multi-layer perceptron as type of neural networks.

Keywords: language learning, neural networks, words, learning rates, supervised learning, multi-layer perceptron.

خلاصة

في إطار مشروع ختم الدروس ، قمنا بإنجاز برنامج تعلّم لغوي للنصوص الفرنسية بواسطة الشبكات العصبية التي تسمح للمستعملين عرض إحصائيات الكلمات المكوّنة للجملة المدخلة أو المستعيدة من المراسلة الفورية الموصولة بتطبيقنا، أهمّها معدّل تعلّمهم. هذه الكلمات تشكّل قاعدة التعلّم وتمثّل مدخلات شبكاتنا العصبية و المعدّلات تمثّل مخرجاتهم. اخترنا التعلّم المشرف و برسبترون متعدّد الطبقات كنوع للشبكات العصبية.

كلمات المفاتيح: تعلّم لغوي، الشبكات العصبية ، الكلمات، معدّل التعلّم، التعلّم المشرف، برسبترون متعدّد الطبقات.