Université de Blida 1–Saad Dahlab

Faculté des sciences

Département d'Informatique

Mémoire présenté par :

Mlle. BENBLIDIA Maria

Pour l'obtention du diplôme de Master

Domaine : Mathématique et Informatique

Filière : Informatique

Spécialité : Ingénierie des Logiciels

Sujet :

Vers une amélioration du Service Client grâce aux Agents Conversationnels et à l'Analyse des Sentiments

Soutenu le 24 Septembre 2020 devant le jury composé de :

M. A. Cherif Zahar	Université de Blida 1	Président
Mme. C. Hireche	Université de Blida 1	Examinatrice
Mme. M. MEZZI	Université de Blida 1	Promotrice

Remerciements

Après avoir rendu grâce à Dieu le Tout Puissant et le Miséricordieux, nous tenons à remercier vivement tous ceux qui ont de près ou de loin contribué à l'élaboration de cet humble travail.

Au terme de ce travail, nous tenons à témoigner de notre profonde gratitude à notre chère promotrice, Dr. M. Mezzi pour son suivi et pour son énorme soutien qu'elle n'a cessé de nous prodiguer tout au long de cette période de travail.

Nous adressons également nos vifs remerciements aux membres du jury pour avoir bien voulu examiner et évaluer ce modeste travail.

Nos remerciements les plus chaleureux s'adressent à notre chère tante Pr. N. Benblidia pour son aide précieuse, son soutien moral et ses nombreux encouragements.

Nous remercions également notre chère amie et camarade de promotion, Bouchetara Rym qui n'a jamais été avare de conseils et qui a toujours été dévouée pour aider tous ses camarades durant ces cinq années d'études.

Résumé

Les exploits de l'intelligence artificielle ont suscité ces dernières années l'intérêt des chercheurs dans le développement et l'adoption généralisée d'assistants virtuels fondés sur l'intelligence artificielle, favorisant ainsi la conversation instantanée. Ces assistants sont connus sous le nom d'*agents conversationnels*, ou plus communément de *chatbots*, et sont utilisés dans un large éventail de domaines. Ils sont devenus de plus en plus sophistiqués en introduisant de nouvelles fonctionnalités d'apprentissage profond dans l'intention d'enrichir l'expérience utilisateur. L'analyse des sentiments fait partie de ces fonctionnalités ; elle permet à l'agent conversationnel de déceler les émotions derrière le message fourni par l'utilisateur.

Notre thèse est située à l'intersection de deux champs de recherche, à savoir les agents conversationnels et l'analyse des sentiments. Nous suggérons d'implémenter un agent conversationnel orienté données doté d'analyse des sentiments basée sur l'apprentissage profond. Notre agent conversationnel sera appliqué au domaine du service clientèle afin de remédier aux problèmes de disponibilité, de temps de réponse et de perception des émotions des utilisateurs.

À cet effet, nous proposons d'utiliser un modèle séquence à séquence doté d'un mécanisme d'attention pour notre système de dialogue, ainsi qu'un réseau neural récurrent pour la détection du sentiment. Les résultats obtenus après l'évaluation de notre système se sont avérés prometteurs.

Mots-clés : Agents Conversationnels, Analyse des Sentiments, Chatbot, Apprentissage Profond, Service Client.

Abstract

Artificial Intelligence's exploits have aroused these past years researcher's interest in the development and the general adaptation of virtual assistants based on artificial intelligence, favoring instant conversation. These assistants are known as *conversational agents* or more commonly as *chatbots*, and are used in a wide range of fields. They have become more and more sophisticated by introducing new deep learning functions, in the interest of enriching the user's experience. Sentiment Analysis is one of these functions; it enables the conversational agent to detect emotions behind messages provided by the user.

Our thesis is situated at the intersection of two research fields, namely Conversational Agents and Sentiment Analysis. We suggest to implement a data oriented conversational agent endowed with sentimental analysis based on deep learning. Our agent will be applied in the customer service area to resolve availability issues, time responses and the detection of user's emotional perceptions.

In this regard, we propose to use a Sequence-To-Sequence model with an Attention Mechanism for our dialogue system, as well as a Recurrent Neural Network to detect emotions. The results obtained after our system's evaluation have been shown to be promising.

Key words: Conversational Agents, Sentiment Analysis, Chatbots, Deep Learning, Customer Service.

ملخص

جلبت انجازات الذكاء الاصطناعي اهتمام الباحثين في الآونة الأخيرة لتطوير مساعدين افتراضيين مستندين على الذكاء الاصطناعي و معتمدين على نطاق واسع, وبالتالي تحسين المراسلة الفورية. يُعرف هؤلاء المساعدين باسم " Conversational Agents " أو عامة " Chatbots "، ويستخدمون في تشكيلة متنوعة من المجالات. لقد أصبحوا متطورين بشكل متزايد من خلال إدماج خاصيات جديدة للتعلم العميق (Deep Learning) بهدف إغناء تجربة المستخدم. تحليل المشاعر أو "Sentiment Analysis" يعتبر أحد تلك الخاصيات؛ تسمح هذه الخاصية Conversational Agent بالكشف عن المشاعر الكامنة وراء الرسالة التي يقدمها المستخدم.

تقع أطروحتنا عند تقاطع مجالين من مجالات البحث العلمي، على النحو التالي: Conversational Agents و Sentiment Analysis. نقترح هنا تطبيق Conversational Agent يعتمد على البيانات مع Sentiment Analysis بناءً على التعلم العميق. سيتم تنفيذ ال Conversational Agent في مجال خدمة العملاء (Customer Service) من أجل معالجة مشاكل التوافر، وقت الاستجابة وإدراك مشاعر المستخدم.

تحقيقا لهذه الغاية، نقترح استخدام نموذج "Sequence-To-Sequence" مع آلية الانتباه (Attention Mechanism) لنظام الحوار الخاص بنا، بجانب نموذج »Recurrent Neural Networks" من أجل تحديد المشاعر. أظهرت النتائج التي تم الحصول عليها بعد تقييم نظامنا أنها واعدة.

الكلمات المفتاحية: روبوتات المحادثة، تحليل المشاعر، Chatbots، التعلم العميق، خدمة العملاء.

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

Contexte global

Au cours de ces dernières années, les prouesses de l'intelligence artificielle ont captivé l'imagination du public et ont suscité l'adoption universelle d'assistants basés sur l'intelligence artificielle. Cependant, à cette cadence rapide de développement et d'innovation se joint une incertitude croissante quant à l'orientation du développement technologique et à son impact sur la société. Néanmoins, le consensus général est que l'amélioration du potentiel des machines à interagir avec les êtres humains de manière plus naturelle est essentielle pour l'épanouissement de la relation entre l'IA et les humains.

L'intelligence artificielle a révolutionné l'environnement de la communication à une vitesse sans précédent. Ceci englobe non seulement les changements drastiques de l'industrie médiatique mais implique également les partenaires avec lesquels nous interagissons. Récemment, l'interaction avec les ordinateurs est devenue très courante. Ils sont allés au-delà du simple fait de devenir un moyen de dialogue entre les personnes et peuvent alors interagir avec les humains sous forme d'agents conversationnels et d'assistants virtuels. En tirant profit des mesures avancées de l'apprentissage profond et du traitement automatique du langage naturel (NLP), l'intelligence artificielle conversationnelle peut être affinée en une véritable transformation de l'expérience utilisateur. Ces derniers n'auront plus à surmonter les limitations des chatbots primitifs qui échouent en raison de leur portée limitée.

Au fil du temps, les chatbots sont devenus de plus en plus élaborés en adoptant de nouvelles fonctionnalités d'intelligence artificielle pour améliorer l'expérience utilisateur. L'une de ces fonctionnalités est l'analyse des sentiments, qui permet à l'agent conversationnel d'identifier les émotions derrière le message de l'utilisateur. Intrinsèquement, l'analyse des sentiments est un processus consistant à déterminer le ton émotionnel derrière une série de mots. Elle est utilisée pour comprendre les opinions et les émotions exprimées dans une mention en ligne. L'origine de l'analyse des sentiments remonte aux années 50, lorsqu'elle était particulièrement utilisée dans

les documents manuscrits. De nos jours, l'analyse des sentiments est amplement utilisée pour analyser des informations subjectives du contenu en ligne, notamment des forums, des blogs, des tweets, des médias sociaux, des commentaires et des critiques. Ceci est réalisé grâce à diverses techniques de NLP, de statistiques et d'apprentissage automatique. Les organisations exploitent aujourd'hui la puissance de l'analyse des sentiments pour déceler des informations précieuses afin de mieux comprendre les conversations des consommateurs et ainsi, prendre des mesures plus efficaces et ciblées.

En combinant l'analyse des sentiments aux agents conversationnels, l'industrie est révolutionnée, remplaçant les travailleurs par des chatbots, les obligeant à accorder plus d'attention au secteur des services. La révolution numérique s'attaque désormais à ce domaine avec des agents conversationnels conçus pour être aussi performants que les représentants du service client ou les assistants d'achat. Cette association d'agents conversationnels et d'analyse de sentiments peut accomplir davantage en déterminant ce que les gens veulent clairement dire lorsqu'ils envoient des e-mails, des tweets ou interagissent avec une marque. Grâce à sa capacité précise d'extraire des informations émotionnelles à partir de textes simples, l'analyse des sentiments favorise un service client, une planification de produit et une mise en œuvre d'agents conversationnels plus efficaces.

Problématique et objectifs

Toute entreprise qui propose un produit ou un service doit répondre pleinement aux besoins de ses clients et être prête à répondre à leurs attentes ou à leur fournir des informations si nécessaire : c'est la raison d'être du service client. Les agents du service client suivent certaines formalités et s'efforcent de répondre aux contraintes administratives et techniques en de brefs délais. Avec l'aide des agents conversationnels, l'intelligence artificielle redéfinit radicalement le domaine du service client.

Dans les premiers stades, les agents conversationnels avaient des capacités limitées en fournissant des réponses standards et peu élaborées. Avec le développement de l'intelligence artificielle et de l'apprentissage automatique, les agents conversationnels sont devenus plus performants et ont intégré de nouvelles fonctionnalités qui

contribuent à améliorer l'expérience utilisateur globalement, et le service client spécialement. L'analyse des sentiments est l'une des fonctionnalités les plus récentes permettant d'élever l'expérience utilisateur à un niveau supérieur. Elle peut systématiquement comprendre le sentiment des clients en fonction de la croissance explosive des données clients disponibles aujourd'hui.

Notre projet se situe ainsi à la croisée de deux domaines de recherche : celui de l'analyse des sentiments et celui des agents conversationnels. Les principaux objectifs de ce travail de recherche sont les suivants :

- Développement d'un agent conversationnel pouvant répondre aux requêtes des clients de façon naturelle et élaborée.
- Intégration de l'analyse des sentiments au système de dialogue pour déterminer si le sentiment de l'utilisateur est positif ou négatif afin d'améliorer l'expérience client.

Organisation du mémoire

Dans le but de faciliter la lecture du présent mémoire, nous allons présenter un bref aperçu des chapitres qui le composent. Ces derniers sont organisés comme suit :

- **Chapitre 1 - Agents Conversationnels :** ce chapitre sera consacré à la définition des agents conversationnels, leurs types et leur fonctionnement ainsi que la présentation des différents agents conversationnels ayant marqué l'histoire. Ensuite, nous allons définir quelques mesures et critères qui permettent d'évaluer les agents conversationnels. Pour finir, nous allons faire le lien entre les agents conversationnels et l'analyse des sentiments en énumérant quelques travaux récents ayant combiné ces deux domaines.
- **Chapitre 2 - Analyse des Sentiments :** dans ce chapitre, nous allons aborder les notions fondamentales de l'analyse des sentiments, notamment la polarité et l'intensité de l'opinion ainsi que les différents niveaux de granularité. Par la suite, nous allons énumérer les diverses approches qui constituent l'analyse des sentiments. Enfin, nous allons présenter les défis et challenges pouvant être rencontrés lors de l'analyse des sentiments.
- **Chapitre 3 - Apprentissage Profond :** l'objectif de ce chapitre est de couvrir les notions principales de l'apprentissage profond, en commençant

par définir l'apprentissage automatique et ses différents types, pour finalement aboutir vers les réseaux de neurones récurrents profonds et les améliorations majeures ayant été réalisées.

- **Chapitre 4 - Conception et Modélisation :** dans ce chapitre, nous allons commencer par rappeler la problématique initiale et les objectifs de ce travail. Nous allons par la suite présenter l'architecture globale que nous avons choisi d'implémenter pour notre système, pour ensuite détailler chacune des étapes de prétraitement des données, de traitement ou d'apprentissage et enfin de post-traitement.
- **Chapitre 5 – Tests et Validation du Système :** ce dernier chapitre sera initialement destiné à la présentation des choix matériels et logiciels permettant de développer notre système. Par la suite, l'architecture de ce dernier sera présentée en rappelant les principales étapes de prétraitement, traitement et post-traitement de façon technique. Finalement, nous introduirons les différentes métriques sélectionnées pour évaluer notre système avant de présenter les résultats de cette évaluation.
- **Conclusion générale :** notre mémoire s'achèvera par une conclusion générale qui rappellera la problématique et les objectifs de notre travail ainsi que les résultats obtenus et les perspectives futures.

Agents Conversationnels

1. Introduction

Pour répondre aux enjeux conversationnels, à l'explosion des canaux de communication, à la nécessité de répondre partout et à tout moment aux utilisateurs : l'intelligence artificielle et les technologies connexes procurent une expérience utilisateur riche et palpable grâce aux agents conversationnels qui sont désormais capables d'interpréter les données non structurées de la même manière que les êtres humains et peuvent apprendre des interactions et des conversations en un rien de temps. Cela engendre une multitude de possibilités faisant naître de nouvelles analyses de rentabilité en peu de temps. Propulsés sur le devant de la scène par les prouesses de l'intelligence artificielle, les agents conversationnels engendrent des changements radicaux du paysage médiatique mais gagnent aussi en popularité dans divers scénarios dans lesquels ils sont utilisés sous forme de chatbots, de socialbots et d'assistants virtuels.

Dans le présent chapitre, nous allons passer en revue les principaux agents conversationnels ayant existé à travers l'histoire, définir leurs types, expliquer leur fonctionnement et finalement citer les différentes mesures permettant leur évaluation.

2. Interaction Homme-Machine

L'interaction homme – machine (IHM) est une pratique interdisciplinaire réunissant ingénierie, ergonomie, conception, psychologie, etc. qui traite de la théorie, de la conception, de la mise en œuvre et de l'évaluation de systèmes interactifs reliant l'homme (les utilisateurs) aux appareils de calcul en prenant en considération ses impacts sociétaux et éthiques [1].

A l'origine, l'IHM, faisant surface dans les années 80 avec l'avènement de l'informatique personnelle, se focalisait principalement sur les ordinateurs, mais depuis lors, elle s'est amplifiée pour inclure toutes les variantes des technologies de l'information [2]. L'objectif de l'IHM est de pouvoir créer un système à la fois convivial, fonctionnel et sûr, facilitant ainsi la communication entre l'homme et la

machine et permettant à l'utilisateur de compléter une tâche le plus rapidement et précisément possible [2].

Un facteur important figure dans l'IHM : divers utilisateurs représentent diverses conceptions ou modèles mentaux en fonction de leurs interactions, et ont également différentes méthodes d'apprentissages et de conservation des connaissances et des compétences ce qu'on appelle différents « styles cognitifs » [3]. En addition, les différences culturelles jouent un rôle primordial. Une autre observation à prendre en considération dans l'étude ou la conception d'IHM est que les technologies d'interface utilisateur (UI) et d'expérience utilisateur (UX) évoluent rapidement, offrant ainsi de nouvelles éventualités d'interaction aux quelles les résultats de recherche antérieures peuvent ne pas s'appliquer [4]. Pour finir, les préférences des utilisateurs varient au fur et à mesure de leur maitrise progressive de nouvelles interfaces [4].

3. Définition d'un agent conversationnel

Un agent conversationnel est tout système de dialogue avec lequel les gens interagissent via une interface de chat. Cette interaction n'est pas effectuée uniquement avec du texte, le système peut également lire (canal d'entrée) et répondre (canal de sortie) avec de la parole, des graphiques, des gestes virtuels ou physiques assistés par l'haptique [5].

L'idée est qu'un chatbot puisse s'insérer dans une conversation de telle sorte qu'on ne distingue plus le bot de l'humain, en répondant de façon identique [6]. De nombreux agents conversationnels s'appuient sur l'intelligence artificielle afin de simuler la façon dont les humains communiquent. Plus précisément, les chatbots intelligents reposent souvent sur l'*apprentissage automatique* pour comprendre le langage, formules des phrases, dégager du sens et des émotions à travers du langage oral et écrit, et ainsi s'améliorer automatiquement en développant une base de connaissances agrémentée à partir de diverses sources. En somme, l'intelligence artificielle est un outil ou service du chatbot lui permettant de s'optimiser au fur et à mesure [6].

Néanmoins, pour tenir une conversation rationnelle, il est indispensable d'en avoir une bonne compréhension, mais la majorité des chatbots ne s'y prêtent pas [5]. Ils se contentent de cerner quelques mots dits « *déclencheurs* » dans les phrases de leur interlocuteur, pour trouver les réponses adéquates dont le schéma est déjà programmé, et étant susceptibles d'aboutir vers une discussion plus profonde d'une façon plus ou

moins intelligente, sans avoir besoin de comprendre ce dont parle l'interlocuteur [7]. Le défaut majeur de cette méthode de « *mots-clés* » est de requérir une base de données immense pour produire des résultats plus ou moins satisfaisants [7]. De plus, cette méthode ne tient que peu de temps (typiquement deux ou trois échanges), avant que le discours du chatbot ne soit perçu comme étant artificiel. Cependant, certains agents conversationnels tentent de remédier à ce problème en combinant deux méthodes : la détection de mots-clés et l'analyse linguistique sur la phrase entrée par l'interlocuteur pour essayer d'en déduire les données, leur permettent de répondre de manière cohérente selon les informations disponibles dans leurs bases de connaissances [8]. Dans le cas où ce premier processus échoue, l'agent utilise la méthode des mots-clés en ayant recours à une base de données classique. Cette dernière n'optimise certes pas la dimension de la base de connaissances, mais elle permet en revanche à l'agent conversationnel d'étudier de nouveaux cas de langue [8].

4. Types d'agents conversationnels

Au cours des dernières années et avec la naissance de nouvelles technologies, le domaine des chatbots est devenu tellement vaste qu'une classification exacte des chatbots est devenue subjective par rapport à leur utilisation. Les agents conversationnels pourraient être divisés en diverses catégories selon différents critères, comme le domaine de connaissance, leur mode d'interaction, leur utilisation et technique de conception. Une classification peut être effectuée selon les critères suivants [9]:

- Le mode d'interaction (basé sur le texte ou basé sur la voix).
- Spécifique à un domaine ou ouvert sur plusieurs domaines.
- L'application du chatbot (orienté tâches ou orienté données).
- Basé sur les règles (rule-based) ou intelligence artificielle (Machine Learning, Deep Learning, etc.).

De façon générale, les agents conversationnels sont classés en deux principales catégories selon leurs objectifs [9]:

4.1. Agents conversationnels orientés tâches

Les chatbots axés sur les tâches sont des programmes à vocation unique conçus pour exécuter une tâche bien particulière et configurés pour entretenir des

conversations courtes, rapides et limitées, généralement dans un domaine fermé [10]. Construits via des éléments graphiques, utilisant des carrousels ou des formats CTA (Call-To-Action), la conversation est dirigée par le chatbot selon un workflow de réponses et d'échanges structurés et spécifiques [11]. Ces réponses sont tirées d'une base de données et déterminées selon des règles implémentées par un développeur. Ces règles peuvent être modifiées selon les besoins pour améliorer incessamment le fonctionnement du chatbot [11].

La figure ci-dessous représente une conversation que nous avons entretenu avec un agent conversationnel axé sur les tâches. Comme on peut le constater, cet agent[1] est incapable d'entretenir une conversation basique qui sort de son contexte initial :

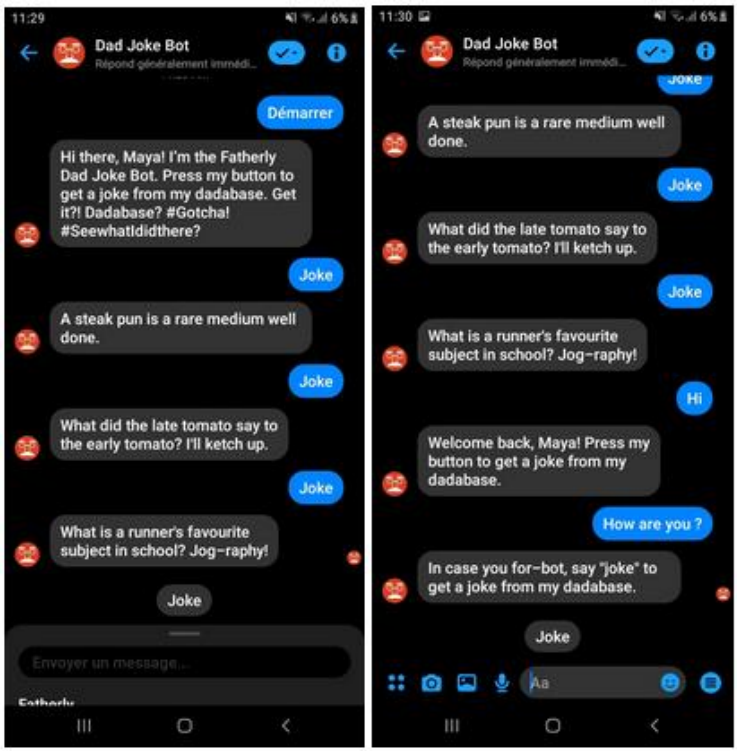

FIGURE 1.1 - Agent conversationnel orienté tâches.

4.2. Agents conversationnels orientés données

Les chatbots axés sur les données sont bien plus sophistiqués, personnalisés et interactifs que ceux axés sur les tâches [10]. Souvent désignés comme assistants numériques ou virtuels, ils reposent sur l'Intelligence Artificielle et le Machine

[1] **URL:** https://www.facebook.com/thedadjokebot/

Learning pour permettre la personnalisation selon les profils des utilisateurs et leurs comportements antérieurs, et ainsi fournir de meilleures réponses [10].

Ce type d'agents a la capacité d'apprendre constamment à partir de ses interactions avec les utilisateurs pour mieux prédire leurs besoins, et ce grâce à la technologie du traitement du langage naturel (NLP) construite sur des Frameworks assez complexes [12]. Il peut notamment classer et stocker les données qu'il récolte de chaque échange, dans des localisations précises afin de pouvoir y accéder de nouveau à l'avenir [12].

La figure ci-dessous montre une discussion que nous avons tenu avec un agent[2] conversationnel orienté données :

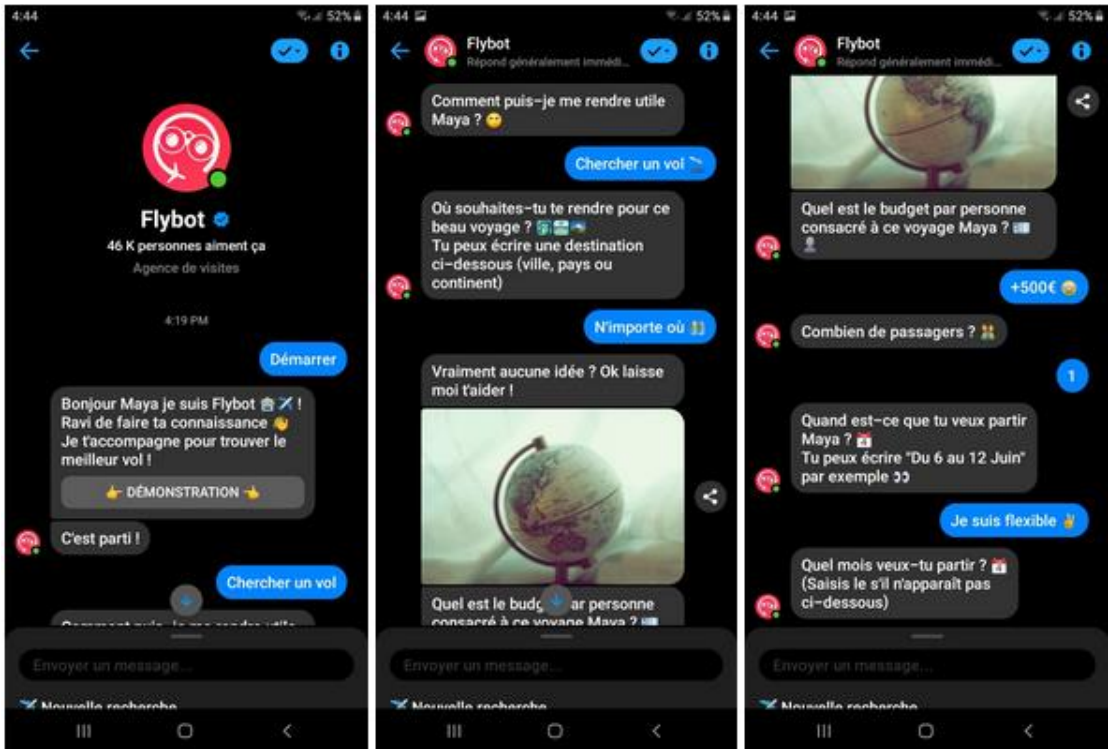

FIGURE 1.2 - Agent conversationnel orienté données.

5. Agents conversationnels à travers le temps

L'idée de créer un agent ayant des processus de pensée et des compétences conversationnelles semblables à celles de l'homme existe depuis des siècles. Les philosophes, les scientifiques, et même les sculpteurs étaient ébahis par l'abstraction d'un automate humaniste [13]. L'évolution des agents conversationnels à travers le temps est représentée par le schéma ci-dessous :

[2] **URL:** https://www.facebook.com/flybotFR/

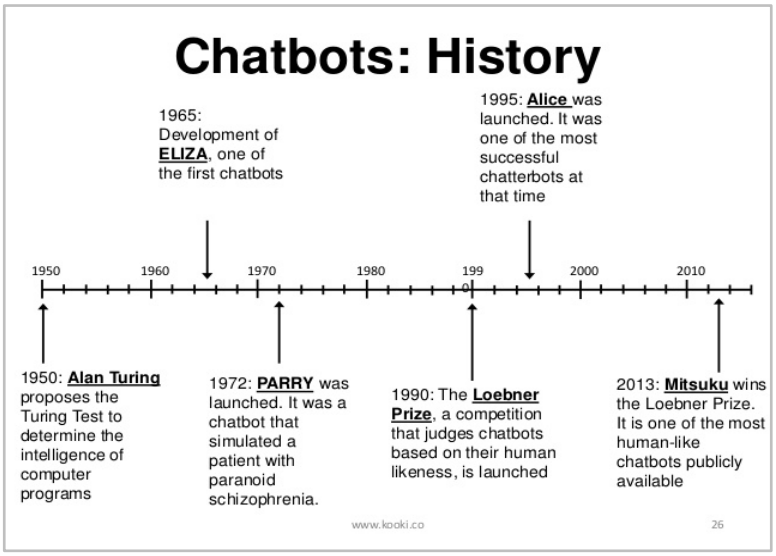

FIGURE 1.3 - Histoire des chatbots.

1950 – Le test de Turing

En 1950, Alan Turing, mathématicien et cryptologue britannique théorise pour la première fois l'idée qu'une machine puisse entretenir une conversation humaine [14]. Il met ainsi au point un test rebaptisé « test de Turing » ayant pour vocation de répondre à la question : « une machine peut-elle penser ? », c'est-à-dire mesure la capacité de l'IA à être confondue avec l'intelligence humaine. Le principe de ce test est de mettre en situation une personne qui va dialoguer avec deux interlocuteurs pendant un certain laps de temps. Si l'interrogateur n'est pas en mesure de décerner l'humain de la machine, à la fin du test, alors la machine aura réussi le test de Turing [14].

1966 - Eliza

Considéré comme le premier agent conversationnel de l'histoire de l'informatique et étant capable de passer avec succès le test de Turing, Eliza a été conçu par Joseph Weizenbaum alors professeur au Massachusetts : Institue Of Technology (MIT) [15]. Cet agent consistait à imiter un psychothérapeute rogérien capable de reconnaitre les mots ou expressions clés de l'entrée de l'interlocuteur pour les utiliser afin de reproduire des réponses cohérentes [16].

1972 - Parry

Alors qu'Eliza simulait un thérapeute rogérien, Parry, conçu par le psychiatre Kenneth Colby à l'université de Stanford, a tenté de simuler le comportement d'un patient atteint de schizophrénie paranoïde [13].

1988 - Jabberwacky

L'objectif de ce chatterbot, créé par le programmeur britannique Rollo Carpenter, est de simuler une conversation humaine d'une façon intéressante, divertissante et humoristique [17]. Il s'agit ici d'une toute première tentative de création d'intelligence artificielle à partir d'interaction humaine, capable de passer le test de Turing [17].

1995 - Alice

Contrairement à Eliza, le chatbot Alice développé par Richard Wallace, est doté d'un système d'identification relatif à la personnalité de l'interlocuteur et dispose d'une base de connaissance bien plus vaste que ses prédécesseurs, contenant 40.000 catégories de connaissances et peut répondre de façon affirmative [18].

2013 - Mitsuku

Créé grâce à la technologie AIML (Artificial Intelligence Markup Language) par Steve Worswick pour le concours du prix Loebner dont il est lauréat à cinq reprises, Mitsuku prétend être un chatbot femelle âgée de 18 ans [19].

Avec l'essor d'internet et de l'informatique mobile, la recherche en IA a donné naissance à diverses interfaces conversationnelles reposant sur le *Machine Learning* et le *Natural Language Processing* où dans les années 2010, Apple, Google, Amazon et Microsoft lancent leurs propres agents conversationnels respectivement Siri, Google Now, Alexa et Cortana, capables de répondre à des questions formulées en langage naturel, mais c'est bien Facebook qui lance le coup d'envoi de la révolution des agents conversationnels avec l'arrivée de Messenger en 2016. Depuis lors, ils sont devenus bien plus que de simples agents conversationnels pour certains, ce sont maintenant des assistants virtuels capables de résoudre des tâches assez complexes.

6. Fonctionnement des agents conversationnels

Le plan de fonctionnement d'un agent conversationnel est toujours le même, quel que soit son niveau, sa thématique ou son périmètre [20] :

1. L'utilisateur communique sa requête dans un langage naturel via une interface de chat textuelle ou vocale.

2. L'agent conversationnel interprète la requête via son moteur pour la comprendre.
3. L'agent conversationnel génère une réponse unique et adéquate à la demande de l'utilisateur.

Cependant, les agents conversationnels orientés tâches utilisent ce que l'on appelle un « *Knowledge Graph* » qui est une sorte d'encyclopédie lisible par les machines [21]. Techniquement parlant, un Knowledge Graph est une variante d'un réseau sémantique avec des contraintes additionnelles dont les caractéristiques, la structure et les utilisations sont constamment en cours de développement [21]. Plus concrètement, le Knowledge Graph représente des données organisées de telle façon qu'une machine puisse aisément les assimiler et en tirer des informations pertinentes [22]. Il fournit également une structure et une interface commune pour toutes les données et permet la création de relations multilatérales intelligentes dans toutes les bases de données, où, structuré en tant que couche de données virtuelle supplémentaire, se trouvera au-dessus des bases de données et pourra les relier à grande échelle, qu'elles soient structurées ou non [22].

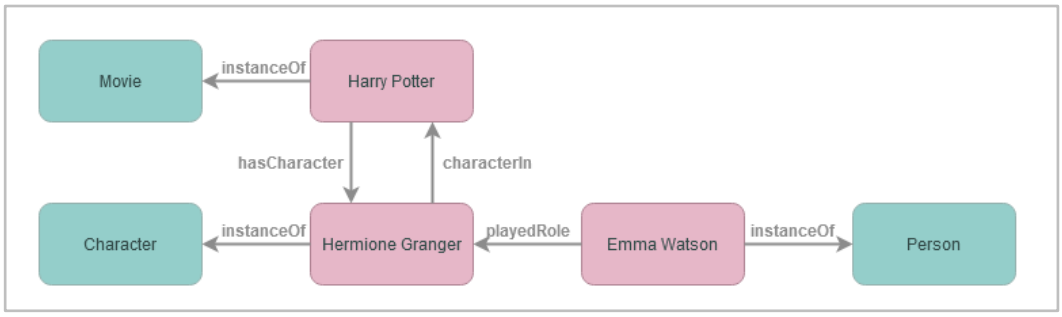

FIGURE 1.4 - Exemple de Knowledge Graph.

Le graphe ci-dessus représente un échantillon d'un large Knowledge Graph. Le graphe permet de répondre directement à des questions telle que : « Qui a joué Hermione Granger dans le film Harry Potter ? ». Grâce au *Natural Language Processing*, une réponse est fournie en transformant d'abord la question en une représentation de forme logique pouvant adresser des entités et des relations [23]. Des dizaines de Knowledge Graphs ont vu le jour et sont mis à contribution avec des API permettant aux développeurs d'applications d'y accéder.

Différents programmes permettent aujourd'hui la création facile de ce genre d'agents conversationnels sans connaissances préalables de langages de

programmation [24]. A titre d'exemple, *Wit.ai* et *Chatfuel* qui sont utilisées pour l'implémentation de chatbots sur Facebook Messenger, ou encore *Kore* qui propose de créer des chatbots sur mesure pour les banques, les services ou le commerce.

Dans ce travail, nous allons nous concentrer sur les agents conversationnels axés sur les données qui sont beaucoup plus élaborés et donc leur fonctionnement plus complexe. Un tel chatbot est constitué de plusieurs ensembles de composants régis par ce qu'on appelle un « *gestionnaire de dialogue* », qui est également responsable de la gestion de l'état et de la stratégie de dialogue [10]. Un cycle d'activités typique d'un agent conversationnel contient l'ensemble des phases suivantes [25]:

1. L'utilisateur transmet sa requête à travers la parole qui sera convertie en texte brut grâce au système de reconnaissance d'entrée. Ce système peut inclure :
   - De la reconnaissance automatique de la parole.
   - De la reconnaissance de gestes.
   - De la reconnaissance d'écriture manuscrite.
2. Ce texte brut est alors analysé par une unité de compréhension du langage naturel (Natural Language Understanding Unit), qui peut inclure à son tour :
   - L'identification de noms propres.
   - Part-of-speech tagging.
   - Analyseur sémantique / syntaxique.
3. Les informations recueillies par cette unité sont ensuite analysées par le gestionnaire de dialogue qui va conserver l'historique et l'état du dialogue et engendrer le flux général de la conversation.
4. Généralement, un ou plusieurs gestionnaires de tâches connaissant le domaine de tâches spécifique sont contactées par le gestionnaire de dialogue.
5. Une sortie est finalement produite grâce au générateur de sorties, qui peut contenir :
   - Un générateur de langage naturel.
   - Un générateur de gestes.
   - Un layout manager.
6. Enfin, la sortie est diffusée à l'utilisateur selon le mode utilisé (de la parole par exemple).

Le cycle d'activité des agents conversationnels ne disposant que d'une interface textuelle (conversations textuelles uniquement), ne comprend que les étapes de 2 à 5 [25] – ceci fera l'objet de notre présente étude –.

7. Critères d'évaluation des agents conversationnels

Au fil des siècles, les scientifiques, psychologues et philosophes ne sont jamais parvenus à définir concrètement la notion de pensée et d'intelligence, de ce fait il devient difficile de spécifier le cadre de l'intelligence artificielle. Vient alors le *test de Turing*, qui, doté d'une extrême simplicité, permet de mettre en place un étalon de mesure, malgré toutes ses imperfections [26]. Ce test n'évalue nullement les connaissances techniques de l'IA, de ce fait ce ne sont pas les capacités d'accumulation de savoir ou de traitement de l'information qui sont mises à l'épreuve, mais simplement sa faculté à gagner un jeu mondain consistant à se faire passer pour un être humain et à faire preuve d'empathie, car c'est bien l'élément clé permettant de développer des IA bienveillante [26].

Cependant, depuis sa naissance, le test de Turing a été sujet de critiques à maintes reprises par les scientifiques et philosophes. La crédibilité du jugement par l'investigateur, l'intérêt de comparer le comportement d'une machine à celui d'un humain, comptent parmi les multiples points contestés par les sceptiques [27].

On peut accuser ce test de ne pas évaluer directement l'intelligence de la machine, mais plutôt sa capacité à imiter le comportement humain [26]. Or, celui-ci n'est pas obligatoirement intelligent. Dans certains cas, on pourrait la qualifier de stupidité artificielle et non pas d'intelligence artificielle. Ceci est le cas de la première IA ayant remporté le prix Loebner, parvenue à duper ses interlocuteurs en mimant des fautes de frappe [28]. Dans d'autres cas, certains comportements intelligents n'ont rien d'humain. A titre d'exemple, si une machine s'avère plus intelligente qu'un être humain en résolvant une équation mathématique complexe, elle risque de ne pas réussir le test car les interrogateurs présumeront qu'il s'agit d'une machine [28]. C'est pourquoi de nombreuses alternatives ont été suggérées pour évaluer la super-intelligence.

A mesure que les agents conversationnels deviennent de plus en plus complexes, il est primordial de développer de nouvelles méthodologies et procédures d'évaluation pour mesurer leur performance. De nombreuses difficultés sont rencontrées pour

définir de nouvelles mesures acceptées par la communauté scientifique car ce domaine est considéré comme étant à un stade précoce de développement [29].

Diverses initiatives ont vu le jour, durant ces dernières années, dont le but est de définir des Frameworks incluant le design et l'évaluation des agents conversationnels tout en considérant deux tendances fondamentales [29]: la première étant la définition de mesures quantitatives pour estimer la qualité de l'agent (ex : EAGLES et DARPA Communicator Projects) [30], la deuxième, consiste en des propositions de définitions de mesures à la fois quantitatives et qualitatives (ex : ELSE et DISC Projects) [20]. De nombreuses autres suggestions peuvent être trouvées dans la littérature pour évaluer les performances des agents conversationnels. A titre d'illustration, Dybkjaer et Bernsen (2000) ont proposé un ensemble de 15 normes pour garantir le bon usage des agents conversationnels [20]. Ces normes peuvent être soit objectives - tirées directement de l'interaction avec le système, à l'exclusion de tout type d'évaluation effectuée par les développeurs ou les utilisateurs -, soit subjectives - comprenant un processus d'évaluation généralement effectué par les utilisateurs finaux de l'agent - [31].

Compte tenu de l'objectif de l'évaluation, deux types d'évaluation peuvent également être discernés : la *black box,* considère la performance globale du chatbot en ne prenant en compte que les entrées et sorties [32], la *crystal box,* se focalise sur les composants de l'agent individuellement, en tenant compte des entrées et des sorties.

Parmi les autres outils les plus couramment utilisés pour l'évaluation des agents conversationnels, on retrouve PARADISE (PARAdigm for DIalogue System Evaluation), une Framework qui assemble diverses fonctionnalités en une seule qui mesure les performances des agents directement corrélées à la satisfaction des utilisateurs [26]. On retrouve également Wizard of Oz qui est un processus consistant à laisser l'utilisateur interagir avec une interface sans savoir que les réponses sont générées par un humain qui tire les leviers et actionne les commutateurs dans les coulisses, au lieu d'un ordinateur [34]. Cette technique est généralement utilisée pour émuler les performances du système.

8. Agents conversationnels et Analyse des Sentiments

L'analyse des sentiments est une technique de plus en plus adoptée dans l'implémentation de nombreuses versions d'agents conversationnels à travers différents domaines d'application. L'analyse des sentiments consiste à interpréter et classifier des émotions (positives, négatives ou neutres) dans des données textuelles en utilisant des techniques d'analyse textuelle. Ces techniques d'analyse sont divisées en deux principales catégories : des techniques qui reposent sur *l'apprentissage automatique* et des techniques qui reposent sur *le lexique*. Ces deux catégories sont à leur tour divisées en plusieurs sous catégories qui seront détaillées dans le chapitre suivant. Par conséquent, l'analyse des sentiments est une couche au-dessus du moteur de compréhension du langage naturel de l'agent conventionnel. C'est une fonctionnalité supplémentaire permettant à l'agent de « comprendre » les sentiments d'un utilisateur. Plusieurs travaux à travers la littérature ont combiné les performances de l'analyse des sentiments avec les agents conversationnels pour en tirer pleinement partie. Parmi ces travaux nous citons :

- La méthode adoptée par Wei et al. [35] vise à concevoir des dialogues non seulement adéquats en termes de contenu, mais qui disposent également de sentiments particuliers. Une analyse de sentiments a été établie l'ensemble des données pour annoter la catégorie des sentiments. La conversation ayant les sentiments les plus importants a été sélectionnée. Tout d'abord, la corrélation du niveau de sentiment est modifiée à l'aide d'un réseau de neurones Séquence à Séquence. Puis des récompenses de sentiments sont introduites pour gratifier le modèle lors de l'apprentissage par renforcement. Une troisième méthode est utilisée afin de réduire le temps d'entrainement par renforcement profond et pour intégrer des balises émotionnelles. Cette méthode consiste à entrainer un réseau neural de dialogue émotionnel. La mémoire externe et la mémoire interne sont combinées avec l'intégration des sentiments pour transférer judicieusement le flux d'information et la classification du sentiment vers la sortie finale du réseau. Cependant, cette méthode présente des inconvénients. Les annotations émotionnelles de l'ensemble des données ont un impact important sur les performances.

Davantage d'échantillons doivent être collectés afin d'améliorer les performances.

- Dans un autre travail, Lee et al. [36] proposent de construire un agent conversationnel selon le modèle Séquence à Séquence conventionnel et suggèrent cinq modèles permettant d'ajuster ou de mettre à l'échelle le sentiment de la réponse de l'agent. Le modèle basé sur le Persona, le modèle Plug and Play, l'apprentissage par renforcement, le réseau de transformation du sentiment et le cycleGAN. Le modèle basé sur le Persona a été à l'origine proposé pour produire des réponses qui imitent celles d'un locuteur spécifique. Il est très similaire au modèle Seq2Seq à l'exception d'informations additionnelles ajoutées au décodeur à chaque pas de temps. Le modèle Plug and Play quant à lui a été emprunté du concept initial pour la génération d'images, pour générer des réponses de conversation dans ce cas-là. L'apprentissage par renforcement à son tour, utilise le même modèle Seq2Seq que le chatbot mais en ajoutant un ensemble de fonctions de récompense conçues pour ajuster les sentiments de la réponse. Pour le réseau de transformation de sentiment, il est très similaire au modèle Plug and Play sauf qu'il cartographie un vecteur latent à un autre vecteur ou maximise la fonction objectif. Finalement le modèle cycleGAN est utilisé pour transformer le sentiment d'une phrase de négatif à positif. Les résultats ont montré que l'apprentissage par renforcement et le cycleGAN sont les plus performants.
- Leggeri et al. [37] proposent de construire un agent conversationnel orienté tâches à travers un algorithme qui a la capacité d'optimiser la performance de l'apprentissage selon les réactions et les émotions produites par la conversation entre l'humain et l'agent. Pour cette raison, une fonction d'erreur a été étudiée visant à corriger l'entrée afin d'améliorer la sortie. Cette méthode se base parallèlement sur le calibrage de l'interprétation du dataset initial ainsi que l'expansion du lexique avec de nouveaux termes. L'idée derrière cette méthode est d'analyser la satisfaction de l'utilisateur afin d'améliorer la compétence du processus d'apprentissage de l'agent et plus formellement, pour créer et ajouter automatiquement de nouveaux

termes annotés à l'ensemble des données d'apprentissage dans le but d'affiner le modèle appris par l'agent. Afin de détecter le sentiment de l'utilisateur, une approche Bayes naïf est utilisée pour construire ce classifieur sur un ensemble particulier de données. La difficulté rencontrée dans cette approche est la nature de la source des données discursive ne peut être définie à l'avance, laissant ainsi le domaine inconnu de l'ensemble non labellisé jusqu'à ce qu'il soit écrit par un utilisateur.

- Dans leur approche, Zhou et al. [38] ont résolu le problème de la génération de réponses « émotionnelles » dans les systèmes de dialogue à domaine ouvert en proposant une Emotional Chating Machine (ECM en abrégé). Dans le but d'obtenir des données labellisées avec des sentiments, un classifieur neural est formé sur un corpus déjà annoté manuellement. Ce classifieur sera utilisé pour labelliser automatiquement les données conversationnelles à grande échelle pour l'apprentissage de l'ECM. Un modèle génératif Seq2Seq est ensuite conçu afin d'exprimer les sentiments de façon naturelle et homogène dans les réponses. Ce modèle est muni de nouveaux mécanismes de génération d'expressions émotionnelles, à savoir, l'incorporation de la classe des sentiments pour capturer le haut niveau d'abstraction dans les expressions émotionnelles, l'état émotionnel interne pour garder l'équilibre grammatical et émotionnel de façon dynamique, et enfin, une mémoire d'émotion externe qui aide à générer des expressions émotionnelles plus claires et cohérentes. La limitation principale rencontrée dans cette méthode est le manque de données d'apprentissage et les erreurs causées par le classifieur d'émotions, de sorte que les réponses ne sont pas appropriées à la catégorie d'émotions.

Le tableau dressé ci-dessous vise à récapituler et à comparer ces différents travaux :

TABLEAU 1.1 – Travaux portant sur l'analyse des sentiments dans les agents conversationnels.

Auteur et Année	Technique d'Analyse des sentiments	Type d'agent conversationnel	Avantages	Inconvénients
Wei et al. (2017)	Approche basée sur le Machine Learning (*modèle Seq2Seq*)	Orienté Données	L'intégration des émotions, la mémoire interne et la mémoire externe contribuent à générer des réponses raisonnables émotionnellement et au niveau du contenu.	Les annotations émotionnelles de l'ensemble des données présentent un impact négatif important sur les performances de l'agent conversationnel.
Lee et al. (2018)	Approche basée sur le Machine Learning (*modèle basé sur le Persona*, *modèle Plug and Play*, *Apprentissage par Renforcement*, *réseau de transformation de sentiments*, *cycleGAN*)	Orienté Données	L'*apprentissage par renforcement* apprend correctement les conceptions visées et propose des réponses diversifiées. Le *cycleGAN* préserve la qualité de la sortie en alignant les mots sur la réponse d'origine.	Le *modèle Plug and Play* et le *réseau de transformation de sentiment* ne sont pas efficaces car la modification du code latent des phrases tout en préservant la qualité et la sémantique de la phrase représente une tâche difficile.
Leggeri et al. (2018)	Approche probabiliste basée sur le Machine Learning (*Bayes naïf*)	Orienté Tâches	L'approche de l'analyse des sentiments améliore le dataset automatiquement et en temps réel. Les réponses et l'apprentissage sont gérés par l'intervention d'un tiers pour les corriger manuellement.	La nature de la source des données discursive ne peut être définie auparavant et laisse le domaine de l'ensemble non annoté inconnu.

Zhou et al. (2018)	Approche basée sur le Machine Learning (modèle Seq2Seq)	Orienté Données	L'ECM peut générer des réponses adéquates émotionnellement et contextuellement	Manque de données d'apprentissage et erreurs causées par le classifieur de sentiments, de sorte que les réponses ne sont pas appropriées à la catégorie d'émotions.

9. Conclusion

A travers ce chapitre, nous avons dans un premier temps découvert l'origine des agents conversationnels, l'interaction homme-machine, suivie par le test de Turing. Nous avons défini les différents types d'agents conversationnels et expliqué brièvement leur fonctionnement. Nous avons par la suite énuméré les différentes évolutions qu'a subi l'agent conversationnel à travers l'histoire pour finalement aboutir vers les critères d'évaluation de ce chatbot.

Le chapitre suivant sera consacré à la présentation du domaine de l'*Analyse des Sentiments*, ses notions fondamentales, ses différentes approches et les divers challenges qu'il présente.

Analyse des Sentiments

1. Introduction

Également appelée *Sentiment Analysis, Opinion Mining* ou *Analyse de Tonalité, l'Analyse des Sentiments* est un processus qui consiste à déterminer l'intonation émotionnelle cachée derrière une série de mots, un paragraphe ou un document entier. L'*Analyse de Sentiments* passe par la détection des émotions (par exemple : le bonheur, la tristesse, la colère, etc.) jusqu'au sarcasme et à l'intention (par exemple : les opinions, les commentaires, les plaintes, etc.). Sous sa forme la plus basique, l'analyse des sentiments attribue une polarité (positive, neutre ou négative) ou des scores de sentiments pondérés aux entités, thèmes et catégories d'un bout de texte, et ce, en combinant des techniques de *Traitement Automatique du Langage* (NLP) et de *Deep Learning*.

L'*Analyse des Sentiments* est régulièrement convoquée dans tout processus décisionnel, que ce soit dans le Business pour évaluer la réputation de son produit, dans le contexte d'une élection pour les demandes d'avis de votes ou la clarification des positions des politiciens, ou encore dans les finances pour suivre l'évolution des prix des matières premières et des actions [39].

Dans ce chapitre, nous allons aborder les principales notions de l'analyse des sentiments en mettant l'accent sur les approches les plus utilisées pour finalement présenter les challenges rencontrées lors de l'analyse des sentiments.

2. Opinion Mining et Analyse des Sentiments

Au sens large, le terme *Opinion Mining* ou *Fouille d'Opinion* est la science qui utilise l'analyse de texte (*Text Mining*) pour comprendre les forces motrices du sentiment public. Elle peut être définie comme une sous-discipline de la linguistique informatique qui vise à considérer un ensemble de résultats de recherche, générer une liste d'attributs (caractéristiques, qualités, etc.) et attribuer un avis à chacun d'entre eux (bon, modéré, mauvais) [40].

Le terme *Analyse des Sentiments*, d'autre part, est employé pour décrire l'analyse automatique de texte en évolution et pour la recherche de valeurs de jugement

prévisible. Elle consiste à identifier la subjectivité, la polarité (positive, neutre, négative) et l'intensité de la polarité (fortement positive, légèrement positive, faiblement positive, etc.) d'un bout de texte [41].

Alors que l'*Analyse des Sentiments* – prédécesseur de la *Fouille d'Opinion* – se penche sur le point de vue et le ressenti des gens sur un sujet donné, la *Fouille d'Opinion* va plus loin pour comprendre d'avantage les facteurs responsables de ce que les gens ressentent. Aujourd'hui, l'*Analyse des Sentiments* et l'*Opinion Mining* font partie du même domaine de recherche [42].

3. Polarité et intensité de l'opinion

Avant de s'intéresser à la polarité, il est primordial de distinguer si une phrase donnée correspond à un fait, un sentiment ou une opinion. Une certaine confusion a eu lieu parmi les chercheurs concernant la différence entre le sentiment, l'opinion et le fait. Profondément distincts, ils n'ont ni origine, ni expression, ni impact commun. Ils sont tels trois longueurs d'onde possibles.

Les faits sont objectifs et mesurables, exprimant des choses concrètes, quantifiables et vérifiables. Ils permettent une affirmation indiscutable. Tandis que les opinions représentent une évaluation, une appréciation, un point de vue ou même un jugement [40]. Elles sont toujours subjectives et discutables, et donc sujettes à débat et argumentation. Quant aux sentiments, ils expriment une attitude mentale causée par une émotion, un ressenti ou une sensation. Il peut s'agir de satisfaction, de craintes, d'embarras ou d'agacement. Les sentiments et les émotions sont deux concepts étroitement liés dans l'étude de la psychologie et des sciences cognitives et sont souvent utilisés de manière interchangeable [42].

De nombreuses études dans le domaine de l'Analyse des Sentiments ont eu recours aux échelles de Likert[3] pour définir les sentiments selon un nombre variable d'étapes. À titre d'illustration, le Stanford Sentiment Treebank a utilisé une échelle de Likert à 7 points pour annoter la tentation. De tels schémas d'annotation de sentiments ont deux aspects : la polarité (valeurs positives ou négatives) et l'intensité (degré auquel le mot,

[3] **Echelle de Lickert** : développée par le psychologue américain Rensis Lickert, c'est un outil psychométrique permettant de mesurer une attitude chez des individus selon une ou plusieurs affirmations pour lesquelles la personne interrogée exprime son degré d'accord ou de désaccord.

la phrase, le document en question est positif ou négatif ; hors de la plage des valeurs positives ou négatives : fort, moyen, faible, neutre, etc.) [42].

Une autre possibilité a également été introduite pour déterminer la polarité des mots : l'*Orientation Sémantique* [43]. Elle représente une mesure de la subjectivité et de l'opinion dans le texte. Fondamentalement, elle saisit un facteur d'évaluation (positif ou négatif) et l'intensité envers un sujet, un individu ou un concept. Deux approches distinctes ont été utilisées dans la littérature pour l'approche basée sur l'Orientation Sémantique, à savoir, l'approche basée sur le corpus ainsi que l'approche basée sur le dictionnaire, le lexique ou la connaissance [44]. Bien que cette approche puisse s'avérer extrêmement utile lorsqu'elle est employée dans l'analyse de l'opinion publique, ses performances ont été limitées dans la littérature en raison d'une couverture inadéquate des caractéristiques de plusieurs termes [44].

4. Niveaux de granularité pour l'Analyse des Sentiments

L'Analyse des Sentiments « basique » se focalise sur une seule et unique dimension : le sentiment global du texte est-il positif ou négatif ? Les techniques de détermination d'un ou plusieurs sentiments généraux relèvent plus d'une forme d'adaptation de méthodes de classification ou de méthodes de découverte du sujet. Le choix du niveau de granularité est basé sur la façon dont l'émotion est exprimée dans le texte. Les mots seuls peuvent susciter des émotions, mais lorsqu'ils sont combinés de façon à former des entités lexicales plus grandes telles que des phrases ou des paragraphes, il est souvent constaté que différentes émotions sont exprimées [45]. La figure ci-dessous décrit les niveaux de granularité qui ont été annotés pour l'Analyse des Sentiments :

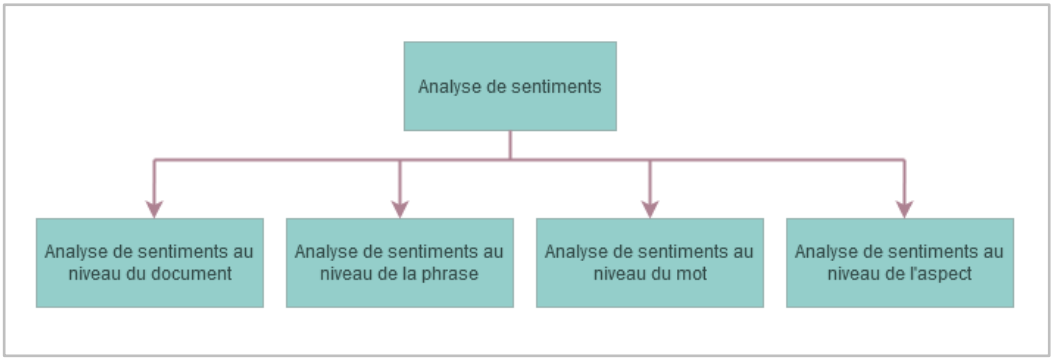

FIGURE 2.1 - Niveaux de granularité pour l'Analyse des Sentiments.

Dans ce qui suit, nous allons définir ces différents niveaux individuellement.

4.1. Au niveau du document

Son principe est de considérer le document entier comme unité d'information de base et de le classer comme exprimant une opinion ou un sentiment positif ou négatif, où, un seul examen d'un seul sujet est pris en considération. Lors de la phase de prétraitement des données, les phrases non pertinentes doivent impérativement être éliminées. Pour la classification au niveau du document, plusieurs méthodes de classification supervisée et non supervisée peuvent être utilisées. Parmi les algorithmes d'apprentissage automatique supervisé pouvant être employés pour entraîner le système, figurent les *Machines à Vecteurs de Support* (Support Vector Machines), la *classification naïve bayésienne* et l'*entropie maximale* [45].

Les phases d'entraînement et d'évaluation du dataset peuvent être effectuées en extrayant les termes d'opinion figurant dans le document et en utilisant des fonctionnalités comme la fréquence de ces termes, les négations et les dépendances. L'étiquetage manuel de la polarité du document est une tâche longue et par conséquent, la note de l'évaluateur disponible par exemple pour le cas des blogs et forums (sous la forme de 1 à 5 étoiles) peut être utilisée [45].

4.2. Au niveau de la phrase

Pour analyser les sentiments à un niveau plus granulaire que le niveau du document, le niveau de la phrase peut être utilisé. L'analyse des sentiments au niveau de la phrase est étroitement liée à la subjectivité. Elle est habituellement divisée en deux classes de problèmes de classification [45]. La première détermine si la phrase exprime ou non un sentiment, la seconde classe cette phrase comme étant positive, neutre ou négative. L'étape initiale vise à distinguer les opinions (phrases subjectives) des faits (phrases objectives), et est connue dans la littérature sous le nom de *Classification de la subjectivité*. La seconde étape est nommée *Classification des sentiments de phrase*. Après avoir discerné les phrases subjectives des phrases objectives, la classification des sentiments de phrase les classe dans des catégories positives, négatives ou neutres, en utilisant les mêmes méthodes de la classification au niveau du document (voir section **Error! Reference source not found.**) [45].

4.3. Au niveau du mot

Annoter chaque mot d'un corpus avec sa connotation émotionnelle correspondante représente une tâche onéreuse en termes de temps et de ressources. Pour cette raison, des techniques innovantes ont vu le jour pour compiler un corpus de de mots avec des connotations émotionnelles [45]. Ces annotations au niveau des mots peuvent être considérées comme des bases de données lexicales de mots affectifs. Ces ressources volumineuses sont principalement utilisées lorsque les connaissances de base font défaut, car elles peuvent résumer l'impact associé à chaque mot. Bien que ces lexiques de mots soient utilisés pour des techniques de correspondance de mots primitifs, ils peuvent présenter des limites en raison du manque de contexte qui peut être tiré à partir d'un seul mot [45].

4.4. Au niveau de l'aspect

L'analyse des sentiments au niveau de l'aspect ou de l'entité permet une analyse encore plus granulaire. Elle implique non seulement la détection du sentiment général lié à une entité, mais aussi la recherche du sentiment pour les aspects de l'entité concernée [45].

Au lieu de spécifier les entités à analyser selon des critères structuraux (mot, phrase, document), ces méthodes au niveau de l'aspect reposent sur une analyse de corrélation entre l'opinion exprimée et la finalité de cette opinion. Généralement, trois principales étapes peuvent être distinguées lors de l'analyse de sentiments au niveau de l'aspect : l'identification, la classification et l'agrégation [46]. La première étape consiste à identifier les paires de sentiment-cible à partir du texte. L'étape qui suit vise à classer ces paires selon un ensemble prédéfini de valeurs de sentiments (nombres positifs et négatifs). à la fin, les valeurs de sentiments de chaque aspect sont agrégées pour fournir un aperçu concis selon les exigences et les spécifications de l'application [46].

5. Approches de l'Analyse des Sentiments

Les approches de l'analyse des sentiments peuvent être divisées en deux catégories : l'*approche basée sur le Machine Learning* et l'*approche basée sur le lexique* [47] :

- L'approche basée sur le Machine Learning combine des algorithmes de Machine Learning connus avec des fonctionnalités linguistiques.

- L'approche basée sur le lexique quant à elle recourt à des collections de termes de sentiments connus et compilés. Ces collections sont appelées lexiques de sentiments.

L'objectif majeur de ces approches est d'identifier les mots de sentiments dans un texte donné en leur attribuant des valeurs de polarité. La figure ci-dessous schématise les différentes approches d'analyse de sentiments qui existent dans la littérature [48] :

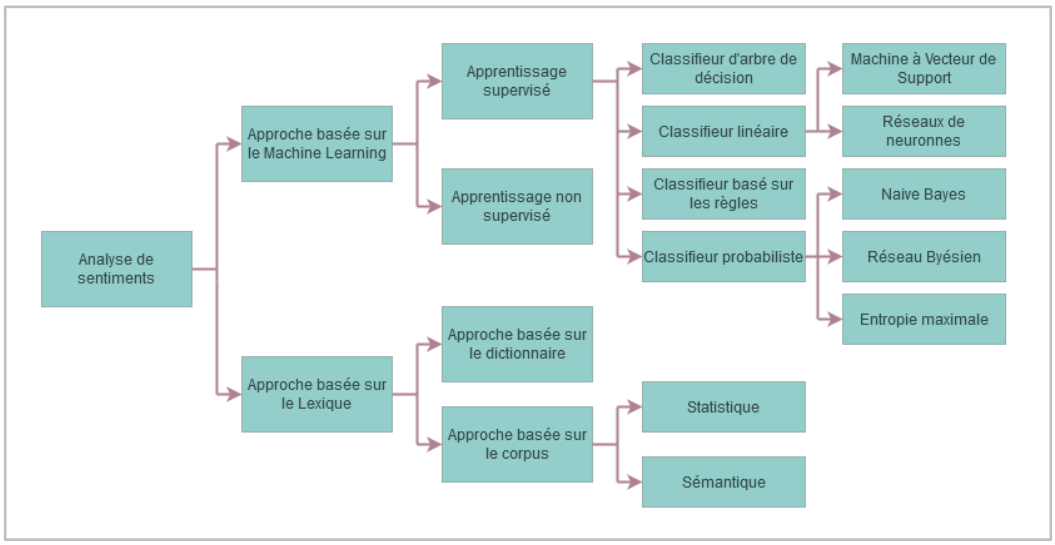

FIGURE 2.2 - Approches de l'Analyse des Sentiments.

Dans ce qui suit, nous allons définir ces différentes approches.

5.1. Approches basées sur le Machine Learning

Arthur Samuel (1959) définit le Machine Learning comme le *domaine d'étude qui donne aux ordinateurs la possibilité d'apprendre sans être explicitement programmés* [49]. Selon cette définition, le Machine Learning peut être appliqué de façon appropriée aux problèmes de la classification de texte, et par voie d'héritage, il peut être conformément associé à l'analyse de sentiments. Des méthodes d'apprentissage supervisé et non supervisé ont été appliquées au défi de l'analyse des sentiments, et les performances se sont avérées efficaces pour certains domaines où la variation d'actualité est minime [50].

5.1.1. Apprentissage supervisé

La méthode de l'analyse des sentiments basée sur l'apprentissage supervisé construit un système qui repose sur des données empiriques étiquetées qui sont fournies en entrée pour créer un classifieur pouvant aisément modéliser le domaine

traité et conduisant à des performances de classification adéquates pour une tâche donnée [50]. Ces méthodes sont appliquées particulièrement aux tâches de classification orientées vers un sujet précis tel que le filtrage de spams. Néanmoins, en raison de la qualité des données d'entrainement fournies à l'algorithme d'apprentissage, les résultats varient pour les techniques appliquées à l'analyse de sentiments.

Il existe de nombreux algorithmes dans le domaine de l'apprentissage supervisé pouvant être adaptés à l'analyse des sentiments. Ces algorithmes incluent le *classificateur d'entropie maximale,* les *machines à vecteurs de support*, les *arbres de décision,* etc. Ils comprennent généralement trois majeures étapes : la collecte de données, le prétraitement, l'entrainement des données & le suivi des résultats [51].

5.1.1.1. Arbres de décision

Les arbres de décision font partie des méthodes d'apprentissage supervisé basées sur une approche géométrique pour segmenter l'espace des entités. Le principe de l'algorithme consiste à constituer une collection de classifieurs appelés classifieurs faibles pour des entités représentées dans le formalisme attribut/valeur, et étant introduits afin de segmenter l'espace des entités permettant ainsi de classer les données [52]. Ils sont utilisés en informatique décisionnelle pour l'exploration des données. Ils emploient une représentation hiérarchique des structures de données sous forme de séquences de décisions (test) pour prédire un résultat ou une classe. Chaque observation devant être affectée à une classe est décrite par un ensemble de variables testées dans les nœuds internes de l'arbre, tandis que les décisions sont prises dans les nœuds feuilles [52].

5.1.1.2. Machines à Vecteurs de Support

Les Machines à Vecteurs de Support (Support Vector Machines) sont une technique de classification linéaire d'apprentissage automatique supervisé qui utilise une fonction appelée *noyau* pour cartographier l'espace des entités en entrée dans un nouvel espace où les classes peuvent être séparées linéairement [53]. Le but de cet algorithme est de trouver un hyperplan dans un espace à N-dimensions (où N est le nombre d'entités), qui classe distinctement les points de données. Afin de séparer les deux catégories de points de données, de nombreux hyperplans possibles peuvent être sélectionnés [50]. L'objectif ici est de trouver un plan ayant la marge maximale, i.e. la

distance maximale qui sépare les deux classes de points (voir la figure ci-dessous). L'optimisation de cette marge de distance permet une meilleure classification des futurs points de données.

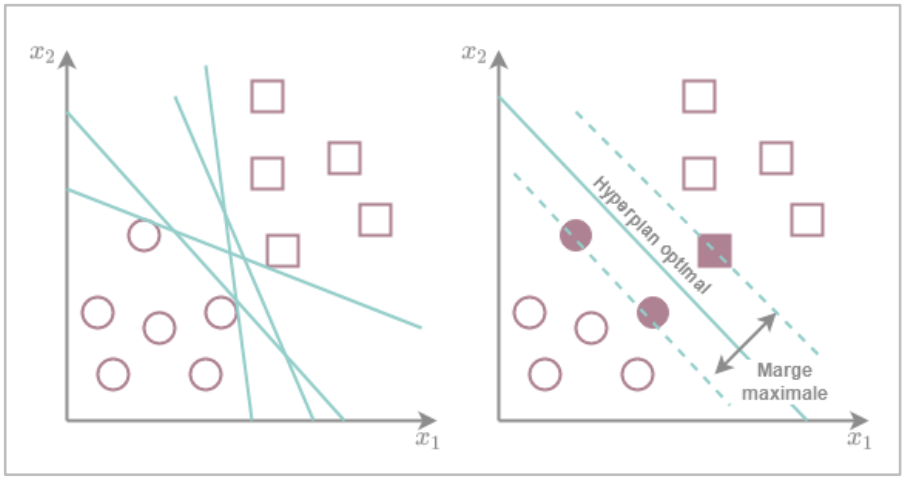

FIGURE 2.3 - Machines à Vecteurs de Support (hyperplans possibles).

Les performances des SVM dépendent de la fonction noyau utilisée : mieux elle est choisie et plus la classification est précise. De nombreuses méthodes pour définir cette fonction existent. Selon [54], la combinaison de noyaux appropriés peut améliorer la performance de la classification des sentiments et donc, une combinaison linéaire non négative de plusieurs noyaux représente une alternative.

5.1.1.3. Réseaux de Neurones

Un réseau de neurones imite la structure neurale du cerveau biologique, où le neurone représente l'unité de base. Il comprend une couche d'entrée, une couche cachée et une couche de sortie. Le neurone reçoit en entrée un vecteur exposant $X(i)$ qui dénote la fréquence du ième mot dans le texte. À chaque neurone est associé un poids A utilisé afin de calculer la fonction d'entrée qui est la fonction linéaire $f(i) = A.(X(i))$. Le signe de cette fonction de prédiction est pris en compte pour attribuer la classe étiquette [55].

La figure ci-dessous représente un réseau de neurones composé d'une couche d'entrée, une couche cachée et une couche de sortie :

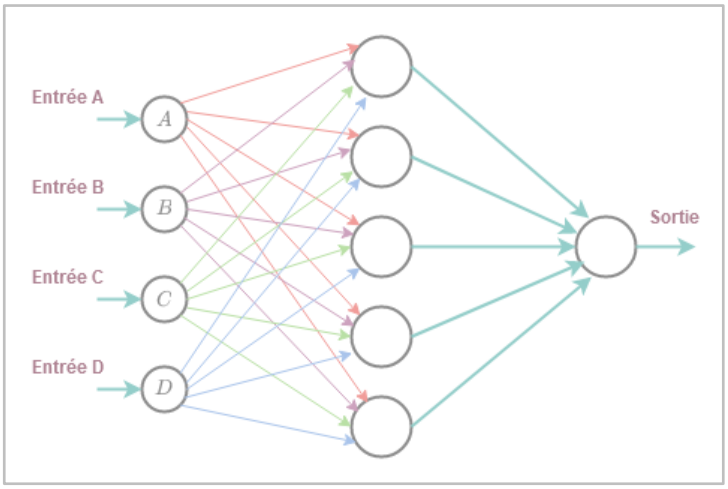

FIGURE 2.4 - Exemple de réseau de neurones.

L'apprentissage du réseau neural comprend deux étapes : la propagation directe (feed-forward) et la rétropropagation. Dans la propagation directe, l'entrée du neurone est multipliée par le poids qui est choisi aléatoirement au début du processus. Certaines fonctions sont utilisées pour normaliser la valeur de sortie entre 0 et 1. Cette valeur est par la suite comparée à la valeur cible pour calculer la valeur de l'erreur, puis une rétropropagation est effectuée. Durant la rétropropagation, les poids sont ajustés en multipliant la valeur d'entrée par la valeur d'erreur, de ce fait, l'apprentissage du réseau neural dépend de l'erreur [56].

5.1.1.4. Bayes Naïf

Il s'agit de l'un des algorithmes de classification probabilistes automatiques les plus simples qui ont été appliqués à la classification de texte, et par conséquent, il est pareillement utilisé pour résoudre les problèmes de l'analyse des sentiments. La particularité de cet algorithme repose sur le fait qu'il ne tient pas compte de l'emplacement du mot dans le texte, d'où son nom *« Bayes naïf »* [57]. Dans ce cadre algorithmique, les unités lexicales qui composent le corpus sont marquées avec une catégorie ou un ensemble de catégories spécifiques et traitées ensuite par calcul. Durant ce processus, chaque document est considéré comme un sac de mots[4] (Bag-of-Words), il est donc supposé que le document ne dispose pas de structure interne et qu'il n'existe aucune relation entre les mots qui le composent. Une fois que le

[4] **Sac de mots :** représentation simplifiée utilisée dans le Traitement Automatique du Langage et la Recherche d'Information, où le texte est représenté comme un sac de ses mots sans tenir compte de la grammaire ni même de l'ordre des mots, sauf que la multiplicité est maintenue.

traitement prend fin, un modèle de classification qui pourra être utilisé pour regrouper des documents invisibles est établi. Les étiquettes qui regroupent ces documents représentent des états émotionnels exprimés textuellement [57].

Le Bayes naïf est basé sur le théorème de Bayes pour calculer la probabilité de chaque terme avec son étiquette correspondante [57] :

$$P(label|entité) = \frac{P(label) * P(entités|label)}{P(entités)} \quad (2.1)$$

Où :

- $P(label)$ est la probabilité antérieure du label (étiquette) dans le dataset.
- $P(entités|label)$ est la probabilité antérieure d'un ensemble d'entités (mots) liée à une étiquette.
- $P(entités)$ est la probabilité antérieure qu'un ensemble d'entités se produise.

Étant donné l'hypothèse naïve qui stipule que tous les mots sont indépendants les uns des autres, l'équation peut être reformulée comme suit [57]:

$$P(label|entité) = \frac{P(label) * P(e_1|label) * \dots * P(e_n|label)}{P(entités)} \quad (2.2)$$

5.1.1.5. Réseaux Bayésiens

Un réseau bayésien est un formalisme graphique qui définit et simplifie une loi commune de probabilités d'un modèle. Il s'agit alors d'un modèle à la fois graphique et probabiliste. Les réseaux bayésiens permettent de trouver des relations entre un grand nombre de mots dans le contexte des problèmes de modélisation et d'apprentissage automatique et fournissent un outil approprié pour représenter ces relations. Un réseau bayésien consiste en un graphe orienté acyclique[5] dans lequel chaque nœud représente une variable aléatoire et les arcs reliant ces nœuds représentent une relation d'influence (i.e. le nœud prédécesseur influence son nœud successeur) qui sont modélisées à travers des distributions de probabilités conditionnelles [58].

Une table de probabilités conditionnelles est établie pour chaque nœud – variable – qui spécifie les probabilités de chaque état envisageable de la variable et déterminées

[5] **Graphe orienté acyclique :** un graphe orienté acyclique est un graphe qui ne comporte aucun circuit.

par les états des variables directement ascendantes. La figure ci-dessous représente un exemple de réseau bayésien simple (A : le texte contient le mot « chat », B : le texte contient le mot « amour », S : le texte exprime un sentiment positif) [58].

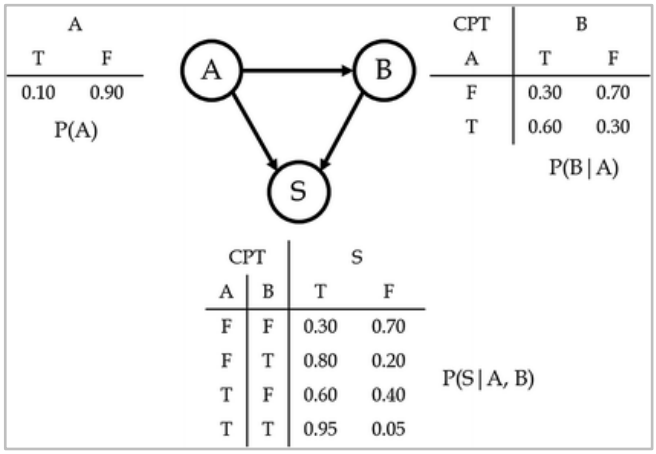

FIGURE 2.5 - Exemple d'un réseau bayésien simple.

5.1.1.6. Entropie Maximale

L'entropie maximale est un classifieur probabiliste d'apprentissage automatique connu sous le nom de classifieur exponentiel conditionnel qui utilise un encodage pour convertir un ensemble d'entités en vecteurs. Ces vecteurs encodés servent par la suite à calculer le poids de chaque entité pouvant ainsi être combinés pour déterminer l'étiquette la plus adéquate pour un ensemble de données. Ce classificateur est paramétré via un ensemble de $X\{poids\}$ utilisé pour associer les entités similaires générées à partir d'un ensemble par $X\{encodage\}$. Spécifiquement, cet encodage associe à chaque paire $\{(entité, label)\}$ un vecteur [50].

Le classifieur d'entropie maximale est une technique qui a fait ses preuves dans de multiples applications de traitement automatique du langage et de classification de texte. Elle ne construit pas d'hypothèse sur les relations entre les entités, et donc pourrait potentiellement être plus efficace lorsque l'hypothèse d'indépendance n'est pas satisfaite [50].

5.1.2. Apprentissage non supervisé

L'objectif principal de la classification de texte généralement et de l'analyse des sentiments spécifiquement est de classer des documents et des textes en un certain nombre de catégories prédéfinies. Pour ce faire, de larges datasets d'entrainement labélisés sont utilisés pour l'apprentissage supervisé. Dans la classification de texte, il

s'avère parfois difficile de créer des étiquettes pour ces données d'entrainement, mais il est plus facile de collecter des données sans étiquettes [59]. Des techniques d'apprentissage non supervisés peuvent surmonter ces difficultés en s'appuyant uniquement sur l'inférence statistique pour s'entrainer sur ces données au lieu de reposer sur un corpus déjà labélisé. A leur tour, ces techniques combinent des éléments lexicaux présentant des similitudes, fournissant ainsi une vue perspicace sur le corpus [48].

5.2. Approches basées sur le lexique

Dans la littérature de recherche, les mots de sentiments sont également appelés mots d'opinion, mots polaires ou mots porteurs d'opinion. Ils peuvent être divisés en deux types : le *type de base,* qui représente les adjectifs comme *beau, merveilleux, mauvais,* et *immonde,* et le *type comparatif* qui est utilisé pour exprimer des opinions comparatives et superlatives et qui comprend les mots comme *mieux*, *pire*, *moins* et *plus* [60]. Plusieurs approches ont été suggérées pour compiler les mots de sentiments, les deux principales approches sont : l'*approche par dictionnaire* et l'*approche par corpus* [60].

5.2.1. Approche basée sur le dictionnaire

Étant donné que la plupart des dictionnaires répertorient des synonymes et antonymes pour chaque mot, il est alors évident d'utiliser un dictionnaire pour compiler des mots de sentiments. Par conséquent, une technique simple dans cette approche consiste à commencer par certains mots-clés de sentiments basés sur la structure des synonymes et des antonymes d'un dictionnaire. Plus précisément, cette méthode fonctionne comme suit : tout d'abord, un ensemble restreint de mots de sentiments positifs et négatifs est collecté manuellement [61]. L'algorithme enrichit par la suite cette collection de mots en recherchant leurs synonymes et antonymes dans des dictionnaires disponibles en ligne comme WordNet [6], et rajoute ces mots nouvellement trouvés à la collection initiale pour que la prochaine itération puisse commencer. Lorsqu'aucun nouveau mot ne peut être trouvé, le processus prend fin pour laisser place à une étape d'inspection manuelle afin de nettoyer la liste obtenue [61].

[6] https://wordnet.princeton.edu/

5.2.2. Approche basée sur le corpus

Deux scénarios ont été proposés pour appliquer l'approche basée sur le corpus, le premier statistique et le deuxième sémantique : (i) étant donné une liste de départ de mots de sentiments connus (couramment utilisés à des fins générales), détecter à partir d'un corpus de domaine d'autres mots de sentiments et leurs orientations, et (ii) à travers un corpus de domaine, adapter un lexique global de sentiments à un nouveau lexique pour ce même domaine [60].

Bien qu'un lexique de sentiments généraux puisse être élaboré grâce à l'approche basée sur le corpus si un corpus très varié et très vaste est disponible, l'approche basée sur le dictionnaire reste plus performante puisqu'un dictionnaire contient un plus grand nombre de mots [60].

6. Défis de l'Analyse des Sentiments

Bien que l'annotation des sentiments en instances positives, négatives et neutres représente une tâche plutôt simple, des instructions claires et concises sont indispensables pour obtenir des annotations de haute qualité. En raison de nombreux facteurs, l'analyse des sentiments est confrontée à plusieurs défis, parmi eux nous citons [62] :

6.1. Détection du sarcasme et de l'ironie

Le sarcasme et l'ironie sont un phénomène courant et difficile à analyser pas uniquement par la machine mais souvent aussi par les humains [63]. Du point de vue de l'attribution d'une seule étiquette de sentiment, le sarcasme et l'ironie sont très délicats car ils peuvent souvent indiquer un sentiment positif de la part de l'orateur, même si ce dernier présente une attitude négative par rapport à quelqu'un ou quelque chose [63]. Le sarcasme et d'autres types de langages ironiques sont intrinsèquement problématiques pour qu'un algorithme puisse les détecter. Il les classe comme un sentiment modérément négatif ou extrêmement positif. La différence entre le sarcasme et l'ironie est encore plus compliquée. Une approche correctement affinée et rigoureuse sur un corpus suffisamment diversifié est nécessaire pour tenir compte du contexte pertinent [64]. Trois approches importantes ont été observées jusqu'à présent pour remédier à ce problème : l'extraction semi-supervisé de modèles pour identifier

les sentiments implicites, l'utilisation d'une supervision basée sur le hashtag et l'incorporation de contexte au-delà du texte cible [65].

6.2. Gestion de la négation

La gestion de la négation joue un rôle important dans l'analyse des sentiments lors de l'attribution d'une polarité à l'adjectif associé, et donc la polarité globale du texte. Les mots de négation incluent « *non* », « *ni* », « *pas* », etc. Par exemple, « *Ce plat est bon* » est classée comme une phrase positive, pendant que « *Ce plat n'est pas bon* » doit être classée comme négative [62]. Ce type de phrase peut être traité en inversant la polarité de l'adjectif qui apparaît après le mot négatif. Cependant, cette solution ne peut pas divertir les cas comme « *Pas surprenant que le plat soit aussi bon* » ou bien « *Non seulement que le plat était bon, mais il était aussi bien servi* ». L'utilisation de modèles mathématiques et de techniques de traitement automatique du langage n'a pas encore complètement résolu le problème de la négation [62]. Pour corriger ce problème, le concept de la portée d'un terme de négation est introduit. En utilisant un arbre d'analyse et des dépendances typées générées par un analyseur et des règles spéciales, une procédure pour identifier la portée de chaque terme de négation est fournie [66].

6.3. Désambiguïsation lexicale

L'ambigüité lexicale fait partie des premiers problèmes rencontrés par tout système reposant sur le traitement automatique u langage, qu'elle soit syntaxique ou sémantique. Le problème de l'ambiguïté syntaxique a été largement résolu par les Part-of-Speech taggers qui déterminent avec une grande précision la catégorie syntaxique d'un terme. Dans certains usages, le sens d'un mot ne peut être prédit qu'en examinant son contexte [67].

La désambiguïsation lexicale (*Word Sense Disambiguation* en anglais) est le processus d'identification du sens de mots appelés mots polysémiques, c'est-à-dire des mots ayant plusieurs sens [67]. Quelques travaux commencent par créer des lexiques dans lesquels chaque mot est associé à sa polarité antérieure en dehors du contexte. La polarité d'un mot dans une phrase et dans un contexte spécifique peut être différente de la polarité antérieure du mot, puisqu'un mot peut apparaître sous plusieurs significations. En addition, il peut s'avérer difficile de déterminer la polarité antérieure de mots comme *grand*, *petit*, *long*, *court*, etc. parce qu'ils ne disposent pas de polarité

propre à eux-mêmes [68]. Afin de déterminer le sens clair des mots, quatre tâches ont été suggérées [68] : (i) Déterminer les limites exactes du texte exprimant une opinion sur une caractéristique. (ii) Utiliser des méthodes appropriées pour identifier le contexte des mots dans les phrases. (iii) Fournir un mécanisme de correspondance de contexte pour obtenir la polarité du contexte associé à partir du lexique. (iv) Établir un lexique, qui contient non seulement la signification des mots dans un domaine spécifique, mais prend également en charge un mécanisme de correspondance de contexte.

6.4. Résolution de l'anaphore

La résolution d'anaphore ou la résolution de prénom désigne le problème de la résolution des références à des éléments antérieurs ou ultérieurs dans le texte. Ces éléments sont en général des expressions nominales, des phrases ou des paragraphes entiers qui représentent des objets du monde réel [69]. Lors de l'analyse des émotions, la plupart des chercheurs décident d'ignorer les pronoms. Il est difficile pour le système de reconnaître à quoi un pronom ou un nom fait référence dans le texte. Dans de nombreuses situations, les pronoms sont indispensables dans la compréhension de la perception de l'utilisateur [69]. Par exemple, « *Ce livre est génial, il est très inspirant* ». Ici, le pronom *il* fait référence au *livre*, on ne peut pas associer *inspirant* à *livre* sans connaître à quoi le pronom *il* fait référence. Une solution à ce problème a été proposée en utilisant la résolution de coréférence des sources. Seulement, elle utilise un clustering partiellement supervisé au lieu de simples algorithmes d'apprentissage supervisé [70]. Une approche d'apprentissage automatique supervisé avec deux fonctionnalités sémantiques a été proposée pour améliorer la précision de la résolution de coréférence [71].

7. Conclusion

Dans le présent chapitre, nous avons d'abord levé l'ambigüité par rapport à la différence entre l'analyse des sentiments et la fouille d'opinion, qui aujourd'hui, font partie du même domaine de recherche. Nous avons par la suite cité les différentes méthodes proposées dans la littérature pour déterminer la polarité et l'intensité d'un sentiment, à savoir l'orientation sémantique principalement. Ensuite, nous avons présenté les différents niveaux de granularité qui permettent d'effectuer une analyse

des sentiments. Le choix du niveau approprié dépend de la façon dont le sentiment est exprimé dans le texte. Nous avons également énuméré les nombreuses approches de l'analyse des sentiments qui existent dans la littérature. Ces approches sont divisées en deux catégories : les approches basées sur le Machine Learning et les approches basées sur le lexique. Enfin, nous avons présenté quelques-unes des difficultés rencontrées lors de l'analyse des sentiments et les solutions qui ont été proposées pour remédier à ces difficultés.

Dans le chapitre suivant, nous présenterons les différentes notions de l'apprentissage profond, à commencer par l'apprentissage automatique et ses différents types.

Apprentissage Profond

1. Introduction

L'apprentissage automatique ou en anglais, le Machine Learning est un sous-domaine de l'intelligence artificielle (IA) qui date du milieu du 20ème siècle et dont l'objectif est de concevoir des systèmes qui apprennent ou améliorent leurs performances en fonction des données utilisées. L'apprentissage automatique est donc une discipline scientifique centrée sur le développement, l'analyse et la mise en œuvre de méthodes automatisées qui donnent à la machine la capacité d'évoluer à travers un processus d'apprentissage sans intervention humaine ni assistance, et d'ajuster ses actions appropriées. Au cours des décennies suivantes, diverses techniques de machine learning ont été développées pour créer des algorithmes capables d'apprendre et de progresser de façon autonome. Parmi ces algorithmes, on retrouve les réseaux de neurones artificiels, sur lesquels est basé le Deep Learning ou l'apprentissage profond.

À travers ce chapitre, nous allons définir les notions fondamentales de l'apprentissage automatique ainsi que les techniques principales de l'apprentissage profond.

2. Apprentissage automatique

Selon la définition de l'informaticien et pionnier de l'apprentissage automatique Tom M. Mitchell, le domaine scientifique de l'apprentissage automatique (ML) est une branche de l'intelligence artificielle qui peut être définie comme suit [72]:

« L'apprentissage automatique est l'étude des algorithmes informatiques qui permettent aux programmes informatiques de s'améliorer automatiquement grâce à l'expérience. »

Par conséquent, l'apprentissage automatique permet au système d'apprendre à partir de données d'entrainement au lieu d'apprendre via une programmation explicite [73]. Cependant, l'apprentissage automatique n'est pas une tâche simple. Comme les algorithmes intègrent progressivement les données d'entraînement, il devient possible de construire des modèles plus précis à partir de ces données [73]. Un modèle d'apprentissage automatique est le produit conçu lors de l'apprentissage d'un

algorithme de machine learning avec des données. Une fois la formation terminée, une sortie peut être obtenue lorsque l'on fournit de nouvelles entrées au modèle [73].

Des techniques d'apprentissage automatique sont indispensables pour optimiser la précision des modèles prédictifs. Il existe différentes méthodes qui varient en fonction de la nature du problème à résoudre, de son type et de la quantité de données [74]. Dans les sous-sections suivantes, nous couvrirons les différentes catégories de l'apprentissage automatique.

2.1. Apprentissage supervisé

Les algorithmes d'apprentissage automatique supervisé peuvent utiliser des exemples étiquetés pour appliquer les connaissances acquises dans le passé à de nouvelles données afin de prédire les événements futurs [75]. En d'autres termes, l'apprentissage supervisé débute habituellement par une collection de données d'entrainement bien définie sous forme de couples entrée-sortie $(x_n, y_n)_{1 \leq n \leq N}$ avec $x_n \in X$ et $y_n \in Y$, pour comprendre la manière dont les données sont classées. L'objectif de l'apprentissage supervisé est de découvrir des modèles au sein des données et de les appliquer au processus analytique. Ces données incluent des caractéristiques x_n liées à leurs étiquettes y_n qui définissent leur signification [75].

Les problèmes d'apprentissage supervisé peuvent être divisés en problèmes de régression et de classification [76]. Ces deux problèmes visent à construire un modèle concis qui peut prédire la valeur d'un attribut associé en fonction des variables de l'attribut. La différence entre ces deux tâches réside dans la nature de l'attribut dépendant : elle est numérique et continue pour la régression, catégorique et discrète pour la classification [76].

2.2. Apprentissage non supervisé

En revanche, des algorithmes d'apprentissage automatique non supervisé sont utilisés dans le cas où les informations utilisées pour l'entrainement ne sont ni annotées ni classées [77]. Afin de comprendre la signification de ces données, il est impératif d'utiliser des algorithmes qui répartissent les données en fonction des clusters ou des tendances qu'ils repèrent. L'apprentissage non supervisé conduit un processus itératif, où les données peuvent être analysées sans intervention manuelle [78]. L'apprentissage non supervisé explore donc la manière dont les systèmes peuvent déduire une fonction

décrivant la structure latente à partir de données non labelisées. Le système ne trouve pas de sortie valide, mais examine les données et peut tirer des conclusions à partir d'ensembles de données pour décrire des structures cachées à partir de données non annotées [77].

2.3. Apprentissage semi-supervisé

Les algorithmes d'apprentissage automatique semi-supervisé se situent entre l'apprentissage supervisé et non supervisé. Ils utilisent parallèlement des données annotées et non annotées pour l'apprentissage - majoritairement une grande quantité de données non étiquetées et une petite quantité de données étiquetées [79]. Les systèmes ayant recours à cette méthode peuvent perfectionner prodigieusement la précision de l'apprentissage. En général, l'apprentissage semi-supervisé est sélectionné lorsque les données labellisées obtenues nécessitent des ressources efficaces et appropriées pour leur formation et apprentissage [79]. Sinon, l'acquisition de données non labellisées ne nécessite généralement aucune ressource additionnelle. Un autre avantage de ce type d'apprentissage vient du fait que l'étiquetage des données nécessite une intervention humaine de l'utilisateur [80]. Cela peut être fastidieux lorsque l'ensemble de données devient très volumineux. Dans ce cas, l'apprentissage semi-supervisé avec seulement quelques étiquettes revêt une signification pratique évidente [80].

2.4. Apprentissage par renforcement

La différence entre l'apprentissage par renforcement et les autres types d'apprentissage est qu'il n'a pas besoin de présentation de des paires d'entrée-sortie étiquetées, ni de correction explicite des actions sous-optimales [81]. L'objectif est de trouver un équilibre entre l'exploration (territoire inconnu) et l'utilisation (connaissances actuelles) en apprenant plutôt par le biais d'une méthode d'essais et d'erreurs [82]. Par conséquent, une série de décisions fructueuses conduira au renforcement du processus, car c'est la solution la plus efficace au problème immédiat. Une simple récompense est nécessaire pour que l'agent sache quelle action est la meilleure à entreprendre. C'est ce qu'on appelle un signal de renforcement [83].

3. Apprentissage profond

L'apprentissage en profondeur est un algorithme abstrait de haut niveau qui permet de modéliser des données à partir d'une grande quantité de données apprises [84]. Contrairement à l'apprentissage automatique traditionnel, l'apprentissage en profondeur cherche à imiter la façon dont notre cerveau extrait et traite les informations. Ceci est effectué en créant des réseaux neuronaux artificiels pouvant extraire des relations et des concepts complexes à partir d'un ensemble de données [84].

3.1. Réseaux de neurones artificiels

Un réseau de neurones artificiel est un système d'apprentissage supervisé composé d'un grand nombre d'éléments communément appelés neurones ou perceptrons. Chacun de ces neurones est capable de prendre des décisions simples et les transmettre aux autres neurones qui sont organisés en couches interconnectées [85]. Ensemble, le réseau de neurones peut émuler n'importe quelle fonction et résoudre pratiquement toutes les tâches, avec un nombre suffisant d'échantillons d'entrainement et de puissance de calcul. Un réseau de neurones « peu profond » ne comprend que trois couches de neurones [85] :

- **Une couche d'entrée :** qui lit les valeurs de données à partir de l'entrée fournie par l'utilisateur.
- **Une couche cachée :** qui est l'endroit où se déroule la majorité de l'apprentissage.
- **Une couche de sortie :** qui génère les prédictions qui sont le résultat du réseau.

Ces différentes couches sont représentées dans l'architecture illustrée dans la figure ci-dessous :

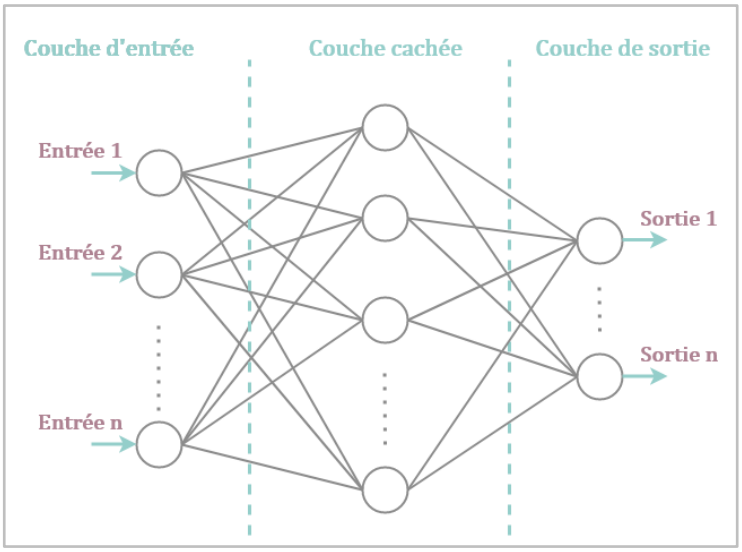

FIGURE 3.1 - Architecture d'un réseau de neurones artificiel.

Chaque perceptron ou neurone représente un algorithme de classification binaire qui simule le fonctionnement des neurones biologiques [86]. Bien que le perceptron ait une structure assez simple, comme il est illustré dans la figure ci-dessous, il a la capacité d'apprendre à résoudre des problèmes très complexes. Par conséquent, le neurone est l'unité de base de calcul du réseau neural [86].

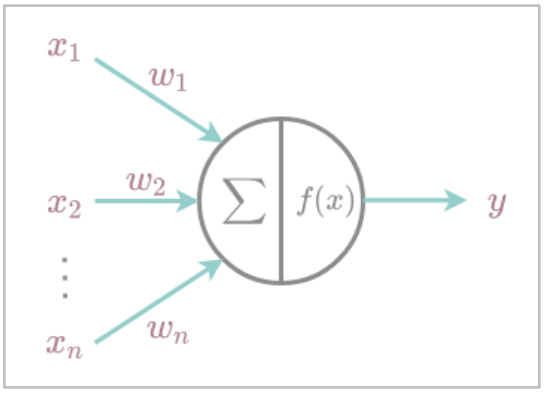

FIGURE 3.2 - Architecture d'un perceptron.

Son principe général est de renvoyer des informations d'entrée multiples. Les informations entrantes peuvent par exemple provenir des informations de sortie d'autres neurones dans le cadre d'un réseau. Plus précisément, les informations entrantes sont notées $x_1, \dots, x_n \in \mathbb{R}$, où pour chaque $i \in \{1, \dots, n\}$ est associé un poids $w_i \in \mathbb{R}$ à x_i , en addition à la valeur de biais notée x_0 associée au poids w_0 [86]. L'information traitée par le neurone est donc la moyenne pondérée calculée selon la formule suivantes [86]:

$$\bar{x} = \sum_{i=0}^{n} w_i x_i = \sum_{i=1}^{n} w_i x_i - w_0 \tag{3.1}$$

Le perceptron a pour rôle de fournir une réponse y comprise entre 0 et 1 à partir de $\bar{x}$. Pour ce faire, une fonction $g: \mathbb{R} \rightarrow [0,1]$, dite *fonction d'activation* est utilisée. Si le résultat est proche de 1, alors le neurone sera *actif*, si le résultat est proche de 0 alors le neurone sera considéré comme *inactif*. Habituellement, la fonction d'activation utilisée est une fonction non linéaire comme la fonction sigmoïde ou la tangente hyperbolique, définies respectivement par les équations suivantes [86]:

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} = 2 \times sigmoid(2x) - 1 \tag{3.3}$$

3.2. Réseaux de neurones profonds

Un réseau de neurones profond a une structure similaire à celle d'un réseau de neurones artificiel, à l'exception d'avoir plusieurs couches cachées entre les couches d'entrée et de sortie [85]. Bien que les réseaux de neurones peu profonds puissent résoudre des problèmes assez complexes, les réseaux de neurones profonds se sont montrés plus précis à mesure que des couches de neurones cachées sont rajoutées [85]. Ces couches cachées additionnelles sont efficaces jusqu'à une limite de neuf à dix couches, au-delà de cette limite, le pouvoir prédictif du réseau commence à diminuer. Aujourd'hui, la majorité des modèles et implémentations de réseaux de neurones profonds se limitent à trois jusqu'à dix couches cachées [87]. Ceci est représenté dans la figure suivante :

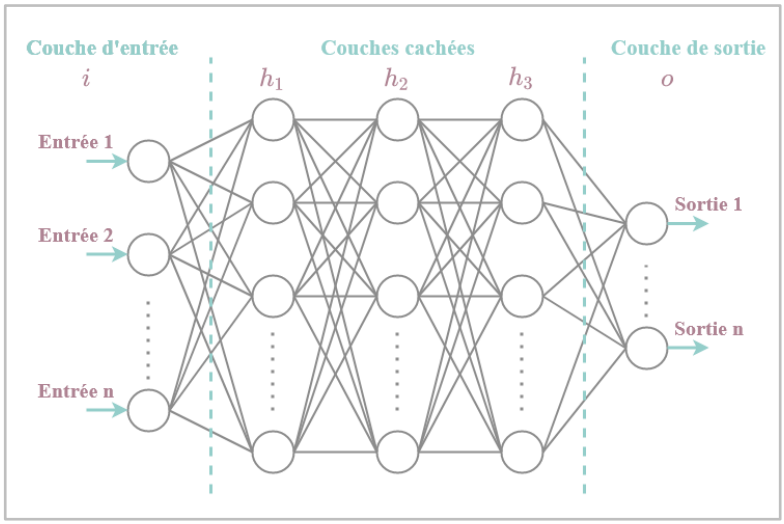

FIGURE 3.3 - Architecture d'un réseau de neurones profond.

De façon générale, les réseaux de neurones profonds sont des réseaux à propagation avant, où les données sont transmises de la couche d'entrée vers la couche de sortie sans rebouclage.

3.3. Propagation avant et rétropropagation

L'entrainement d'un réseau de neurones implique l'apprentissage des valeurs des paramètres (poids et biais), et représente la partie la plus authentique de l'apprentissage profond. Ce processus d'apprentissage peut être considéré comme un processus itératif *qui va et qui revient* en traversant les couches du réseau neural [88]. Le chemin de l'entrée vers la sortie est appelé « *propagation avant* », tandis que le chemin inverse, c'est-à-dire de la sortie vers l'entrée est appelé « *rétropropagation* » [88].

La phase initiale de la propagation vers l'avant a lieu lorsque le réseau est alimenté par les données d'apprentissage et que celles-ci traversent l'ensemble du réseau pour calculer les prédictions [89]. En d'autres termes, les données d'entrée sont transmises à travers le réseau de façon à ce que l'ensemble des neurones ajustent les informations qu'ils reçoivent de la couche précédente et les diffusent aux neurones de la couche suivante. Lorsque les données ont parcouru toutes les couches et que tous les neurones ont exécuté leurs calculs, la couche de sortie sera atteinte avec la prédiction calculée pour l'ensemble des données d'apprentissage passé en entrée [89].

Par la suite, une *fonction de coût* sera utilisée pour estimer l'erreur (ou la perte) et pour comparer et évaluer la précision de la prédiction par rapport au résultat correct [90]. Idéalement, le coût devrait être nul, c'est-à-dire qu'il ne devrait y avoir aucune

divergence entre la valeur estimée et la valeur prédite. Ainsi, lors de l'entrainement du modèle, les poids des connexions neurales seront graduellement accommodés jusqu'à l'obtention de la bonne prédiction [91]. La fonction de coût est donc une valeur unique, généralement mesurée par la différence moyenne entre le résultat estimé et le résultat attendu. Le choix de cette fonction dépend des poids du réseau et de ses biais ainsi que de la nature des données d'apprentissage. Parmi les fonctions de coût couramment utilisées, on retrouve [91]:

- **L'Erreur Quadratique Moyenne :** également appelée « *écart moyen quadratique* », représente la différence quadratique moyenne entre les valeurs estimées et la valeur réelle. Elle est calculée comme suit :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{3.4}$$

  Où n est le nombre d'entrées des données d'apprentissage, Y_i est la valeur estimée pour l'entrée i et $\hat{Y}_i$ est la valeur réelle pour l'entrée i.

- **La fonction d'erreur d'entropie croisée :** mesure la performance d'un modèle de classification dont la sortie est une valeur de probabilité comprise entre 0 et 1. Elle est calculée selon l'équation suivante :

$$H(p,q) = -\sum_{\forall x} p(x)\log(q(x)) \tag{3.5}$$

  Où $p(x)$ est la distribution réelle et $q(x)$ est la distribution estimée, définies sur une variable discrète x.

Une fois la perte calculée grâce à la fonction de coût adéquate, les données sont propagées de la couche de sortie vers les couches cachées, d'où son nom : rétropropagation. Néanmoins, sur la base de la contribution de chaque neurone des couches cachées à la sortie du réseau, ces neurones ne reçoivent qu'un fragment du signal global de perte [90]. Ce processus est itéré couche par couche jusqu'à ce que l'ensemble des neurones du réseau reçoit un signal de perte qui décrit leur contribution relative à la perte globale. La figure suivante résume ces étapes d'apprentissage :

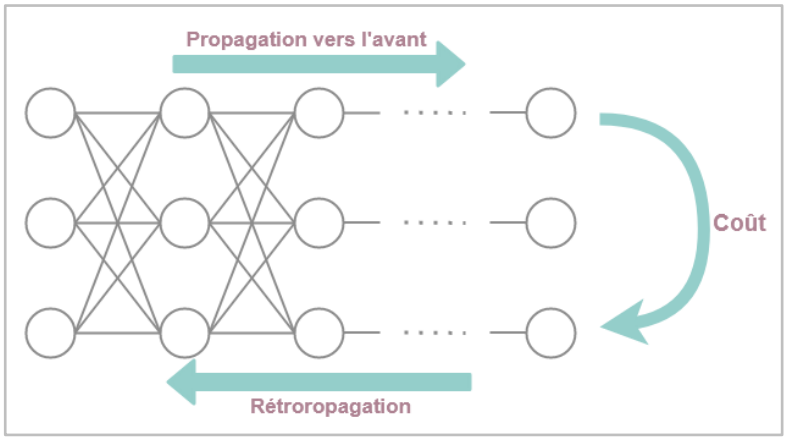

FIGURE 3.4 - Processus d'apprentissage du réseau de neurones.

3.4. La Descente de Gradient

Une fois les informations propagées à travers le réseau grâce à la rétropropagation, les poids des connexions entre les neurones peuvent être ajustés. Le but ici est de minimiser la valeur de la perte pour la rendre aussi proche que possible de zéro pour la prochaine utilisation du réseau. Pour cela, un algorithme d'optimisation appelé « Descente de Gradient » permettant de trouver le minimum de n'importe quelle fonction convexe[7] est utilisé [92]. Cette technique utilise le calcul de la dérivée (ou gradient) de la fonction de perte pour ajuster les poids par petits incréments, de sorte qu'on puisse visualiser dans quelle direction « descendre » vers le minimum total [92]. Cela est généralement réalisé par lots de données (batches) dans des itérations successives (époques) de l'ensemble de données d'apprentissage transmis au réseau à chaque itération [93]. La figure qui suit illustre le processus de la Descente de Gradient, qui sera expliqué par la suite :

[7] **Fonction convexe :** une fonction f, définie, dérivable (donc continue) sur un intervalle I est convexe sur I si sa représentation graphique est entièrement située au-dessus de chacune de ses tangentes.

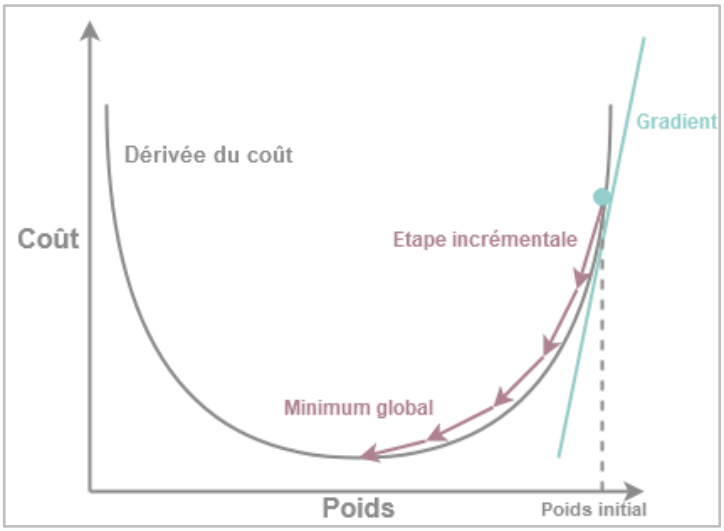

FIGURE 3.5 - Descente de Gradient.

Lors de la mise à jour des paramètres, la descente du gradient utilise la première dérivée de la fonction de perte qui est le gradient (il fournit la pente d'une fonction à un point donné). Ce processus consiste à relier les dérivées de la perte de chacune des couches cachées aux dérivées de la perte de sa couche supérieure, tout en intégrant sa fonction d'activation dans le calcul [93]. C'est pour cette raison que les fonctions d'activation doivent être dérivables. Dans chaque itération, une fois la valeur du gradient de la fonction de perte correspondante à chaque neurone attribuée, les valeurs des paramètres sont mises à jour dans le sens opposé de celui indiqué par le gradient. En effet, le gradient pointe toujours vers la direction dans laquelle la valeur de la fonction de coût augmente [93]. C'est pour cela que la direction dans laquelle la fonction de perte a tendance à diminuer est obtenue en utilisant le négatif du gradient.

3.5. Réseaux de neurones convolutifs

Les réseaux de neurones convolutifs (également baptisés *réseaux de neurones à convolution, CNN* ou encore *ConvNets*) sont une forme particulière de réseaux de neurones artificiels multicouches conçus pour traiter des données d'images bidimensionnelles bien qu'ils puissent être utilisés avec des données unidimensionnelles et tridimensionnelles [94]. L'architecture des connexions neurales de ces réseaux s'inspire de la structure du cortex visuel des mammifères [94]. Ils sont composés de plusieurs couches de neurones combinées avec des fonctions mathématiques à plusieurs paramètres ajustables pouvant prétraiter une quantité d'informations restreinte, comme le montre la figure suivante :

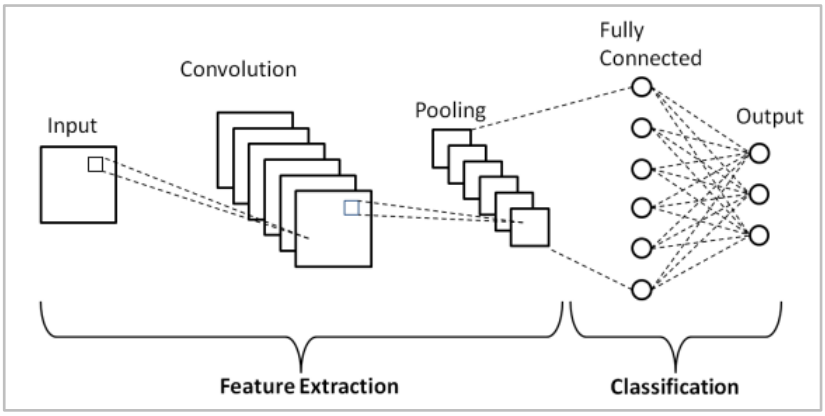

FIGURE 3.6 - Réseau de neurones convolutif.

Un réseau de neurones convolutif est caractérisé par une première couche dite « convolutionnelle » (généralement une à trois premières couches). Cette couche, comme son nom l'indique, est basée sur le principe mathématique de la convolution et tente d'identifier la présence d'un motif ou caractéristiques (par exemple dans une image ou un signal) [95]. La convolution est une opération linéaire qui consiste simplement à appliquer un filtre, qui est une matrice bidimensionnelle de poids, à l'entrée du réseau, ce qui entraîne l'activation [95]. L'application répétée du même filtre à l'entrée produit une carte d'activations appelée *cartes des caractéristiques*, qui montre la position et l'intensité des caractéristiques détectées dans l'entrées. Le filtre utilisé a une dimension plus petite que celle des données d'entrée. Ce filtre est multiplié via un produit scalaire par un bloc de l'entrée ayant la même tailles que le filtre [94]. Le but de l'utilisation d'un filtre plus petit que l'entrée est parce qu'il permet au même filtre (matrice de poids) d'être multiplié par la matrice d'entrée (l'image) plusieurs fois à différents emplacements. Plus précisément, le filtre est systématiquement appliqué à chaque partie ou bloc des données d'entrés de la taille du filtre, de gauche à droite et de haut en bas. Une fois la carte des caractéristiques créée, chacune de ses valeurs peut alors être transférée par non-linéarité, comme la fonction ReLU dont l'équation est la suivante, où pour tout réel x [94] :

$$f(x) = \max(0, x) \tag{3.6}$$

L'application systématique d'un même filtre aux données d'entrée s'est avérée efficace : si le filtre est conçu pour détecter un certain type de caractéristiques sur l'entrée, l'application cohérente de ce filtre à l'ensemble de l'image d'entrée permet alors au filtre de détecter cette caractéristique où qu'elle soit sur l'image [94].

3.6. Réseaux de neurones récurrents

Les réseaux de neurones récurrents, également connus sous le nom de RNNs (pour *Recurrent Neural Networks* en anglais), sont un type de réseaux de neurones permettant aux prédictions antérieures d'être utilisées comme entrées via des états cachés (en anglais *hidden states*) [96]. Ceci est réalisé en propageant l'information dans les deux sens, y compris de couches cachées à la couche d'entrée. C'est ce mécanisme qui les rapproche du vrai fonctionnement du système nerveux, qui n'est pas à sens unique et qui les distingue des autres types de réseaux de neurones grâce à l'utilisation de boucles de rétraction pour traiter une séquence de données qui forme le résultat final, qui lui-même peut être une séquence de données [96]. Ces boucles de rétroaction rendent l'information durable, effet généralement équivalent à la mémoire. La figure ci-dessous illustre une représentation d'un RNN :

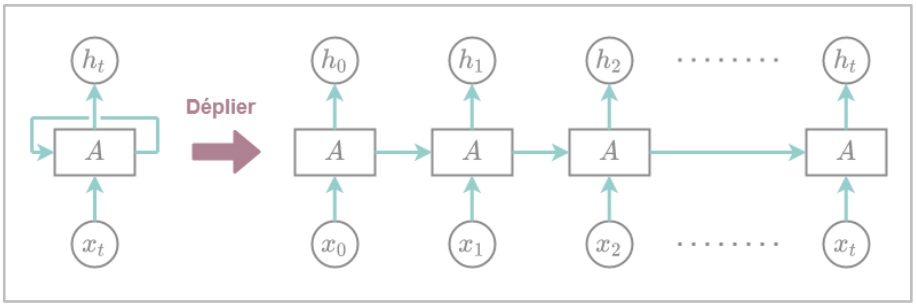

FIGURE 3.7 - Réseau de neurones récurrent.

Le bloc étiqueté A est un simple réseau de neurones à propagation directe déjà familier. Le côté droit de la figure montre le réseau A à chaque pas de temps t, c'est-à-dire à $t = 0$, l'entrée x_0 est assimilée par le réseau pour générer la sortie h_0, l'entrée du pas de temps suivant est donc x_1. Seulement, il y a une entrée additionnelle du pas de temps précédent à partir du bloc A. Ainsi, le réseau neural prend en compte non seulement l'entrée actuelle, mais dispose aussi du contexte des entrées précédentes [96]. La figure suivante représente un échantillon du réseau neural récurrent, c'est-à-dire une unité unique du RNN :

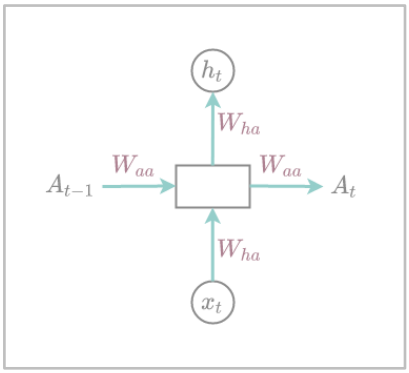

FIGURE 3.8 - Unité d'un réseau de neurones récurrent.

Les formules régissant le calcul de la sortie d'une unité du RNN à l'instant t sont les suivants [96]:

$$A_t = g(W_{ax}x_t + W_{aa}A_{t-1} + b_a) \quad (3.7)$$

$$h_t = g(W_{ya}A_t + b_y) \quad (3.8)$$

Où :

- A est la sortie de la couche cachée.
- $g(\,)$ est la fonction d'activation.
- W est la matrice de poids.
- x est l'entrée.
- h est la sortie à un pas de temps t.
- b est le biais.

Pour le RNN, la disparition du gradient est un problème majeur, puisque les réseaux de neurones qui ne peuvent plus être entrainés convenablement perdront inévitablement leurs performances. Cette disparition se produit dans les réseaux de neurones multicouches utilisés pour traiter des données complexes [97]. Ajouter efficacement ces paramètres dès les premières couches nécessite beaucoup de temps et de ressources. Une façon de résoudre ce problème consiste en l'utilisations d'unités *LSTM* (*Long Short-Term Memory*) à mémoire court terme étendue. Les réseaux de neurones récurrents dotés d'unités LSTM sont capables de classer les données dans des cellules de mémoire à long terme ou à court terme. De cette manière, ils peuvent faire la distinction entre les données importantes devant être mémorisées et réinjectées dans le réseau et les données devant être « oubliées » [97].

4. Réseaux de neurones récurrents modernes

Bien que les réseaux de neurones récurrents soient largement utilisés, ils ne sont pas suffisamment efficaces pour résoudre les problèmes actuels d'apprentissage de séquences. Par exemple, compte tenu de l'instabilité numérique lors du calcul du gradient, les réseaux de neurones modernes sont utilisés beaucoup plus souvent dans la pratique [97].

4.1. Long Short-Term Memory

Les cellules *Long Short-Term Memory* sont conçues pour surmonter le problème de la disparition du gradient (en anglais, *Vanishing Gradient*) dans les réseaux de neurones récurrents et leur permettre de conserver les informations plus longtemps par rapport aux RNNs traditionnels [98]. Les LSTM ont la capacité de maintenir une erreur constante qui leur permet de continuer l'apprentissage sur de nombreux pas de temps et se propager à travers les couches [98].

Les LSTM disposent d'une structure en chaine qui contient quatre réseaux de neurones et différents blocs de mémoires appelés *cellules*. Les informations sont sauvegardées par les cellules et les manipulations de la mémoire sont effectuées par ce qu'on appelle des « *portes* », qui sont des mécanismes internes permettant de réguler le flux d'information, comme nous pouvons le constater dans l'architecture de la cellule LSTM représentée dans la figure ci-dessous [99]:

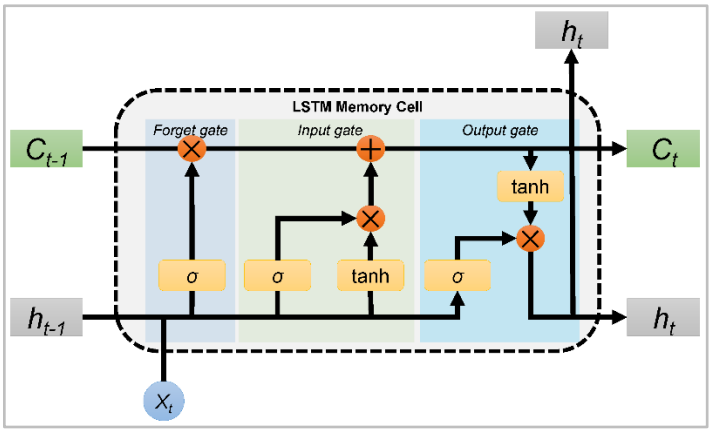

FIGURE 3.9 - Architecture d'une cellule LSTM.

Ces portes ont la capacité de déterminer quelles données de la séquence d'entrée sont importantes à conserver ou à éliminer. Ce faisant, les informations pertinentes

sont transmises le long de la chaine de séquences pour effectuer des prédictions. Chacune de ces portes est discutée ci-dessous [99]:

- **La porte d'oubli :** les informations n'étant plus utiles pour l'état de la cellule sont éliminées par la porte d'oubli. Deux entrées x_t (entrée à l'instant t) et x_{t-1} (sortie de la cellule précédente) sont transmises à la porte et multipliées par les matrices de poids suivies par l'ajout du biais. Le résultat de cette opération est passé à une fonction d'activation qui produit un résultat binaire. Si pour un état de cellule donné la sortie est 0, l'information est alors oubliée, dans le cas contraire, c'est-à-dire 1, l'information se voit donc conservée pour une utilisation future.
- **La porte d'entrée :** l'ajout d'informations pertinentes à l'état de la cellule est réalisé grâce à la porte d'entrée. Les informations sont régulées via une fonction *sigmoïde* dans un premier temps et la porte filtre les valeurs devant être retenues de la même façon que la porte d'oubli en utilisant les paramètres h_{t-1} et x_t. Par la suite, la fonction *tangente hyperbolique*, dont la sortie varie de -1 à +1, est utilisée pour construire un vecteur qui inclut toutes les valeurs potentielles de h_{t-1} et x_t. Pour finir, les valeurs de ce vecteur sont multipliées par les valeurs régulées afin d'obtenir les informations à conserver.
- **La porte de sortie :** la porte de sortie est chargée d'extraire les informations utiles à partir de l'état actuel de la cellule et à la présenter en tant que sortie. Pour commencer, un vecteur est généré grâce à l'application de la fonction *tangente hyperbolique* sur la cellule. Ensuite, les informations sont régulées grâce à la fonction sigmoïde pour filtrer les valeurs à conserver à l'aide des entrées h_{t-1} et x_t. A la fin, les valeurs du vecteur sont multipliées par les valeurs régulées pour constituer la sortie qui représente en même temps l'entrée de la prochaine cellule.

4.2. Modèle Encodeur-Décodeur

L'architecture encodeur – décodeur est un modèle de conception de réseaux de neurones qui calcule directement la probabilité conditionnelle d'une séquence de sortie à partir d'une séquence d'entrée donnée sans supposer un alignement constant, c'est-

à-dire $P(y_1, \dots, y_O | x_1, \dots, x_I)$ où les longueurs d'entrée et de sortie, respectivement I et O, peuvent être distinctes [100], comme le montre la figure ci-dessous :

FIGURE 3.10 - Architecture du modèle encodeur-décodeur.

Comme le montre la figure ci-dessous, l'architecture est divisée en deux parties : l'encodeur et le décodeur, qui sont généralement des réseaux de neurones récurrents. L'encodeur a pour rôle d'encapsuler les informations passées en entrée en des vecteurs d'état interne appelés « *tensors* » [100]. Les sorties de l'encodeur sont alors rejetées et seuls les états internes sont conservés. Ces derniers seront par la suite transmis au décodeur pour générer la sortie. Dans la traduction automatique par exemple, l'encodeur convertit une phrase source telle que « Bonjour tout le monde » en un état qui est un vecteur capturant ses informations sémantiques [101]. Par la suite, le décodeur exploite cet état pour générer la phrase cible traduite, par exemple « Hello world ».

4.3. Modèle Séquence à Séquence

Le modèle *Séquence à Séquence* ou *Seq2Seq* est basé sur l'architecture de l'encodeur – décodeur pour générer une séquence de sortie de longueur fixe pour une séquence d'entrée de longueur fixe, où la longueur de l'entrée et de la sortie peut différer [102]. Dans ce contexte, la séquence est une liste de symboles correspondants aux mots d'une phrase. L'idée est d'utiliser un réseau de neurones récurent pour chacun de l'encodeur et du décodeur, qui fonctionnent ensemble traiter les séquences. Cependant, comme la version traditionnelle des RNN souffre du problème de la disparition du gradient, une version plus avancée est utilisée pour remédier à ce problème, à savoir les LSTM et GRU. Le modèle Seq2Seq est constitué de trois parties [102]: un encodeur, un vecteur intermédiaire (ou encodeur) et un décodeur, comme le montre la figure ci-dessous :

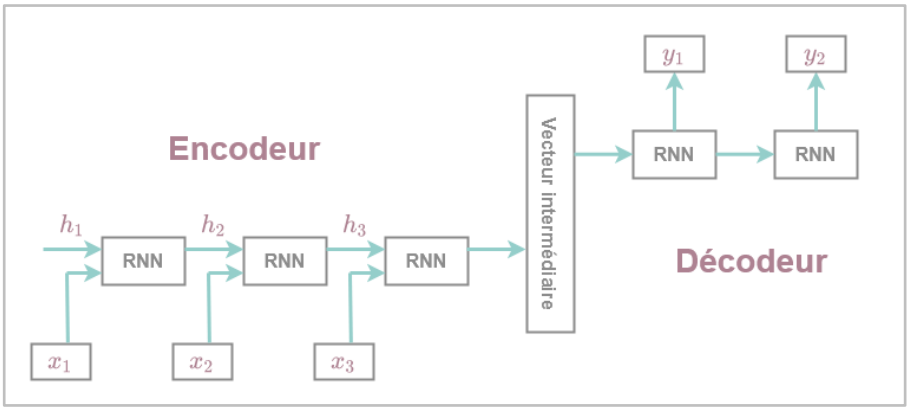

FIGURE 3.11 - Architecture du modèle séquence à séquence.

- **L'encodeur :** il consiste en une pile de plusieurs unités récurrentes (généralement des LSTM ou GRU pour des performances optimales), chacune acceptant un élément unique de la séquence d'entrée, rassemble des informations sur cet élément et les propage vers l'avant. Dans le cas des systèmes de dialogue par exemple, la séquence d'entrée représente l'ensemble de tous les mots formant la question. Chaque mot est représenté par x_i, où i et la position du mot dans la séquence. Les états cachés ou internes sont calculés selon la formule suivante :

$$h_t = f\left(W^{(hh)} h_{t-1} + W^{(hx)} x_t\right) \qquad (3.9)$$

  Cette formule simple exprime le résultat d'un réseau neural récurrent ordinaire où, les poids appropriés à l'état interne précédent h_{t-1} et au vecteur d'entrée x_t sont appliqués.

- **Le vecteur intermédiaire :** il s'agit de l'état caché final généré par la partie encodeur du modèle Seq2Seq. Il est également calculé à l'aide de la formule ci-dessous. Ce vecteur est destiné à encapsuler les informations de toute la séquence d'entrée pour aider le décodeur à effectuer des prédictions précises. Ce vecteur sert alors d'état caché initial pour la partie décodeur.
- **Le décodeur :** il est constitué d'une pile de plusieurs unités récurrentes, chacune d'elles prédit un résultat y_t à l'instant t. Chaque unité assimile l'état caché de l'unité qui la précède pour produire son propre état caché. Dans le cas des systèmes de dialogue vu précédemment, la séquence de sortie n'est autre que l'ensemble des mots qui composent la réponse.

Chaque mot de la séquence est représenté par y_i, où i est la position du mot dans la séquence. La formule suivante permet de calculer tout état caché h_i :

$$h_t = f\left(W^{(hh)} h_{t-1}\right) \tag{3.10}$$

Comme on peut le constater, l'état caché précédant est utilisé pour calculer le suivant. La sortie y_t à l'instant t est à son tour calculée à l'aide de l'équation ci-dessous :

$$y_t = softmax(W^s h_t) \tag{3.11}$$

Les sorties sont calculées sur la base de l'état caché à l'instant actuel avec le poids approprié $W(s)$. La fonction Softmax est utilisée pour construire un vecteur de probabilité permettant de déterminer la sortie finale.

4.4. Mécanisme d'attention

Le concept de l'attention a été initialement introduit comme solution pour résoudre le problème majeur entourant le modèle Séquence à Séquence, où il a remporté un grand succès. Etant donné que seul le dernier état caché de l'encodeur est utilisé comme vecteur intermédiaire pour le décodeur, le modèle Seq2Seq basique se trouve souvent incapable de gérer avec précision de longues séquences d'entrées [103]. Par ailleurs, le mécanisme d'attention permet de résoudre directement ce problème car il préserve et utilise tous les états cachés de la séquence d'entrée durant l'étape de décodage. A cette fin, il crée une cartographie unique pour chaque instant t de la sortie du décodeur vers tous les états cachés de l'encodeur [103]. Cela veut dire que pour chaque sortie effectuée par le décodeur, ce dernier a accès à l'ensemble de la séquence d'entrée et peut désigner sélectivement des éléments particuliers à partir de la séquence afin de générer la sortie. De ce fait, ce mécanisme permet au modèle de se concentrer et d'accorder plus d'« attention » aux éléments pertinents de la séquence d'entrée si nécessaire [103].

L'attention est donc une méthode de *pooling* généralisée avec un alignement de biais sur les entrées. L'élément au cœur du mécanisme d'attention est la *couche d'attention*, ou appelée simplement *attention* [104]. A chaque instant j dans le décodeur, à défaut d'utiliser un contexte fixe (l'état caché final de l'encodeur), un vecteur de contexte différent C_i est employé pour générer le mot y_i [104]. Ce vecteur intermédiaire C_i consiste en la somme pondérée des états cachés de l'encodeur, il est calculé comme suit :

$$C_i = \sum_{j=1}^{n} \alpha_{ij} h_j \tag{3.12}$$

Où n est la longueur de la séquence d'entrée, h_j est l'état caché à l'instant j et α_{ij} est calculée selon la formule suivante :

$$\alpha_{ij} = \exp(e_{ij}) \Big/ \sum_{k=1}^{n} \exp(e_{ik}) \tag{3.13}$$

Tel que e_{ij} est le modèle d'alignement selon l'état caché antérieur du décodeur s_{i-1} et le $j$ème état caché de l'encodeur. Ce modèle d'alignement est configuré comme un réseau neural à propagation avant qui est conjointement entrainé avec le reste du modèle. La figure ci-dessous représente une illustration graphique du mécanisme d'attention [103]:

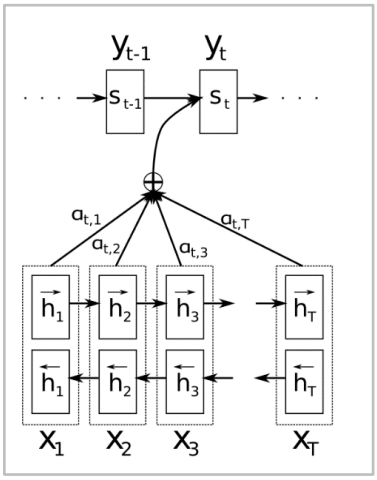

FIGURE 3.12 - Représentation du mécanisme d'attention.

Chaque état caché de l'encodeur code des informations sur le contexte local dans une partie donnée de la séquence. Au fur et à mesure que les données se propagent du mot 0 au mot n, ces informations de contexte local s'estompent, ce qui oblige le décodeur à atteindre un pic au niveau de l'encodeur afin de connaitre les contextes locaux. Diverses parties de la séquence d'entrée disposent d'informations essentielles pour produire les différentes parties de la sortie. Autrement dit, chaque mot de la séquence de sortie est aligné sur différents mots de la séquence d'entrée. Le modèle d'alignement fournit une mesure de correspondance entre l'élément i de la sortie et les entrées vers la position j. En partant sur cette base, la somme pondérée des états cachés est utilisée pour générer chaque mot de la séquence de sortie [103].

5. Conclusion

L'apprentissage automatique est l'étude de techniques puissantes qui peuvent apprendre de l'expérience. Étant donné que les algorithmes d'apprentissage automatique accumulent généralement plus d'expérience sous la forme d'observation de données ou d'interaction avec l'environnement, ses performances s'améliorent. L'apprentissage profond quant à lui est une fonction d'intelligence artificielle (IA) qui imite le fonctionnement du cerveau humain lors du traitement des données pour la création de modèles de prise de décision. L'apprentissage profond est aussi un sous-ensemble de l'apprentissage automatique, qui utilise des réseaux pouvant être entrainés selon un apprentissage non supervisé à partir de données non structurées ou non étiquetées.

Dans le présent chapitre, nous avons élaboré une présentation de l'apprentissage machine et ses différentes catégories, ainsi que l'apprentissage profond dont nous avons énuméré les différentes techniques.

Dans le chapitre suivant, nous entamerons la conception et la modélisation de notre système en expliquant les contraintes rencontrées lors de ce processus et en détaillant chacune des techniques utilisées.

Conception et Modélisation

1. Introduction

Après avoir présenté une vue globale sur l'existant dans le domaine de l'Analyse des sentiments et le domaine des agents conversationnels, le moment est venu de passer à la phase créative du projet qu'est la conception. Le but premier de la conception est de fixer les choix techniques qui permettent de créer un système répondant à un besoin en tenant compte des contraintes. Cette conception doit servir de support pour l'implémentation et la maintenance du système. La modélisation du système est réalisée en s'appuyant sur les connaissances acquises à travers les chapitres précédents.

Dans ce chapitre, nous allons définir la problématique initiale de l'analyse des sentiments dans les conversations humain-agent, pour ensuite présenter et expliquer la modélisation de notre solution proposée.

2. Problématique et rappel des objectifs

Le service client représente l'assistance ou le conseil fourni aux clients lors de leur interaction avec la marque ou l'entreprise. Il s'agit en général d'un coût lié aux besoins du client et, dans la majorité des cas, cela n'arrive qu'après que le client ait pris une décision d'achat. Les agents de service clientèle suivent certaines procédures et peinent à répondre aux contraintes techniques et administratives. L'intelligence artificielle redéfinit catégoriquement le domaine du service client. Nulle part cette révolution fondamentale de l'expérience client n'est plus évidente que dans la nouvelle vague d'agents conversationnels. Les chatbots basés sur l'apprentissage automatique sont devenus largement disponibles et sont capables de résoudre automatiquement les requêtes sans aucune intervention humaine.

Comme cela a été décrit dans l'état de l'art, plusieurs méthodes ont été exploitées dans le domaine de l'analyse des sentiments : méthodes basées sur le Machine Learning et méthodes basées sur le lexique. Dans le contexte d'un système de détection de sentiments appliqué aux conversations humain-agent, différents aspects doivent être pris en considération pour sélectionner la méthode la plus appropriée. Tout

d'abord, il est indispensable de s'interroger sur le niveau de précision de l'analyse produite. Cela est lié à ce que nous avons décrit précédemment concernant la distinction nécessaire entre les différents niveaux de granularité de l'expression des sentiments. Sur un plan plus technique, le choix de l'approche de l'analyse des sentiments doit être basé sur une connaissance des ressources disponibles. Par conséquent, dans le cadre des méthodes d'apprentissage automatique supervisé, un corpus annoté est essentiel pour les phases d'apprentissage et de test. Cependant, à notre humble connaissance, il n'existe aucun corpus conversationnel comprenant des annotations émotionnelles telles qu'elles sont exprimées dans le contenu verbal. Toutes ces contraintes ont dû être prises en compte lors de l'élaboration de notre système.

3. Architecture du système

Afin de répondre aux différentes problématiques que nous avons expliqué dans la section précédente, nous avons choisi de diviser notre système de chatbot, que l'on va nommer *« iBuy »*, en deux principaux modules. Le premier module est dédié à la classification du sentiment de la requête fournie par l'utilisateur que l'on va nommer « *système conversationnel* ». Le deuxième module qui sera à son tour nommé « *classifieur de sentiments* », vise à répondre à cette requête. Ces deux sous-systèmes sont illustrés dans la figure ci-dessous, qui représente l'architecture globale du système. Pour chacun des deux modules, un corpus différent a été utilisé pour entrainer et tester les modèles, car comme mentionné précédemment, il est impossible de trouver un corpus conversationnel doté d'annotations de sentiments.

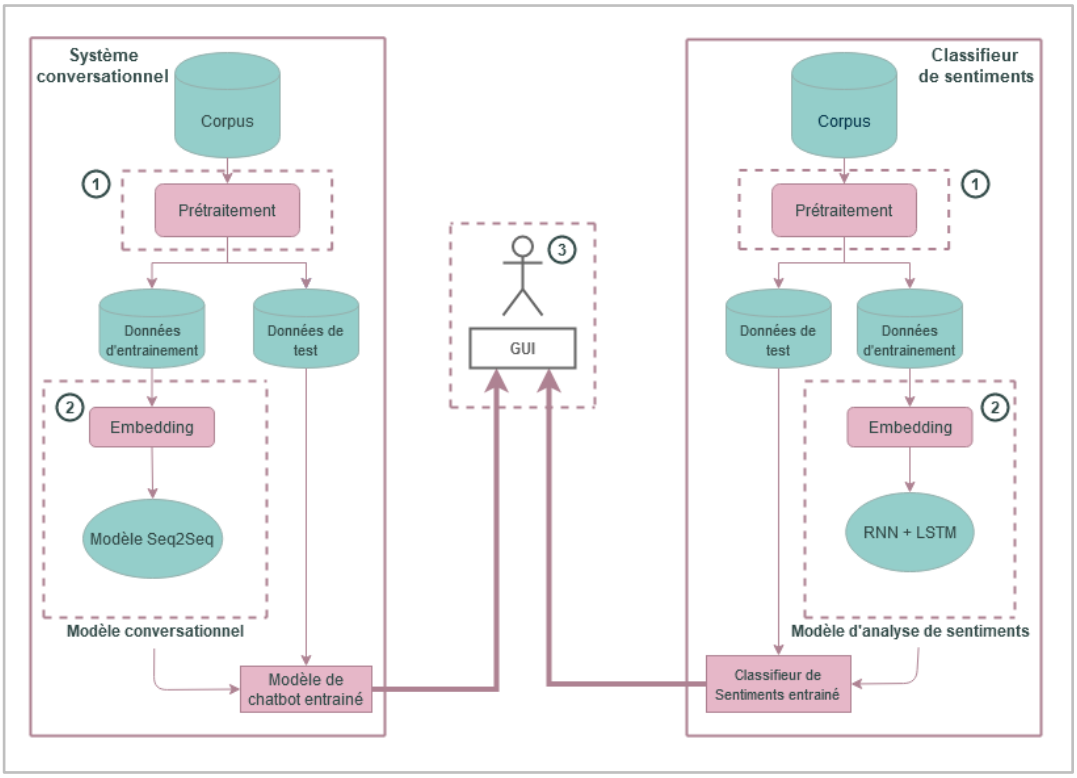

FIGURE 4.1 - Architecture globale du système.

Dans le but de créer ces deux sous-systèmes, nous avons procédé en plusieurs étapes. Nous avons cartographié l'ensemble du processus pour illustrer la mise en œuvre de notre solution à commencer par l'acquisition des corpus, jusqu'au déploiement du système sur une interface graphique pour interagir avec l'utilisateur final. Les étapes principales de la construction peuvent être énumérées à travers les tâches de *prétraitement*, d'*apprentissage* ou de *traitement* et de *post-traitement* (respectivement *1*, *2* et *3* dans la figure ci-dessus). Ces étapes seront détaillées dans les sections qui suivent.

3.1. Prétraitement

Pour l'étape de prétraitement des données, le processus est le même pour chacun de nos deux modules. Chaque séquence des deux corpus passe par les mêmes étapes de prétraitement. Ces étapes sont illustrées dans la figure suivante :

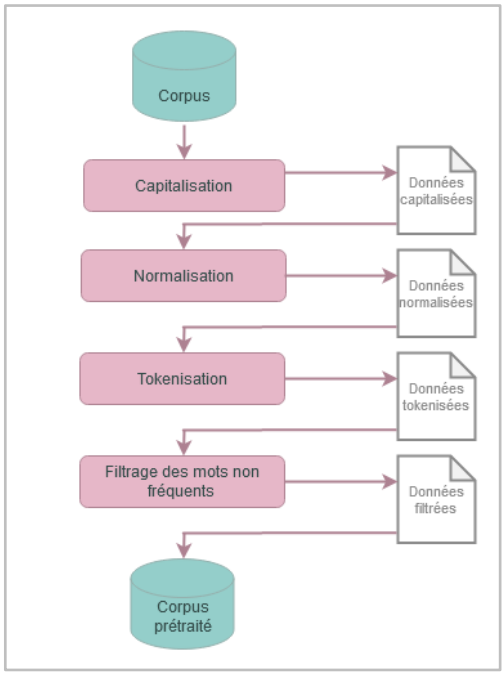

FIGURE 4.2 - Etapes de prétraitement.

3.1.1. Capitalisation

Bien que souvent négligée, la *capitalisation* ou la mise en minuscule, ou en anglais *lower-casing,* est l'une des formes les plus simples et les plus efficaces de prétraitement de texte. Elle convient à la plupart des problèmes d'exploration de texte (Text *Mining*) et de NLP et contribue de manière significative à la cohérence de la sortie attendue.

La capitalisation consiste à réduire toutes les lettres en minuscule. C'est souvent une bonne pratique : cela va permettre aux instances comme « *Produit* » au début d'une phrase de correspondre à la requête de « *produit* », car deux mots comme ceux-ci ayant la même signification, s'ils ne sont pas convertis en minuscules alors ils constitueront des mots non similaires dans le modèle d'espace vectoriel. D'autre part, une telle pratique pourrait assimiler des mots ayant un sens complètement différent. De nombreux noms propres sont dérivés de noms génériques et ne sont alors distingués qu'en fonction de circonstances spécifiques. Ces mots comprennent par exemple des noms de personne (Black, black), des noms d'entreprise (General Motors), etc.

3.1.2. Normalisation

Avant de poursuivre le prétraitement du corpus, ce dernier doit être normalisé. La *normalisation* fait généralement référence à un ensemble de tâches connexes visant à

mettre tout le texte dans un même format ; c'est-à-dire convertir le texte en une seule forme canonique. Normaliser le corpus avant de le traiter garantit la cohérence de l'entrée avant que toute opération ne soit effectuée. Cependant, la normalisation nécessite de connaitre le type de texte devant être normalisé et comment le traiter. Dans notre cas d'étude, nos deux corpus sont en langue anglaise. Parmi les opérations incluses dans le processus de la normalisation, nous retrouvons :

- L'élimination des espaces blancs dupliqués.
- La suppression de la ponctuation et des caractères spéciaux.
- La substitution de contractions (très fréquent en Anglais, par exemple : « I'm » → « I am »).
- La conversion des nombres en mots pour ne garder que les caractères alphabétiques.

3.1.3. Tokenisation

La tokenisation est une tâche de NLP qui consiste à diviser un bout de texte en unités plus petites nommées « *tokens* ». Un token est une instance d'une séquence de caractères dans un document spécifique qui sont combinés ensemble comme une entité sémantique pouvant s'avérer utile pour le traitement. Qu'il s'agisse de diviser un paragraphe en phrases, une phrase en mots ou un mot en caractères, ces tokens sont soit des phrases, des mots ou bien des caractères. Ainsi, la tokenisation peut être généralement divisée en trois catégories : la tokenisation de phrases, de mots et de caractères (caractères de n-gramme). Dans notre cas, nous utiliserons la tokenisation de phrases en mots, comme illustré dans la figure suivante :

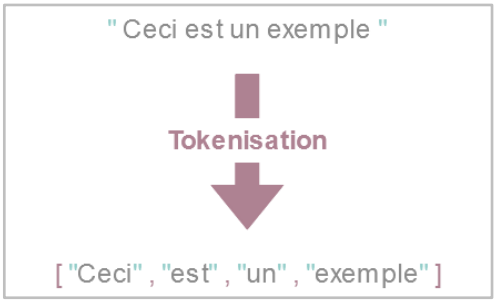

FIGURE 4.3 - Exemple de Tokenisation.

3.1.4. Filtrage des mots non fréquents

Afin d'optimiser le traitement des données, il est important d'éliminer les mots non fréquents. Ceci est effectué en créant un vocabulaire qui associe chaque mot du corpus

à son nombre d'occurrences. Ce dernier est calculé en utilisant ce qu'on appelle « *Frenquency Distribution* », qui indique la fréquence de chaque élément du vocabulaire dans le texte. Il s'agit d'une « distribution » car elle montre comment le nombre total de tokens est réparti entre les différents éléments du vocabulaire. Ensuite, un certain seuil doit être désigné pour pouvoir éliminer les mots qui apparaissent le moins dans le corpus. Dans notre cas, nous avons choisi de filtrer 5% des mots non fréquents. Ainsi, le dictionnaire final ne contiendra que les tokens ayant un nombre d'occurrences supérieur à ce seuil.

3.2. Traitement

Le processus d'apprentissage comporte différentes étapes. Tout d'abord, les données prétraitées sont divisées en deux différentes parties : les données d'entrainement et les données de test. Les données d'entrainement sont par la suite transformées en vecteurs de nombres réels, décrits dans un modèle vectoriel et facilitant ainsi leur interprétation par la machine. Les vecteurs de mots résultants sont introduits dans le modèle Deep Learning de chacun de nos deux sous-systèmes. Ainsi, la réponse correspondant à la question sera générée pour le cas du système conversationnel et le sentiment de la requête de l'utilisateur sera déterminé pour le cas du système d'analyse de sentiments.

3.2.1. Padding et Bucketing

Avant de commencer l'entrainement, il est nécessaire de savoir que dans le cas de problèmes de prédiction des séquences de longueur variable, les données doivent être transformées de sorte que chaque séquence ait la même longueur. De ce fait nous utilisons le « remplissage » ou « Padding » en anglais pour convertir les séquences de longueur variable à partir du corpus en séquences de longueur fixe. Nous utilisons les symboles spéciaux suivants pour compléter chaque séquence :

- **SOS** : Start Of String (début de la séquence).
- **EOS** : End Of String (fin de la séquence).
- **PAD** : remplisseur.
- **OUT** : mot pas dans le vocabulaire, correspond aux 5% des mots qui ont été filtrés auparavant.

Considérons la paire requête-réponse suivante :

Question: *how are you*
Réponse: *i am fine*

Supposons que nous souhaitons que la longueur de nos séquences soit fixée à 10, alors cette paire sera convertie de la manière suivante :

Question: *[PAD, PAD, PAD, PAD, PAD, PAD, PAD, « you », « are », « how »]*
Réponse: *[SOS, « i », « am », « fine », EOS, PAD, PAD, PAD, PAD, PAD]*

L'introduction du remplissage résout le problème des séquences de longueur variable, mais le cas des séquences de taille importante doit être considéré. Si par exemple la plus longue phrase de nos deux datasets est de longueur 100, alors toutes les autres phrases devront être encodées sur une longueur de 100 éléments afin qu'aucun mot ne soit perdu. A titre d'illustration, dans la phrase « how are you », la séquence encodée comptera 97 symboles « PAD ». Cela éclipsera les informations pertinentes de la phrase.

En plaçant les phrases dans des buckets de différentes tailles, le « *Bucketing* » résout en quelque sorte ce problème. Considérons la liste de compartiments suivante : « (5 ,10), (10 ,15), (15 ,20), (20,30) ». Si une requête et sa réponse sont de longueur 3 (comme indiqué dans l'exemple précédent), alors nous plaçons la phrase dans le bucket (5,10). La requête sera donc de longueur 5 et sa réponse de longueur 10. Pendant l'exécution du modèle (apprentissage ou test), nous utilisons un modèle différent pour chaque bucket, compatible avec la longueur de la requête et sa réponse. Tous les modèles ont les mêmes paramètres et fonctionnements alors de façon identique. Si nous utilisons par exemple le bucket (5,10), nos séquences seront encodées comme suit :

Question: *[PAD, PAD, « you », « are », « how »]*
Réponse: *[SOS, « i », « am », « fine », Eos, PAD, PAD, PAD, PAD, PAD]*

3.2.2. Word Embedding

Pour chacun de nos deux modèles, les données prétraitées sont au préalable transformées en ce qu'on appelle *Word Embeddings*, qui va servir à compresser l'espace des entités passées en entrée en une dimension plus petite. Par exemple, supposons que nous disposons de 5 000 mots dans notre cas d'étude et que nous décidons de prétraiter ces données pour ensuite établir une matrice qui comporte les termes du corpus. Cette matrice sera clairsemée et la séquence *[« natural », « language », « processing »]* aura alors un vecteur d'une dimension de 5 000, où toutes les valeurs sont à zéro sauf les trois éléments correspondant à ces mots. Dans ce cas, cette matrice sera transmise comme entrée au modèle, et celui-ci devra calculer le poids de chaque entité individuellement (50 000 au total). Comme on peut le constater, cette approche est gourmande en mémoire.

Pour remédier à ce problème, nous utiliserons le « *prolongement de mots* » ou « *Word Embeddings* » en anglais. Le Word Embedding est le terme collectif désignant un ensemble de techniques de modélisation dans le traitement automatique du langage, dans lequel les mots et les phrases du vocabulaire sont représentés sous forme de vecteurs réels. Abstraitement, il s'agit d'une intégration mathématique d'un espace de plusieurs dimensions par mot à un espace vectoriel continu de dimension beaucoup plus petite. Chaque mot est représenté par un vecteur de valeurs réelles, généralement de dizaines ou de centaines de dimensions. Cela contraste avec les milliers ou millions de dimensions nécessaires pour la représentation de mots épars, comme l'encodage *one – hot*[8]. La représentation distribuée est apprise de l'utilisation des mots. En conséquence, ces mots utilisés de manière similaire ont des représentations équivalentes, capturant naturellement leur signification. Ainsi, le Word Embedding fournit une représentation plus dense des mots et leurs significations relatives. Il constitue une amélioration par rapport à la représentation clairsemée utilisée dans les modèles *bag – of – words*.

Les méthodes de Word Embedding apprennent une représentation vectorielle de valeur réelle pour un vocabulaire de taille fixe prédéfini à partir d'un corpus donné. Ce processus d'apprentissage est soit rattaché à un modèle de réseau neural, soit à un

[8] **Encodage one-hot :** Un encodage one-hot ou encodage 1 parmi n consiste à représenter des états en utilisant pour chacun une valeur dont la représentation binaire n'a qu'un seul chiffre 1.

processus non supervisé utilisant des statistiques de document. Dans notre cas, nous utilisons l'algorithme *Global Vectors for Word Representation*, ou *GloVe* développé par Pennigton et al. [105], qui est une extension de la méthode *Word2Vec* pour un apprentissage efficace des vecteurs de mots. Les représentations basiques des mots dans un modèle d'espace vectoriel ont été développées en utilisant de techniques de factorisation matricielle comme *l'Analyse Sémantique Latente*[9] (LSA) qui se sont avérés efficaces en faisant appel à des statistiques textuelles globales, mais ne sont pas aussi performantes que les méthodes apprises telles que Word2Vec pour saisir le sens et le démontrer dans des tâches comme le calcul d'analogie.

GloVe est donc une méthode qui combine des statistiques globales de techniques de factorisation matricielle comme LSA avec un apprentissage local basé sur le contexte dans Word2Vec. GloVe n'utilise pas de fenêtres pour définir ce contexte local, mais utilise plutôt les informations statistiques de l'ensemble du corpus pour construire un contexte de mots explicites ou une matrice de co-occurrences de mots. Le résultat est un modèle d'apprentissage qui peut conduire à une meilleure fusion de mots.

3.2.3. Algorithme d'apprentissage

La Descente de Gradient est l'un des algorithmes les plus connus pour effectuer l'optimisation et est de loin la méthode la plus courante pour optimiser les réseaux neuraux. Elle convient aux scénarios où la fonction est facile à distinguer des paramètres utilisés dans le réseau. Il est plus accessible de minimiser des fonctions continues au lieu de fonctions discrètes. Cette technique produit des résultats souvent satisfaisants ; mais si la taille de l'ensemble de données d'apprentissage devient importante, alors la technique se détériorera et ne convergera pas bien. Elle peut également ne pas converger vers un minimum global s'il existe plusieurs minimums locaux.

La méthode de Descente de Gradient Stochastique surmonte cette lacune en prenant au hasard des échantillons de données et en mettant à jour les paramètres selon la fonction de coût. De plus, sa vitesse de convergence est plus rapide que celle de la

[9] **Analyse sémantique latente :** méthode statistique qui identifie les composants sémantiques sous-jacents des mots. Elle repose sur l'idée que les mots d'un corpus de textes apparaissant dans les mêmes contextes linguistiques ont des significations similaires.

descente de gradient classique, et il n'accumule pas de poids intermédiaires, économisant ainsi la mémoire.

Pour l'apprentissage de chacun de nos deux modèles, nous avons opté pour *Adam* (*Adaptive Moment Estimation*) qui est un algorithme d'optimisation qui peut remplacer la descente de gradient stochastique pour mettre à jour les poids itératifs du réseau en fonction de l'ensemble d'apprentissage [106]. Adam calcule ainsi les taux d'apprentissage adaptatif (learning rates en anglais) de chaque paramètre. En plus de conserver une moyenne exponentiellement décroissante des gradients q_t , Adam maintient également une moyenne en décroissance exponentielle des gradients précédents p_t, semblable à une impulsion. Bien que l'impulsion puisse être considérée comme une balle descendant une pente, Adam agit comme une balle lourde avec friction qui a tendance à préférer les minimums plats sur la surface d'erreur.

L'algorithme d'Adam est brièvement mentionné ci-dessous [106]:

Tant que $\boldsymbol{w_t}$ ***ne converge pas*** *faire :*

Calculer le gradient $\boldsymbol{g_t = \frac{\partial f(x,w)}{\partial w}}$

Calculer $\boldsymbol{p_t = m_1 . p_{t-1} + (1 - m_1) . g_t}$

Calculer $\boldsymbol{q_t = m_2 . q_{t-1} + (1 - m_2) . g_t}$

Calculer $\boldsymbol{\hat{p}_t = \frac{p_t}{(1 - m_1^t)}}$

Calculer $\boldsymbol{\hat{q}_t = \frac{q_t}{(1 - m_2^t)}}$

Mettre à jour le paramètre $\boldsymbol{w_t = w_{t-1} - \alpha . \frac{\hat{p}_t}{\sqrt{\hat{q}_t} + \epsilon}}$

Retourner $\boldsymbol{w_t}$

Où w_t est le poids à mettre à jour, p_t et q_t sont respectivement les estimations du premier moment (la moyenne) et du second moment (la variance non centrée) des gradients, et $\hat{p}_t$ et $\hat{q}_t$ sont les estimations des premier et deuxième moments corrigées par le biais.

3.2.4. Fonction de coût et fonction d'activation

La fonction de perte est utilisée pour optimiser les valeurs des paramètres du réseau de neurones. Cette fonction cartographie un ensemble de valeurs de paramètres du

réseau à une valeur scalaire qui définit dans quelle mesure ces paramètres sont efficaces pour accomplir la tâche dont le réseau est chargé. La fonction de perte que nous avons choisi d'appliquer à nos deux modèles de réseaux neuraux des deux sous-systèmes est la fonction d'*entropie croisée* que nous avons défini dans le chapitre précédent. L'entropie croisée est une mesure de différence entre deux distributions de probabilité $p(x)$ et $q(x)$ pour une variable aléatoire ou un ensemble d'événements donné x. Elle est définie par la formule suivante :

$$H(p,q) = -\sum_{\forall x} p(x)\log(q(x)) \qquad (4.1)$$

L'entropie est le nombre de bits requis pour transférer un événement choisi aléatoirement à partir d'une distribution de probabilité. Les distributions asymétriques ont une entropie plus faible, tandis que les distributions avec la même probabilité d'événements ont une entropie plus élevée. L'entropie croisée est donc basée sur le concept de l'entropie et calcule le nombre de bits nécessaires pour représenter ou diffuser un événement moyen d'une distribution par rapport à une autre.

La fonction d'activation est une formule mathématique qui détermine la sortie d'un réseau de neurones. Cette fonction est reliée à chaque neurone du réseau et définit s'il doit être activé en fonction du fait que l'entrée de chaque neurone soit pertinente pour la prédiction du modèle. Nous avons choisi d'appliquer la fonction d'activation *Softmax* sur la dernière couche de chacun de nos deux modèles de réseaux de neurones. La fonction Softmax, également connue sous le nom de *fonction exponentielle normalisée,* prend en entrée un vecteur $z = \{z_1, \dots, z_n\}$ de n nombres réels et le normalise en une distribution probabiliste composée de n probabilités proportionnelles aux exponentielles du vecteur d'entrée. La distribution de probabilité signifie que le vecteur résultat $\sigma(z)$ ne dépasse pas un total de 1. Il est évident que si certaines composantes du vecteur d'entrée sont négatives ou supérieures à 1, elles seront alors dans la plage (0,1) après l'application de Softmax, dont l'équation est la suivante :

$$\sigma(z) = \frac{e^{zj}}{\sum_{i=1}^{n} e^{zi}} \quad \text{pour tout } j \in \{1, \dots, n\} \qquad (4.2)$$

3.2.5. Fonctionnement du système conversationnel

Dans ce qui suit, nous allons expliquer l'architecture neuronale choisie pour notre système conversationnel ainsi que les étapes du processus suivi pour la génération de réponse.

3.2.5.1. Architecture neuronale du système conversationnel

Comme nous l'avons mentionné précédemment, un agent conversationnel orienté tâches repose sur les techniques de Traitement Automatique du Langage Naturel et de Deep Learning. Par conséquent, nous avons opté de construire l'architecture de notre système conversationnel en se basant sur le modèle *Sequence - To – Sequence* avec des cellules *LSTM*. Ce modèle a été introduit dans l'apprentissage de représentation de séquences à l'aide de l'encodeur-décodeur RNN ; il est depuis lors devenu le modèle de référence pour les systèmes de dialogue et la traduction automatique. Notre modèle est également doté d'un *mécanisme d'attention* qui lui permet de se focaliser sur certaines parties de l'entrée pour générer la sortie précise. Notre modèle est schématisé dans la figure ci-dessous :

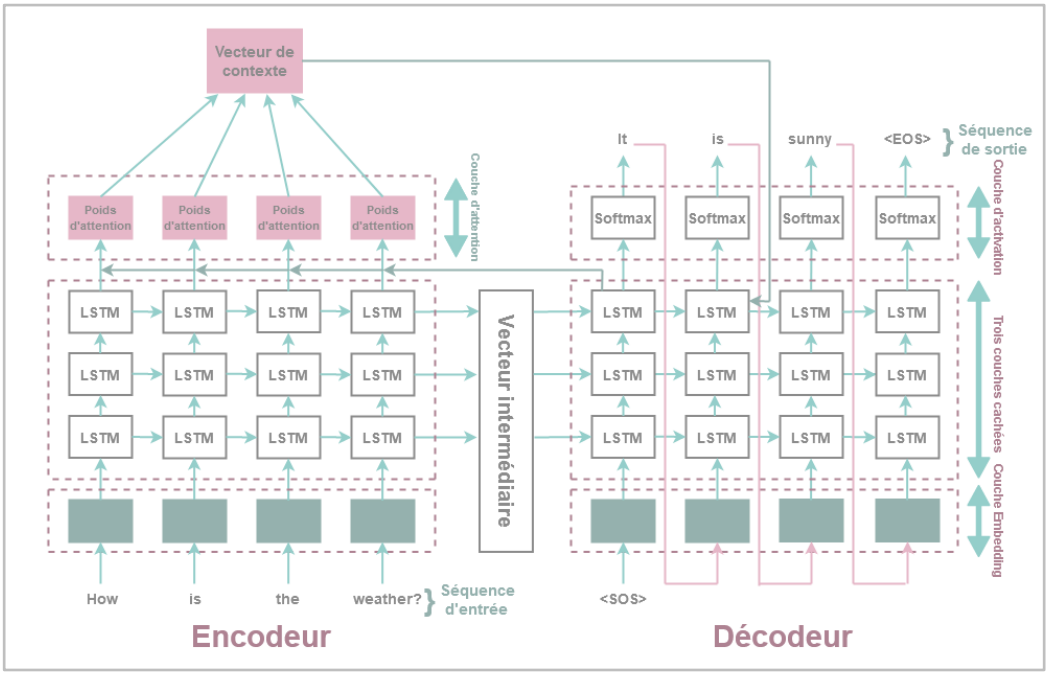

FIGURE 4.4 - Schéma représentatif de l'architecture neuronale du système conversationnel.

Comme nous l'avons expliqué dans le précédent chapitre, un modèle Séquence à Séquence est composé de deux parties : un encodeur et un décodeur. La partie de notre

encodeur dispose d'un mécanisme d'attention, qui a également été expliqué dans le chapitre précédent. Chacun de ses composants peut être défini comme suit :

- **L'encodeur :** la fonction de l'encodeur est de traiter ou coder l'intégration d'entrées qui sont des vecteurs de longueur variable et de générer des états intermédiaires qui consistent en des vecteurs de longueur fixe. Dans notre cas, la séquence d'entrée représente la question fournie au réseau. Comme on peut le constater dans la figure ci-dessus, notre encodeur est constitué d'une première couche d'entrée pour recevoir la séquence d'entrée, une couche dite *Embedding* dont le rôle est de convertir chaque mot de la séquence d'entrée en un vecteur représentatif et interprétable par le réseau (en utilisant la technique de Word Embedding, GloVe, vue précédemment). Cette couche Embedding est suivie par trois couches cachées récurrentes comportant des cellules LSTM, dont la dernière couche est reliée à la couche d'attention du mécanisme d'attention.
- **Le décodeur :** le décodeur pour sa part génère mot par mot la séquence de sortie, qui correspond à la réponse de la question, à partir de la représentation latente de longueur fixe fournie par l'encodeur. Comme notre encodeur, le décodeur est à son tour composé d'une première couche Embedding, suivie de trois couches cachées récurrentes constituées de cellules LSTM. La couche Embedding sert ici à encoder la sortie de chaque cellule LSTM pour la passer en entrée à la cellule suivante, ce qui va permettre au décodeur de connaitre apriori l'encodage qui précède chaque encodage cible. De plus, grâce au mécanisme d'attention, le décodeur apprend également à sélectionner les encodages auxquels il doit prêter attention. Ces encodages, qui représentent la sortie de la dernière couche cachée de LSTM, sont transmis vers la couche d'activation qui est composée de fonctions d'activation Softmax pour générer la sortie finale du modèle.
- **Le mécanisme d'attention :** le rôle du mécanisme d'attention repose sur l'utilisation de toutes les informations contextuelles de la séquence d'entrée afin de pouvoir décoder la séquence cible. L'élément central du mécanisme d'attention est la couche d'attention. Cette dernière est reliée à la dernière

couche cachée de LSTM pour calculer les poids d'attention pour chaque mot de la séquence d'entrée afin de produire le vecteur de contexte. Les informations de contexte produites par la dernière couche cachée de LSTM seront intercalées avec le vecteur de contexte produit par la couche d'attention, avant de passer à la couche d'activation. Le but ici est de tenir compte à la fois du contexte local du décodeur et du contexte calculé à partir de la question qui est l'entrée de l'encodeur.

3.2.5.2. Génération de réponse

Le processus de génération de réponse de notre système conversationnel comporte les étapes suivantes :

i. Récupération des données prétraitées.

ii. Transformation au préalable de ces données en un certain format des cibles pour pouvoir être introduites dans chacun de l'encodeur et du décodeur du modèle Seq2Seq :
   - Ces données doivent être divisées en lots (l'encodeur est alimenté par un lot de 10 questions à la fois, par exemple).
   - Chaque réponse du lot cible doit commencer par le token <SOS> et se terminer par le token <EOS>.

iii. Application du Padding et du Bucketing : toutes les séquences d'un lot, qu'elles soient des questions ou des réponses, doivent impérativement avoir la même longueur.

iv. L'entrainement de l'encodeur et du décodeur se fait en parallèle :
   - Un lot de questions est fourni à l'encodeur qui les transformera d'abord en vecteurs de mots pour produire l'état de l'encodeur.
   - L'état de l'encodeur est fourni au décodeur pour produire les réponses adéquates selon le lot de réponses parallèles.

Chacune de ces étapes est illustrée dans le schéma suivant :

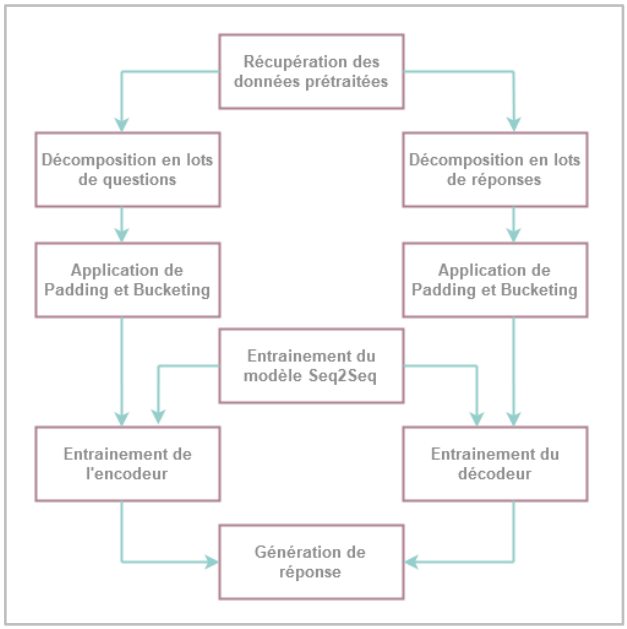

FIGURE 4.5 - Etapes du processus de génération de réponse.

3.2.6. Fonctionnement du classifieur de sentiments

Dans ce qui suit, nous allons expliquer l'architecture neuronale adoptée pour notre classifieur de sentiments ainsi que les étapes du processus suivi pour la détection du sentiment.

3.2.6.1. Architecture neuronale du classifieur de sentiments

Pour l'architecture de notre système d'analyse des sentiments, nous avons choisi de construire notre modèle en utilisant un réseau de neurones récurrents muni de cellules LSTM. Comme le montre la figure ci-dessous, notre modèle est constitué d'une couche d'entrée suivie d'une couche Embedding reliée à trois couches récurrentes cachées avec des cellules LSTM, et d'une couche d'activation avec des fonctions d'activation Softmax permettant de produire la sortie finale.

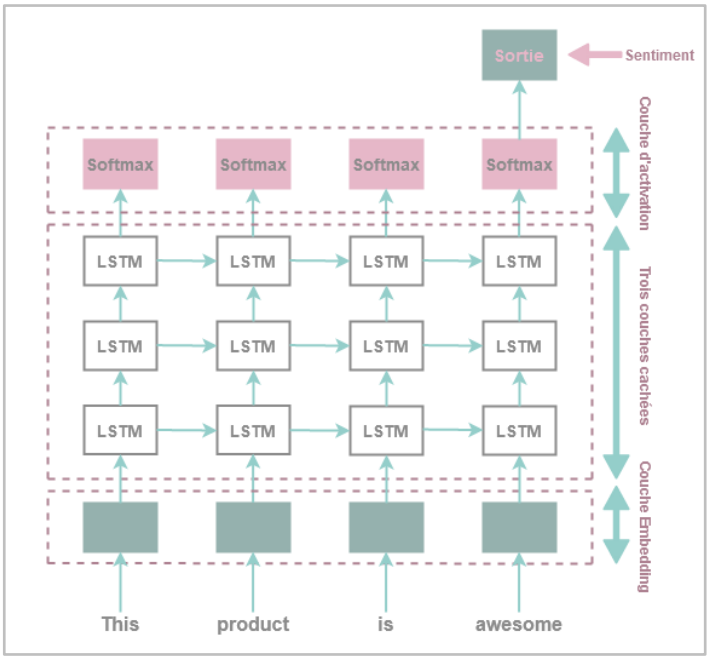

FIGURE 4.6 - Schéma représentatif de l'architecture neuronale du classifieur de sentiments.

Dans notre modèle, chaque mot reçoit d'abord un vecteur de caractéristiques à partir de la couche Embedding. Par la suite, cette séquence de caractéristiques codée passe en outre par le réseau neuronal récurrent doté de cellules LSTM pour obtenir des informations de séquence. Enfin, les informations de séquence codées sont converties pour former la sortie via la couche d'activation. En particulier, les états cachés des cellules LSTM peuvent être concaténés dans la dernière étape - c'est-à-dire la dernière cellule -, et transmis à la couche de sortie pour générer le résultat final.

3.2.6.2. Détection du sentiment

Le processus de détection du sentiment de notre classifieur de sentiments comporte les étapes ci-dessous :

i. Récupération des données prétraitées.

ii. Conversion de ces données en une longueur spécifique pour pouvoir être traitées par notre modèle :

- Eliminer les valeurs aberrantes : données extrêmement courtes ou longues.
- Appliquer le Padding ou couper les données restantes pour obtenir des avis de la même longueur.

iii. Décomposition des données en lots (comme nous l'avons fait précédemment).

iv. L'entrainement du modèle se fait en introduisant un lot d'avis au réseau :

- Chaque avis est préalablement converti en vecteur de mots avant d'être traité.
- Le sentiment de l'avis est prédit par le réseau et émis en sortie, qu'il soit positif ou négatif.

Chacune de ces étapes est illustrée dans le schéma suivant :

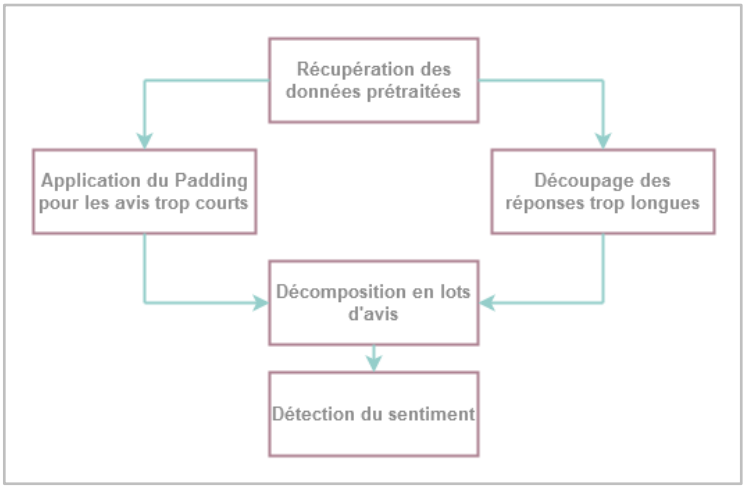

FIGURE 4.7 - Etapes du processus de détection du sentiment.

3.2.7. Dropout pour la régularisation

Le problème majeur de l'Apprentissage Profond est de créer un réseau de neurones performant non seulement sur les données d'apprentissage mais aussi sur les nouvelles entrées. Cependant, le sur-ajustement[10] représente un sérieux problème dans des réseaux pareils. Diverses stratégies utilisées dans l'Apprentissage Profond sont clairement conçues pour diminuer l'erreur de test, potentiellement au détriment d'une erreur d'apprentissage accrue. Ces stratégies sont collectivement connues sous le nom de *régularisation*. Le Dropout est l'une de ces stratégies de régularisation que nous avons adopté pour nos deux modèles.

Le terme *Dropout* fait référence au rejet ou à l'abandon d'unités (à la fois visibles et cachées) dans le réseau de neurones. Le Dropout modifie le réseau lui-même en supprimant aléatoirement les neurones du réseau à chaque itération de la phase d'apprentissage. Lorsque différents ensembles de neurones sont éliminés, ceci

[10] **Sur-ajustement :** en anglais **Overfitting**, désigne le fait que le modèle prédictif produit par l'algorithme de Machine Learning s'adapte bien à l'ensemble d'apprentissage.

équivaut à entrainer des réseaux de neurones différents qui se sur-ajustent de différentes façons, ce qui évite aux unités neuronales de trop s'adapter. La figure suivante illustre un exemple de réseau de neurones avant et après application du Dropout :

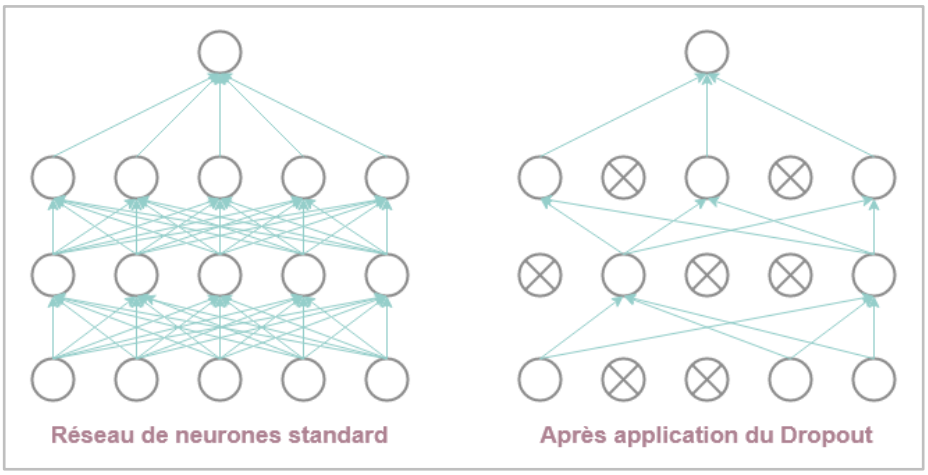

FIGURE 4.8 - Régularisation d'un réseau de neurones avec dropout.

Lors de la phase de test, il devient facile d'estimer l'effet de la moyenne des prédictions de tous les réseaux éclaircis en utilisant simplement un réseau non éclairci dont les poids sont inférieurs. Cela diminue considérablement le sur-ajustement et fournit une amélioration significative comparé à d'autres méthodes de régularisation. Selon Srivastava et al. [107], renoncer à 20% des unités d'entrée et 50% des unités cachées s'est souvent avéré optimal, c'est pourquoi nous avons choisi d'appliquer un Dropout selon ces valeurs.

3.3. Post-traitement

Le post-traitement représente l'étape finale de notre système. Les étapes du post-traitement sont résumées comme suit :

i. Au cours de l'apprentissage de nos deux modèles, chacun d'eux génère deux valeurs lors de chaque époque de l'entrainement : la valeur de la fonction de coût et la valeur de précision. Ces valeurs servent au test et à l'utilisation des modèles. La requête fournie par l'utilisateur est tout d'abord interceptée par nos deux modèles.

ii. Le modèle Seq2Seq du système conversationnel permet de générer la réponse appropriée à la requête, qui sera ensuite retournée à l'utilisateur.

iii. Le modèle RNN du classifieur de sentiments permet à son tour de détecter si le sentiment de la requête est positif ou négatif.

La figure ci-dessous schématise le processus de post-traitement de notre système global :

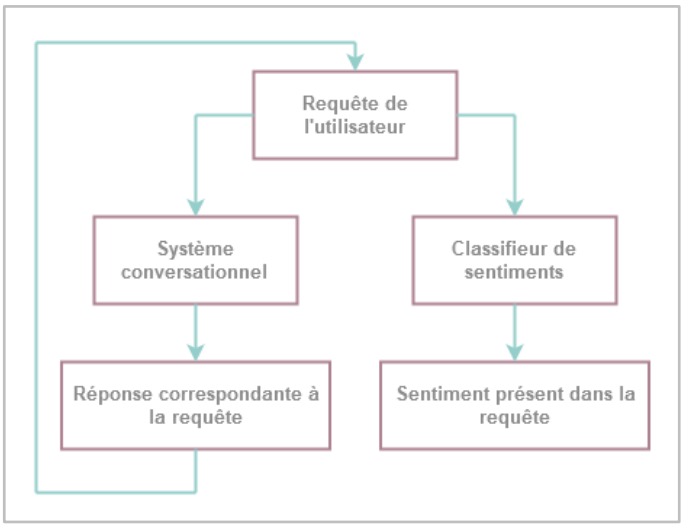

FIGURE 4.9 - Schéma représentatif du post-traitement du système global.

Le schéma ci-dessous résume les étapes de prétraitement, de traitement et de post-traitement par lesquelles passe une requête fournie par l'utilisateur :

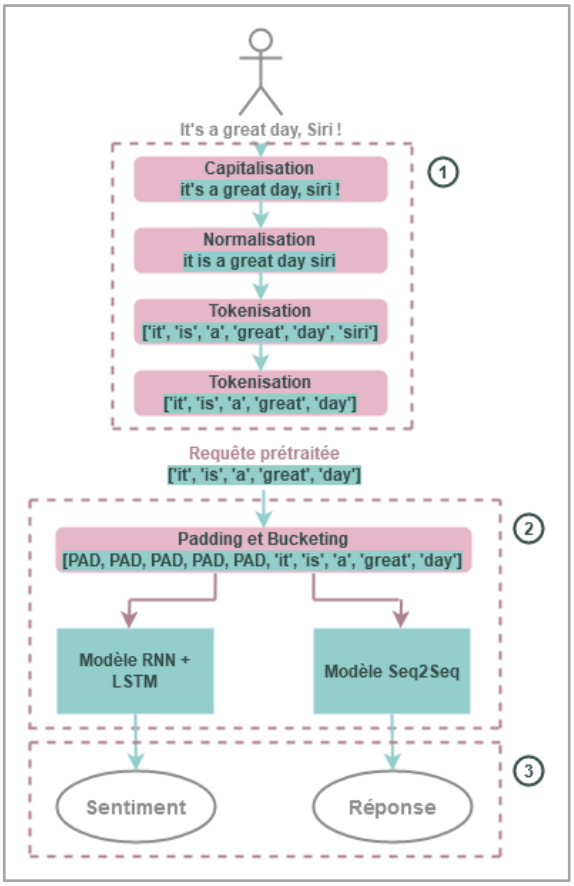

FIGURE 4.10 - Schéma récapitulatif des étapes de prétraitement, de traitement et de post-traitement.

4. Conclusion

À travers ce chapitre, nous avons rappelé la problématique initiale et les objectifs à atteindre de notre système. Pour ce faire, nous avons énuméré les différentes étapes de prétraitement par lesquelles passent les corpus utilisés pour l'apprentissage des deux modules qui constituent notre système global. Ensuite, nous avons présenté le processus d'apprentissage ou de traitement de chacun de nos deux sous-systèmes ainsi que leurs architectures et leurs fonctionnements. Enfin, nous avons cité les diverses étapes de post-traitement du système global.

Le chapitre suivant sera dédié à l'implémentation et la mise en œuvre de notre système qui sera par la suite évalué.

Tests et Validation du Système

1. Introduction

Pour mettre en œuvre un système ou une application informatique, l'environnement et le domaine d'application de ces derniers doivent constituer les principaux critères de sélection des approches et des outils utilisés pour la conception et l'implémentation. Le présent chapitre sera donc consacré à la présentation des outils matériels et logiciels pour la concrétisation et l'implémentation de notre système conformément à la description de son fonctionnement. Ce système devra par la suite être évalué en utilisant des normes et mesures spécifiques pour déterminer son efficacité selon les corpus ayant servi à son apprentissage et son évaluation.

2. Environnement de travail

Afin d'atteindre l'objectif visé par ce travail, nous avons utilisé certaines ressources matérielles et logicielles que nous citons ci-dessous :

2.1. Ressources matérielles

Pour chacune des phases d'implémentation et d'entrainement de test de notre système, nous avons exploité notre machine personnelle dont les spécifications sont les suivantes :

TABLEAU 5.1 - Caractéristiques de la machine utilisée.

Processeur	Intel® Core™ i5-6300U CPU @ 2.40GHz 2.50GHz
RAM	8.00 Go

2.2. Ressources logicielles

Pour l'implémentation de notre système, nous avons utilisé le célèbre langage de programmation multi-usage et multiparadigme Python (voir le logo dans la figure ci-dessous). Ce dernier est un langage de script structuré et open source de haut niveau. Initialement développé en 1969 par Guide Van Rossum, il est comme la majorité des

applications et outils open source, maintenu par une communauté de développeurs éparpillés à travers le monde.

FIGURE 5.1 - Logo du langage de programmation Python.

Grâce à ses bibliothèques spécialisées, ce langage peut être utilisé dans divers contextes et s'accommoder à tout type d'application. Néanmoins, il est considérablement utilisé pour l'automatisation des tâches élémentaires mais tant fastidieuses, notamment la mise en œuvre de modèles d'apprentissage automatique.

Python, tout comme d'autres langages de programmation, dispose de plusieurs versions. Les deux principales versions sont la version 2, publiée en 2000 et la version 3 publiée à son tour en 2006. Dans notre travail nous avons, à l'évidence, utilisé la version 3 qui est plus récente, puisque le support de la deuxième version est fini depuis le 1er janvier 2020.

Pour développer notre système, nous avons utilisé l'environnement de développement libre et multiforme pour Python, Spyder (voir le logo dans la figure ci-dessous). Cet environnement de développement intègre plusieurs bibliothèques à usage scientifique telles que NumPy, Matplotlib et SciPy. Spyder a été créé et développé en 2008 par Pierre Raybaut, et depuis 2012 est maintenu par une communauté de programmeurs qui partagent le trait commun de faire partie de la communauté scientifique Python.

FIGURE 5.2 - Logo de l'IDE Spyder.

Comparé à d'autres IDE de développement scientifique, Spyder dispose d'un ensemble remarquable de fonctionnalités open – source, multiplateformes, écrites en Python et peuvent être utilisées sous une licence non copyleft. Spyder est extensible grâce à des plugins et API. Il inclut la prise en charge d'outils de contrôle de données

interactifs et intègre des dispositifs d'assurance qualité et d'introspection particuliers au code Python.

2.3. Bibliothèques utilisées

Les sections suivantes seront consacrées à la présentation des bibliothèques utilisées pour le développement de notre système.

2.3.1. TensorFlow

TensorFlow est une plateforme d'apprentissage automatique open source de bout en bout (voir le logo dans la figure ci-dessous). Il est doté d'un écosystème exhaustif et flexible de bibliothèques, de ressources et d'outils communautaires permettant aux chercheurs de promouvoir les avancées de l'apprentissage automatique, et aux développeurs de créer et de déployer aisément des applications d'apprentissage automatique.

FIGURE 5.3 - Logo de la bibliothèque TensorFlow.

TensorFlow a vu le jour aux mains d'ingénieurs et de chercheurs membres de l'équipe Google Brain au sein de l'organisation de recherche sur l'intelligence artificielle de Google. TensorFlow fournit des API Python et C++ statistiques, ainsi qu'une API rétro compatible non cautionnée pour d'autres langages.

2.3.2. NumPy

NumPy est la bibliothèque open-source fondamentale du calcul scientifique en Python pour le traitement de tableaux à usage général (voir le logo dans la figure ci-dessous). Elle fournit des tableaux multidimensionnels de haute performance et des outils pour les manipuler.

FIGURE 5.4 - Logo de la bibliothèque NumPy.

Au-delà de ses nombreuses utilisations dans le cadre scientifique, Numpy peut aussi être utilisée en tant que conteneur multidimensionnel performant pour des données

générales. Ce qui rend NumPy accessible et productif est sa syntaxe de haut niveau, sa large variété de fonctions mathématiques complètes et de générateurs de nombres aléatoires ainsi que sa capacité de prise en charge d'un large éventail de plateformes matérielles et informatiques.

2.3.3. NLTK

NLTK (Natural Language ToolKit) est une collection de modules de programmes open-source, de didacticiels et d'ensembles de problèmes permettant de créer des programmes pour l'analyse de texte. Ce kit a été à l'origine créé par Steven Bird et Edward Loper en conjonction avec des cours de linguistique computationnelle à l'université de Pennsylvanie en 2001.

NLTK inclut le traitement statistique du langage naturel et est lié à des corpus labélisés. Joint à la librairie, NTLK offre des tutoriels, des échantillons de données, des démonstrations graphiques ainsi qu'une documentation complète sur l'interface de programmation API.

3. Description du système

Pour le bon fonctionnement de chaque partie de notre système, nous avons eu recours à certaines librairies spécialisées. Nous allons d'abord expliquer le fonctionnement de notre système pour ensuite présenter les bibliothèques ayant servi à son apprentissage et son fonctionnement.

3.1. Acquisition des corpus

Dans le but de mener les observations nécessaires à la création, l'implémentation, l'entrainement et la validation de nos deux modules formant notre système, nous avons exploité deux corpus : le corpus open-source Twitter[11] pour entrainer le système conversationnel, et le dataset IMBD Movie Reviews[12] pour entrainer notre classifieur de sentiments.

Le corpus conversationnel de Twitter est constitué de 50 000 paires de question-réponse d'une longueur moyenne de 250 mots pour chacun des ensembles de questions et de réponses. Quant au corpus de IMBD Movie Reviews, il est composé de 25 000 avis divisés en deux catégories, placées chacune dans un dossier indépendant : des avis

[11] **URL:** https://github.com/Marsan-Ma-zz/chat_corpus
[12] **URL:** http://ai.stanford.edu/~amaas/data/sentiment/

positifs et des avis négatifs. Chacune de ces catégories contient à son tour 12 500 avis et chacun de ces avis est conservé dans un fichier .txt, comme le montre le tableau suivant :

TABLEAU 5.2 - Corpus utilisé pour l'Analyse des Sentiments.

Source	Avis positifs	Avis négatifs	Total des avis
IMBD Movie Reviews	12 500	12 500	25 000

La figure ci-dessous a pour but de montrer la structure du corpus IMBD utilisé (après nettoyage) :

Indice	Type	Taille	Valeur
0	str	1	bromwell high is a cartoon comedy it ran at the same time as some othe ...
1	str	1	homelessness or houselessness as george carlin stated has been an issu ...
2	str	1	brilliant overacting by lesley ann warren best dramatic hobo lady i ha ...
3	str	1	this is easily the most underrated film inn the brooks cannon sure its ...
4	str	1	this is not the typical mel brooks film it was much less slapstick tha ...
5	str	1	this isnt the comedic robin williams nor is it the quirkyinsane robin ...
6	str	1	yes its an art to successfully make a slow paced thriller the story u ...
7	str	1	in this critically acclaimed psychological thriller based on true even ...
8	str	1	the night listener 2006 12 robin williams toni collette bobby cannaval ...
9	str	1	you know robin williams god bless him is constantly shooting himself i ...
10	str	1	when i first read armistead maupins story i was taken in by the human ...

FIGURE 5.5 - Structure du corpus utilisé pour l'Analyse des Sentiments.

3.2. Architecture de fonctionnement

Le fonctionnement de notre système global peut être résumé selon les étapes illustrées dans la figure ci-dessous. Pour notre système conversationnel et classifieur de sentiments deux ensembles ont été introduits : les questions et réponses de l'ensemble d'apprentissage et les questions de l'ensemble de test pour le système conversationnel ; les avis et leurs étiquettes de l'ensemble d'apprentissage et les avis de l'ensemble de test pour le classifieur de sentiments. Le système conversationnel produit en sortie les réponses aux questions de test, tandis que le classifieur de sentiments produit en sortie les sentiments des avis de test.

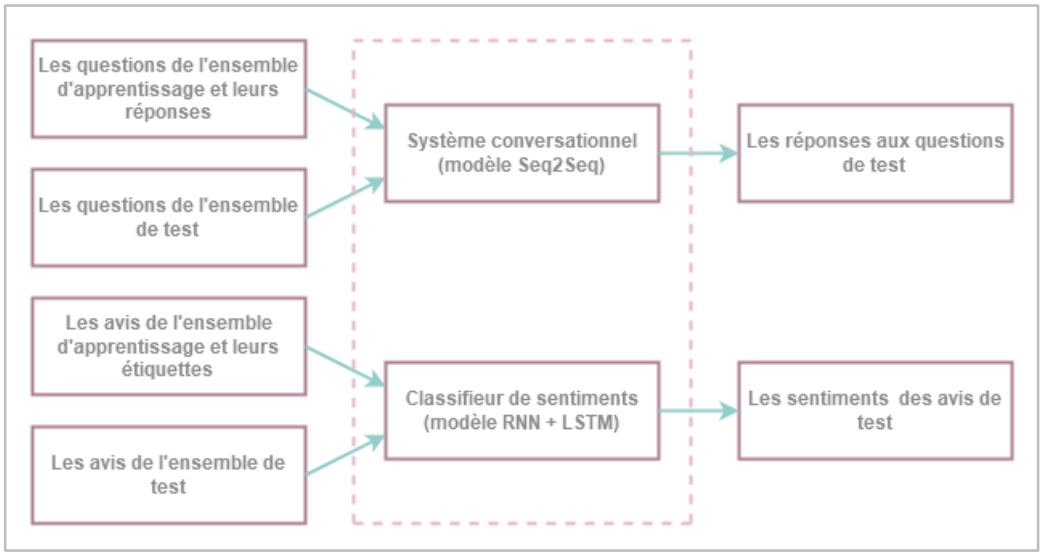

FIGURE 5.6 - Architecture de fonctionnement du système global.

3.2.1. Prétraitement

La phase initiale de chaque sous-système comprend le prétraitement des données, avant tout apprentissage à l'aide de bibliothèques spécialisées. À cette fin, nous commençons par la première étape du prétraitement, qui est la capitalisation. Cette étape consiste à réduire toutes les lettres qui composent les données de chaque ensemble de données en minuscules. L'image ci-dessous montre un exemple de requête avant et après application de la capitalisation :

FIGURE 5.7 - Exemple de capitalisation.

La prochaine étape est l'étape de normalisation. Cette étape consiste à conserver toutes les données sur un même pied d'égalité en les convertissant en une seule forme canonique identique. Ceci est effectué en éliminant les espaces blancs dupliqués et les caractères spéciaux, ne laissant que les caractères alphabétiques et en convertissant les caractères numériques en mots. La figure suivante illustre cette étape :

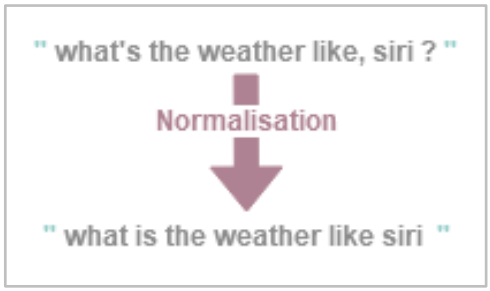

FIGURE 5.8 - Exemple de normalisation.

Après avoir effectué une capitalisation et une normalisation, vient le tour de l'étape de tokenisation. Comme son nom l'indique, le rôle de cette étape est de transformer un morceau de texte en unités plus petites nommées « tokens », comme il est montré dans la figure suivante :

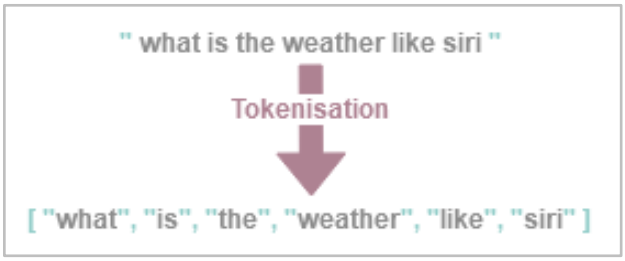

FIGURE 5.9 - Exemple de tokenisation.

Vient la dernière étape de la phase de prétraitement, qui est le filtrage des mots non fréquents. Premièrement, un vocabulaire est construit pour affecter à chaque mot son nombre d'occurrence. Ceci est effectué grâce à la fonction de Frequency Distribution intégrée dans la bibliothèque NLTK. Ensuite, 5% des mots les moins fréquents sont éliminés du vocabulaire. Ceci va permettre d'optimiser le traitement.

Les données finalement prétraitées passent par une phase de préparation, qui consiste à diviser les dataset en ensemble d'apprentissage, de test et de validation. Ceci est expliqué dans la section suivante.

3.2.2. Préparation des données

Pour construire chacun de nos deux modèles finaux, nous divisons nos corpus en plusieurs datasets : en particulier, trois datasets sont utilisés au cours de la création de nos modèles. Nos deux corpus initiaux sont donc chacun divisé en trois parties : un ensemble d'entrainement, un ensemble de test et un ensemble de validation.

Initialement, le modèle est adapté à un ensemble d'apprentissage représentant 70% du corpus global et qui consiste en des exemples qui servent à ajuster les paramètres

du modèle. Ce dernier est entrainé sur l'ensemble d'apprentissage à l'aide de méthodes d'optimisation pour produire un résultat qui est ensuite comparé avec la cible associée à chaque vecteur d'entrée de l'ensemble d'apprentissage. Les paramètres sont donc ajustés en fonction du résultat de la comparaison produit et de l'algorithme utilisé.

Progressivement, l'ensemble de validation permet au modèle ajusté de prédire des réponses aux observations (des réponses aux questions pour le système conversationnel et un sentiment à la requête pour le classifieur de sentiments). Ainsi, le dataset de validation fournit une approximation non biaisée de l'accommodation tout en ajustant les hyperparamètres du modèle. Finalement, l'ensemble de test est utilisé pour fournir une évaluation non biaisée de l'ajustement final du modèle sur l'ensemble d'apprentissage.

3.2.3. Traitement

Avant de créer nos deux modèles, il est nécessaire de construire pour chacun d'entre eux une matrice de Word Embedding. Pour ce faire, nous avons eu recours à un modèle pré-entrainé sur GloVe pour générer les Word Embeddings de chacune de nos questions et réponses du dataset conversationnel et des avis du dataset de sentiments. Pour la matrice qui représente les labels ou les étiquettes de sentiments, nous utilisons des vecteurs de taille 2 sous forme de [1,0] pour un sentiment positif et [0,1] pour un sentiment négatif. La fonction *lookup* de la bibliothèque TensorFlow est ensuite appelée pour générer nos vecteurs de mots sous forme de Tensor 3D de dimensionalité *batch size * longueur de séquence maximale * taille du Word Embedding*, où le *batch size* signifie la taille du lot passé en entrée à chacun des réseaux. Ceci est illustré à travers le code ci-dessous :

```
In [4]: data = tf.Variable(tf.zeros([batchSize, maxSeqLength,
numDimensions]),dtype=tf.float32)
   ...: data = tf.nn.embedding_lookup(wordVectors,input_data)
   ...: print(data)
Tensor("embedding_lookup_1:0", shape=(24, 250, 50), dtype=float32)
```

FIGURE 5.10 - Création de Tensors 3D.

Dans ce qui suit, nous allons expliquer les démarches suivies pour l'implémentation de chaque modèle de sous-système.

i. Construction du modèle conversationnel :

Pour la construction de notre modèle Seq2Seq en charge du système conversationnel, nous avons utilisé la bibliothèque TensorFlow pour créer quatre principales fonctions qui forment le corps de notre modèle. La première fonction vise à mettre en place l'encodeur en créant une première couche de LSTM de la classe *LSTM cell* de TensorFlow. À cette couche est appliqué un Dropout de 50% (durant l'entrainement, 50% des neurones seront comme inexistants et leurs poids non mis à jour). Ensuite, la première couche LSTM est multipliée par le nombre de couches. La sortie de cette fonction est l'état du décodeur qui est notre vecteur intermédiaire. La figure ci-dessous représente le code permettant d'implémenter cette fonction :

```
In [2]: def encoder_rnn(rnn_inputs, rnn_size, num_layers,
keep_prob, sequence_length):
   ...:     lstm = tf.contrib.rnn.BasicLSTMCell(rnn_size)
   ...:     lstm_dropout = tf.contrib.rnn.DropoutWrapper(lstm,
input_keep_prob = keep_prob)
   ...:     encoder_cell =
tf.contrib.rnn.MultiRNNCell([lstm_dropout] * num_layers)
   ...:     encoder_output, encoder_state =
tf.nn.bidirectional_dynamic_rnn(cell_fw = encoder_cell,
   ...:
cell_bw = encoder_cell,
   ...:
sequence_length = sequence_length,
   ...:
inputs = rnn_inputs,
   ...:
dtype = tf.float32)
   ...:     return encoder_state
```

FIGURE 5.11 - Implémentation de la fonction permettant de créer l'encodeur.

Le pseudo-algorithme ci-dessous résume les étapes de création de l'encodeur :

Début

Construction de la 1ère couche LSTM

Application-Dropout()

*Multiplier la 1ère couche LSTM * nombre de couches*

Retourner l'état de l'encodeur

Fin

Pour décoder le vecteur intermédiaire qui est l'état final de l'encodeur, une fonction qui repose sur le mécanisme d'attention est créée. Cette fonction joue alors le rôle de la couche d'attention grâce à une fonction intégrée qui va décoder les poids d'attention en utilisant plusieurs paramètres d'attention. La figure ci-dessous représente le script qui permet d'implémenter cette fonction :

```
In [4]: def decode_training_set(encoder_state, decoder_cell,
decoder_embedded_input, sequence_length, decoding_scope,
output_function, keep_prob, batch_size):
   ...:     attention_states = tf.zeros([batch_size, 1,
decoder_cell.output_size])
   ...:     attention_keys, attention_values,
attention_score_function, attention_construct_function =
tf.contrib.seq2seq.prepare_attention(attention_states,
attention_option = "bahdanau", num_units =
decoder_cell.output_size)
   ...:     training_decoder_function =
tf.contrib.seq2seq.attention_decoder_fn_train(encoder_state[0],atte
ntion_keys,attention_values,attention_score_function,attention_cons
truct_function,name = "attn_dec_train")
   ...:     decoder_output, decoder_final_state,
decoder_final_context_state =
tf.contrib.seq2seq.dynamic_rnn_decoder(decoder_cell,training_decode
r_function,decoder_embedded_input,sequence_length,scope =
decoding_scope)
   ...:     decoder_output_dropout = tf.nn.dropout(decoder_output,
keep_prob)
   ...:     return output_function(decoder_output_dropout)
```

FIGURE 5.12 - Implémentation de la fonction permettant de décoder le vecteur intermédiaire.

La troisième fonction implémentée a pour rôle de construire la deuxième partie du modèle Seq2Seq qui est le décodeur. Son principe est similaire à celui de la fonction de l'encodeur : une première couche de LSTM est créée à laquelle un Dropout est appliqué pour ensuite être multipliée par le nombre de couches. Par la suite, les poids des neurones sont initialisés via une distribution normale tronquée et les biais sont initialisés à zéro. Les paramètres sont combinés dans une fonction de sortie pour générer la sortie finale du décodeur. Cette fonction est implémentée grâce au code illustré dans la figure suivante :

```
In [5]: def decoder_rnn(decoder_embedded_input,
decoder_embeddings_matrix, encoder_state, num_words,
sequence_length, rnn_size, num_layers, word2int, keep_prob,
batch_size):
   ...:     with tf.variable_scope("decoding") as decoding_scope:
   ...:         lstm = tf.contrib.rnn.BasicLSTMCell(rnn_size)
   ...:         lstm_dropout = tf.contrib.rnn.DropoutWrapper(lstm,
input_keep_prob = keep_prob)
   ...:         decoder_cell =
tf.contrib.rnn.MultiRNNCell([lstm_dropout] * num_layers)
   ...:         weights = tf.truncated_normal_initializer(stddev =
0.1)
   ...:         biases = tf.zeros_initializer()
   ...:         output_function = lambda x:
tf.contrib.layers.fully_connected(x,num_words,None,scope =
decoding_scope,weights_initializer = weights,biases_initializer =
biases)
   ...:         training_predictions =
decode_training_set(encoder_state,decoder_cell,decoder_embedded_inp
ut,sequence_length,decoding_scope,output_function,keep_prob,batch_s
ize)
   ...:         decoding_scope.reuse_variables()
   ...:         test_predictions =
decode_test_set(encoder_state,decoder_cell,decoder_embeddings_matri
x,word2int['<SOS>'],word2int['<EOS>'],sequence_length -
1,num_words,decoding_scope,output_function,keep_prob,batch_size)
   ...:     return training_predictions, test_predictions
```

FIGURE 5.13 - Implémentation de la fonction permettant de créer le décodeur.

Le pseudo-algorithme ci-dessous résume les étapes de création du décodeur :

Début

Construction de la 1ère couche LSTM

Application-Dropout()

*Multiplication de la 1ère couche LSTM * nombre de couches*

Initialisation des poids et des biais

Retourner la sortie finale du décodeur

Fin

La fonction finale combine les trois fonctions précédentes en charge de la création de l'encodeur, le vecteur intermédiaire et le décodeur pour mettre en place notre modèle Seq2Seq final. Cette fonction finale est créée grâce au script de la figure ci-dessous :

```
In [6]: def seq2seq_model(inputs, targets, keep_prob, batch_size,
sequence_length, answers_num_words, questions_num_words,
encoder_embedding_size, decoder_embedding_size, rnn_size,
num_layers, questionswords2int):
   ...:     encoder_embedded_input =
tf.contrib.layers.embed_sequence(inputs,answers_num_words +
1,encoder_embedding_size,initializer =
tf.random_uniform_initializer(0, 1))
   ...:     encoder_state = encoder_rnn(encoder_embedded_input,
rnn_size, num_layers, keep_prob, sequence_length)
   ...:     preprocessed_targets = preprocess_targets(targets,
questionswords2int, batch_size)
   ...:     decoder_embeddings_matrix =
tf.Variable(tf.random_uniform([questions_num_words + 1,
decoder_embedding_size], 0, 1))
   ...:     decoder_embedded_input =
tf.nn.embedding_lookup(decoder_embeddings_matrix,
preprocessed_targets)
   ...:     training_predictions, test_predictions =
decoder_rnn(decoder_embedded_input,decoder_embeddings_matrix,encode
r_state,questions_num_words,sequence_length,rnn_size,num_layers,que
stionswords2int,keep_prob,batch_size)
   ...:     return training_predictions, test_predictions
```

FIGURE 5.14 - Implémentation de la fonction permettant de créer le modèle Seq2Seq final.

Pour entrainer ce modèle, un ensemble d'hyperparamètres devra être défini et fourni à la fonction du modèle Seq2Seq. Trois ensembles d'hyperparamètres ont de ce fait été choisis pour entrainer le modèle indépendamment. Ces ensembles seront discutés et comparés ultérieurement dans ce chapitre.

ii. Construction du classifieur de sentiments :

Pour l'implémentation du classifieur de sentiments, la bibliothèque TensorFlow a également été utilisée. Tout d'abord, une première couche de LSTM est créée et à laquelle est appliqué un Dropout. Cette couche de LSTM est combinée au Tensor 3D de Word Embeddings et introduite à la fonction *dynamic RNN* de TensorFlow pour être multipliée par le nombre de couches. Les poids sont ensuite initialisés via une distribution normale tronquée et les biais initialisés à zéro. La fonction dynamic RNN génère une première sortie qui est le vecteur du dernier état caché du réseau. Ce vecteur est redimensionné et multiplié par la matrice finale des poids et des biais. Le produit de cette multiplication est finalement passé à la dernière couche qui est la fonction d'activation Softmax pour produire la sortie finale du réseau.

Le pseudo-algorithme ci-dessous résume les étapes de construction du classifieur de sentiments :

Début

Construction de la 1ère couche LSTM

Application-Dropout()

Application de la fonction dynamic RNN

Génération du vecteur du dernier état caché du RNN

Redimensionnement du vecteur

Multiplication du vecteur par la matrice finale des poids et biais

Retourner la sortie finale

Fin

---

4. Mesures d'évaluation

Dans les sections suivantes, nous allons présenter les mesures choisies pour évaluer nos deux sous-systèmes, ainsi que les résultats obtenus.

4.1. Mesure d'évaluation du système conversationnel

Dans cette section nous allons dans un premier temps présenter la métrique d'évaluation que nous avons choisi pour le système conversationnel, pour ensuite discuter les résultats de l'évaluation obtenus en utilisant cette métrique.

4.1.1. BLEU Score

Bilingual Evaluation Understudy Score, abrégé en BLEU Score, est une mesure permettant d'évaluer des séquences de texte générées à travers des séquences de texte de référence. Bien que développée particulièrement pout mesurer la précision d'un texte traduit automatiquement, cette mesure peut être utilisée pour estimer l'exactitude du texte généré pour un ensemble de tâches de traitement automatique du langage naturel qui incluent le Résumé automatique de texte, la Simplification de texte, l'Annotation automatique d'images et pour notre cas, et les Agents conversationnels. BLEU Score a été l'une des premières métriques à démontrer une puissante corrélation avec le jugement humain et reste l'un des outils d'évaluation automatisés à faible coût les plus répandus.

BLEU Score utilise la méthode de modélisation n-gramme de taille 1 à 4 pour comparer le texte généré avec le texte de référence dans les données de test. Le score BLEU de cette comparaison peut être calculé soit au niveau de la phrase (chaque phrase générée obtient un score individuel) ou au niveau du corpus (l'ensemble du corpus obtient un score unique). Dans les deux cas, le nombre de correspondances entre la séquence générée et la séquence de référence sera calculé de manière pondérée, sachant que ces correspondances sont indépendantes de la position. Un degré de correspondance plus grand implique un degré de similitude plus grand avec la séquence de référence et donc un BLEU score plus élevé. Il convient de noter que le BLEU score est généralement rapporté sous forme de pourcentage, c'est-à-dire de 0 à 100%. Le BLEU score est donc calculé selon la formule mathématique suivante :

$$BLEU = \min\left(1, \frac{output\_length}{reference\_length}\right)\left(\prod_{n=1}^{4} precision_n\right)^{1/4} \quad (5.1)$$

Où :

- $output\_length$: est la longueur de la séquence générée.
- $reference\_length$: est la longueur de la séquence de référence.
- $precision_n$: est la précision du n-gramme qui est calculée en additionnant les correspondances des n-grammes pour chaque phrase générée S dans le corpus C, comme suit :

$$precision_n = \frac{\sum_{S \in C} \sum_{ngram \in S} Count_{matched}(ngram)}{\sum_{S \in C} \sum_{ngram \in S} Count(ngram)} \quad (5.2)$$

Dans notre cas, nous implémentons l'approche de BLEU Score au niveau de la phrase pour calculer les scores de chaque texte de réponse généré par notre système conversationnel. L'exemple ci-dessous vise à mieux comprendre le fonctionnement de l'algorithme BLEU Score au niveau de la phrase avec les paramètres n-gramme de 1 à 4.

Séquence de référence: This is a test for BLEU Score

Séquence hypothèse: This is a test for Score

Les résultats sont illustrés dans le tableau suivant :

TABLEAU 5.3 - Exemple d'évaluation selon la mesure BLEU Score.

BLEU Score	Scores individuels n-gramme			
	1-gramme	2-gramme	3-gramme	4-gramme
0.67 = 67%	0.84 = 84%	0.67 = 67%	0.63 = 63%	0.56 = 56%

Comme nous pouvons le constater, plus les modèles de n-gramme sont petits et plus le score est élevé. Il s'agit ici de la méthodologie générale de fonctionnement de BLEU Score.

4.1.2. Evaluation du système conversationnel

Dans le but d'entrainer le modèle du système conversationnel, nous avons exploité trois différentes configurations utilisant une variété d'hyperparamètres pour évaluer la performance du modèle sur ces derniers. Le tableau ci-dessous montre les trois configurations choisies selon ces hyperparamètres :

TABLEAU 5.4 - Configurations utilisées pour l'apprentissage du modèle du système conversationnel.

Hyperparamètres	Configurations		
	[1]	[2]	[3]
Epoques	100	500	50
Taille du batch	32	16	128
Nombre de couches	3	3	3
Taux d'apprentissage	0.001	0.0001	0.001
Keep Probability	0.75	0.5	0.7
Taille du RNN	512	1024	128
Taille du Word Embedding	512	1024	128

Avec :

- Nombre d'époques : nombre d'itérations de la phase d'apprentissage.
- Taille du batch : taille du lot passé en entrée au réseau.
- Nombre de couches : nombre de couches LSTM qui constituent le réseau.

- Taux d'apprentissage : définit l'ajustement des poids par rapport au gradient de perte.
- Keep Probability : valeur utilisée pour contrôler le taux du Dropout appliqué lors de l'apprentissage du réseau.
- Taille du RNN : nombre des entrées de l'encodeur.
- Taille du Embedding : taille des Word Embeddings.

Pour réaliser cette évaluation, nous utilisons la mesure du BLEU Score définie dans la section précédente, en utilisant des 1, 2, 3 et 4-gramme et en appliquant ces trois configurations sur l'ensemble de validation du dataset, qui est composé de 247 paires de question-réponse. Un BLEU score proche de 0 signifie que le score est mauvais, tandis qu'un score proche de 100 signifie que le score est bon. Les résultats de l'évaluation sont présentés dans le tableau ci-dessous :

TABLEAU 5.5- Résultats de l'évaluation du système conversationnel avec BLEU Score.

Configuration	BLEU	1-gramme	2-gramme	3-gramme	4-gramme
[1]	24.03	49.65	42.53	35.43	24.03
[2]	33.21	55.05	49.83	43.38	33.21
[3]	20.19	45.51	39.29	31.49	20.19

Comme nous pouvons le constater, la deuxième *[2]* configuration du modèle surpasse les deux autres configurations et obtient le BLEU score le plus élevé. Ces résultats peuvent être justifiés comme suit :

- Certains articles suggèrent d'utiliser une taille de batch plus grande si la mémoire du système peut les prendre en charge, ce qui n'est pas le cas de notre machine qui dispose de performances limitées. D'autres recommandent également de modifier la taille du batch plutôt que de modifier le taux d'apprentissage. Néanmoins, un taux d'apprentissage élevé peut être utilisé lorsque la taille du lot est importante. Seulement, une taille de lot réduite permet d'obtenir une faible valeur finale de perte, ce qui a été le cas pour la configuration *[2]*.
- Un modèle contenant un nombre important de neurones par couche est connu pour augmenter la précision et la capacité de représentation du

modèle [108]. La configuration *[2]* comporte le plus grand nombre de neurones par couche, et par conséquent la taille d'Embedding la plus élevée car elle doit correspondre au nombre de neurones par couche.

- Appliquer un Dropout sur les unités neuronales moins souvent offre aux unités plus d'opportunités de participer et de « conspirer » les unes avec les autres pour s'adapter à l'ensemble d'apprentissage. Le choix du paramètre de Dropout pour la configuration *[2]* s'appuie sur l'article de Srivastava et al. [107], qui démontre que l'application d'un Dropout de 50% sur les unités cachées du réseau s'est dévoilé être le plus optimal.
- Le nombre de couches cachées d'un réseau de neurones est en général fixé d'une à trois couches. Habituellement, un réseau de neurones à trois couches cachées est plus efficace qu'un réseau à une ou deux couches. Mais augmenter encore plus le nombre de couches cachées apporte rarement plus de précision [108].
- Le taux d'apprentissage est connu pour être l'hyperparamètre le plus important. Il n'existe pas de relation simple entre ce taux et la capacité du modèle, mais généralement, si une valeur correcte est fixée pour le taux d'apprentissage alors la capacité du modèle sera maximisée. Cette valeur est généralement liée aux données du dataset.
- Le manque de travaux relatifs aux systèmes de dialogue en utilisant le même corpus et la même mesure d'évaluation, ne nous a pas permis de comparer notre modèle aux travaux de la littérature.

4.2. Mesures d'évaluation pour le classifieur de sentiments

Dans cette section nous allons dans un premier temps présenter la métrique d'évaluation que nous avons choisi pour le système conversationnel, pour ensuite discuter les résultats de l'évaluation obtenus en utilisant cette métrique.

4.2.1. Précision, Rappel et F-mesure

Dans les systèmes de recherche d'informations, de reconnaissance de forme et de classification binaire (comme le cas de l'analyse des sentiments), il existe plusieurs mesures à envisager pour évaluer la performance et l'efficacité du système. Parmi ces mesures nous retrouvons : *la précision, le rappel et la F-mesure*. La précision représente la proportion des cas pertinents parmi les instances récupérées, tandis que

le rappel est la fraction des instances pertinentes récupérées par rapport à l'ensemble total des instances pertinentes. La F-mesure représente enfin la moyenne harmonique de la précision et du rappel.

- **Précision :** c'est une mesure qui quantifie le nombre de prédictions positives correctes effectuées. Elle calculée en divisant le nombres d'échantillons positifs correctement prédits par le nombre total d'échantillons positifs prédits. La formule de la précision est la suivante :

$$Precision = \frac{VraiPositifs}{VraiPositifs + FauxPositifs} \quad (5.3)$$

- **Rappel :** il s'agit d'une mesure qui quantifie le nombre de prédictions positives correctes faites sur la base de toutes les prédictions positives qui auraient pu être faites. A l'opposé de la précision qui ne prend en considération que les prédictions positives correctes parmi toutes les prédictions positives, le rappel fournit une indication des prédictions positives manquantes. Ainsi, le rappel permet une certaine couverture de la classe positive grâce à sa formule :

$$Rappel = \frac{VraiPositifs}{VraiPositifs + FauxNegatifs} \quad (5.4)$$

- **F-mesure :** elle permet de combiner la précision et le rappel en une seule mesure qui inclut les deux propriétés pour les exprimer en un seul score, grâce à la somme harmonique des deux comme le montre la formule suivante :

$$F - mesure = \frac{2 \times Precision \times Rappel}{Precision + Rappel} \quad (5.5)$$

4.2.2. Evaluation du classifieur de sentiments

Afin d'évaluer notre modèle de classification de sentiments, nous utilisons les mesures de précision, rappel et F-mesure. Pour ce faire, nous utilisons l'ensemble de validation de notre dataset pour calculer les scores. Dans le but de comparer les résultats de cette évaluation, nous nous basons sur le travail de Tarımer et al. [109] qui ont implémenté trois modèles différents en les entrainant sur le même dataset que nous avons utilisé. Ces modèles consistent en un arbre de décision, un classifieur Bayes naïf et une machine à vecteurs de supports. Les résultats de cette comparaison sont mentionnés dans le tableau ci-dessous :

TABLEAU 5.6 - Résultats de la comparaison du classifieur de sentiments avec les travaux existants.

Modèles	Précision	Rappel	F-mesure
Notre modèle	0.675	0.79	0.728
Arbre de décision	0.75	0.755	0.75
Bayes naïf	0.715	0.715	0.71
SVM	0.727	0.726	0.725

Nous pouvons constater que les résultats obtenus par notre modèle sont satisfaisants dans l'ensemble, bien que sa valeur de précision soit la plus basse. Pour mettre encore plus en évidence nos résultats, nous les comparons à un autre travail réalisé par Patel et Tiwari [110] qui utilisent un réseau de neurones récurrent pour classifier les sentiments du même dataset que nous avons utilisé. Les résultats sont mentionnés ci-dessous :

TABLEAU 5.7 - Résultats de la comparaison du classifieur de sentiments à un modèle similaire.

Modèles	Précision	Rappel	F-mesure
Notre modèle	0.675	0.79	0.728
RNN	0.875	0.871	0.873

Comme nous pouvons le voir, les résultats obtenus par le modèle RNN sont plus élevés que les résultats obtenus par notre modèle, bien que l'architecture ne soit pas très différente. Ceci est dû au nombre insuffisant d'itérations pour entrainer notre modèle, puisque les performances de notre machine sont limitées. Pour améliorer cela, nous devons alors augmenter le nombre d'itérations pour entrainer notre modèle davantage et ainsi obtenir des résultats meilleurs.

5. Discussion

En combinant la configuration *[2]* du modèle utilisé dans le système conversationnel avec le classifieur de sentiments, nous avons pu atteindre l'objectif principal de ce travail. Cependant, les résultats obtenus devront être améliorés en utilisant un algorithme de réglage (fine-tuning) permettant de sélectionner les

meilleurs hyperparamètres pour notre modèle conversationnel pour ensuite l'entrainer davantage. Notre classifieur de sentiments devra également subir des itérations d'entrainement supplémentaires afin d'optimiser les résultats. La figure ci-dessous représente une conversation que nous avons entretenu avec notre chatbot « iBuy ». Comme nous pouvons le constater, les réponses du chatbot devront être améliorées grâce à l'utilisation d'un corpus plus spécialisé dans le service client.

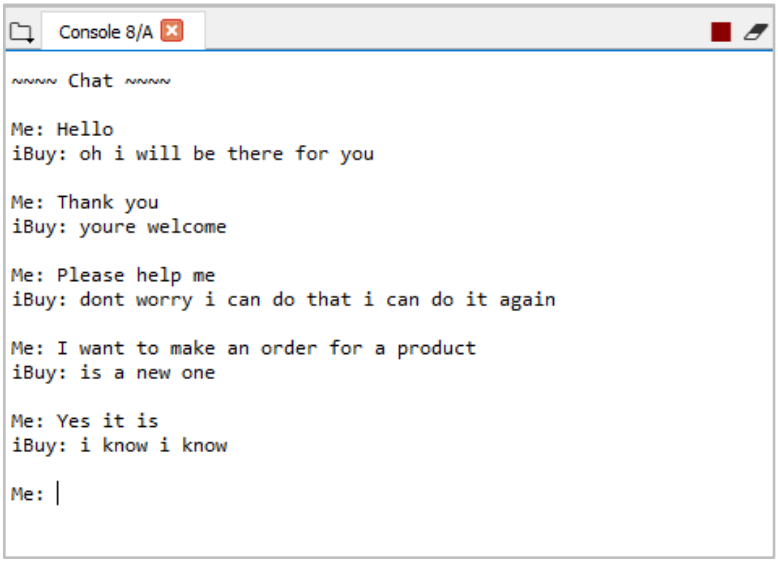

FIGURE 5.15 - Exemple de conversation.

Nos deux sous-systèmes agissent à l'unisson pour communiquer avec l'utilisateur et indiquer si ce dernier vit une expérience positive ou négative grâce à l'analyse des sentiments. L'intérêt d'un tel chatbot est d'aider les entreprises à économiser les coûts du service client en accélérant les temps de réponse et en répondant à la plupart des questions de routine. Certaines améliorations pouvant être apportées pour exploiter au mieux l'analyse des sentiments dans notre chatbot de service clientèle sont les suivantes :

- Suggérer un transfert vers un agent humain si le sentiment obtenu après avoir analysé la requête de l'utilisateur est négatif.
- Utiliser le sentiment obtenu pour déterminer si une compagne de marketing est bien interceptée par les clients. Les outils d'analyse marketing traditionnels ne sont pas aussi performants et précis que l'analyse des sentiments pour prédire l'état émotionnel exact du client.
- Désigner des ambassadeurs de la marque en déterminant les clients ayant le score de sentiments le plus élevé.

6. Conclusion

Ce dernier chapitre a été consacré à la présentation de l'implémentation de notre système, en commençant par les ressources matérielles et logicielles utilisées. Nous avons par la suite décrit l'architecture de notre système de façon technique. Nous avons également présenté les différentes mesures choisies pour évaluer nos deux sous-systèmes. Pour conclure, nous avons exposé les résultats obtenus après cette évaluation, que nous avons, par la suite, discuté.

Conclusion Générale

Grâce aux progrès de l'intelligence artificielle, le domaine du service client a été transformé. Cette révolution éminente de l'expérience client est particulièrement évidente dans la nouvelle vague d'agents conversationnels. Les chatbots fondés sur l'apprentissage automatique avec analyse des sentiments sont devenus en mesure de traiter automatiquement les requêtes sans intervention manuelle et en temps opportun.

Par conséquent, notre travail vise à mettre en place un agent conversationnel axé sur les données et doté d'analyse des sentiments en se basant sur l'apprentissage profond. Cependant, nous avons rencontré une difficulté majeure au cours de cette tâche : aucun corpus conversationnel annoté par des labels de sentiments n'est disponible. Cette contrainte a dû être prise en considération pour l'implémentation de notre solution et nous a conduits à diviser notre système global en deux sous-systèmes, à savoir un système conversationnel et un classifieur de sentiments. L'architecture du système conversationnel est basée sur un modèle Séquence à Séquence muni d'un mécanisme d'attention, tandis que l'architecture du classifieur de sentiments se base sur un Réseau de Neurones Récurrent constitué de cellules LSTM.

Afin de mener des évaluations sur nos deux sous-systèmes, nous avons pour chacun d'entre eux choisi certaines métriques d'évaluation. Pour estimer l'efficacité du système conversationnel, nous avons opté pour la mesure du BLEU Score que nous avons appliqué à trois différentes configurations préalablement choisies et entrainées. Les résultats de cette comparaison montrent que l'une de ces trois configurations s'est avérée particulièrement efficace. En effet, le BLEU Score obtenu pour la configuration *[2]* est de *33.21*, ce qui est supérieur aux deux autres configurations, ayant obtenu des scores de *24.03* et *20.19*. Quant au classifieur de sentiments, nous avons calculé la Précision, le Rappel ainsi que la F-mesure de notre modèle. Nous avons par la suite comparé nos résultats aux résultats obtenus dans d'autres travaux de recherche. Pour commencer, nous avons utilisé trois différents types d'analyse de sentiments pour les comparer à notre modèle : la valeur de Rappel de notre modèle (*0.79*) dépasse les autres, mais les valeurs de Précision et de F-mesure sont légèrement inférieures (*0.67*

et *0.728* respectivement). Ensuite, nous avons mis en balance notre modèle à un modèle RNN. Les résultats obtenus par ce dernier sont légèrement supérieurs au notre. Néanmoins, ces comparaisons ont permis de voir que la performance de notre modèle d'analyse des sentiments est tout de même satisfaisante.

Malgré les résultats obtenus, plusieurs améliorations et perspectives sont envisageables. Parmi ces perspectives futures, nous proposons de rajouter une couche à notre système conversationnel permettant de détecter le sentiment présent dans une requête, et ce, si un corpus comportant des annotations de sentiments sera disponible d'ici là. Ceci va permettre à l'agent conversationnel d'adapter ses réponses en fonction du sentiment de l'utilisateur. Pour optimiser l'expérience client et mettre en valeur le classifieur de sentiments de notre agent conversationnel au profit du service client, nous suggérons les améliorations suivantes :

- Exploiter les sentiments obtenus après l'analyse sur plusieurs messages d'utilisateurs pour mesurer le degré de satisfaction des clients par rapport à une campagne de marketing.
- Si le sentiment obtenu après analyse de la requête du client est négatif, recommander un transfert vers un agent humain.
- Nommer des ambassadeurs de la marque[13] en déterminant quels clients ont les scores émotionnels les plus élevés.

Pour conclure, nous pouvons dire que ce travail nous a été extrêmement bénéfique. Il nous a permis d'exploiter les connaissances acquises tout au long de notre parcours universitaire en les renforçant et en les enrichissant davantage. Nous avons également pu découvrir de nouveaux horizons et nous familiariser avec les domaines de l'apprentissage profond et du traitement automatique du langage naturel.

[13] **Ambassadeur de marque :** en marketing, un ambassadeur est une personne qui réalise la promotion d'une marque, d'un produit ou d'une entreprise de manière spontanée.

Bibliographie

[1] G. J. Kim, *Human-computer interaction: fundamentals and practice*. Boca Raton: CRC Press, 2015.

[2] T. F. Waddell, B. Zhang, et S. S. Sundar, « Human–Computer Interaction », in *The International Encyclopedia of Interpersonal Communication*, American Cancer Society, 2015, p. 1-9.

[3] J. A. Jacko, *Human Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications, Third Edition*. CRC Press, 2012.

[4] M. Kurosu, *Human-Computer Interaction. User Interface Design, Development and Multimodality: 19th International Conference, HCI International 2017, Vancouver, BC, Canada, July 9-14, 2017, Proceedings*. Springer, 2017.

[5] B. Hancock, A. Bordes, P.-E. Mazaré, et J. Weston, « Learning from Dialogue after Deployment: Feed Yourself, Chatbot! », *arXiv:1901.05415 [cs, stat]*, juin 2019,. [En ligne]. Disponible sur: http://arxiv.org/abs/1901.05415.

[6] M. Dahiya, « A Tool of Conversation: Chatbot », *International Journal of Computer Sciences and Engineering*, p. 5, 2017.

[7] J. Jia, « The Study of the Application of a Keywords-based Chatbot System on the Teaching of Foreign Languages », p. 11.

[8] S. A. et Dr. John, « Survey on Chatbot Design Techniques in Speech Conversation Systems », *ijacsa*, vol. 6, n° 7, 2015, doi: 10.14569/IJACSA.2015.060712.

[9] S. Hussain, O. Ameri Sianaki, et N. Ababneh, « A Survey on Conversational Agents/Chatbots Classification and Design Techniques », in *Web, Artificial Intelligence and Network Applications*, Cham, 2019, p. 946-956, doi: 10.1007/978-3-030-15035-8_93.

[10] R. Rzepka, M. Ptaszynski, et P. Dybala, « LINGUISTIC AND COGNITIVE APPROACHES TO DIALOGUE AGENTS », p. 73.

[11] S. Leggeri, A. Esposito, et L. Iocchi, « Task-oriented Conversational Agent Self-learning Based on Sentiment Analysis », p. 12.

[12] Z. Yu, L. Nicolich-Henkin, A. W. Black, et A. Rudnicky, « A Wizard-of-Oz Study on A Non-Task-Oriented Dialog Systems That Reacts to User Engagement », in *Proceedings*

of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue, Los Angeles, 2016, p. 55-63, doi: 10.18653/v1/W16-3608.

[13] T. Zemčík, « A Brief History of Chatbots », *DEStech Transactions on Computer Science and Engineering*, oct. 2019, doi: 10.12783/dtcse/aicae2019/31439.

[14] J.-P. Delahaye, « L'intelligence artificielle et le test de Turing », p. 3.

[15] S. Diederich, A. Brendel, et L. Kolbe, « On Conversational Agents in Information Systems Research: Analyzing the Past to Guide Future Work », *Wirtschaftsinformatik 2019 Proceedings*, févr. 2019, [En ligne]. Disponible sur: https://aisel.aisnet.org/wi2019/track13/papers/1.

[16] H. Shum, X. He, et D. Li, « From Eliza to XiaoIce: challenges and opportunities with social chatbots », *Frontiers Inf Technol Electronic Eng*, vol. 19, nº 1, p. 10-26, janv. 2018, doi: 10.1631/FITEE.1700826.

[17] K. K. Fitzpatrick, A. Darcy, et M. Vierhile, « Delivering Cognitive Behavior Therapy to Young Adults With Symptoms of Depression and Anxiety Using a Fully Automated Conversational Agent (Woebot): A Randomized Controlled Trial », *JMIR Ment Health*, vol. 4, nº 2, p. e19, juin 2017, doi: 10.2196/mental.7785.

[18] B. AbuShawar et E. Atwell, « ALICE Chatbot: Trials and Outputs », *Computación y Sistemas*, vol. 19, nº 4, Art. nº 4, déc. 2015, doi: 10.13053/cys-19-4-2326.

[19] S. Kaushik, D. Gupta, L. Kharb, et D. Chahal, *Information, Communication and Computing Technology: Second International Conference, ICICCT 2017, New Delhi, India, May 13, 2017, Revised Selected Papers*. Springer, 2017.

[20] P.-M. Diana, *Conversational Agents and Natural Language Interaction: Techniques and Effective Practices: Techniques and Effective Practices*. IGI Global, 2011.

[21] D. Fensel *et al.*, *Knowledge Graphs: Methodology, Tools and Selected Use Cases*. Springer Nature, 2020.

[22] J. Bockhorst, D. Conathan, et G. Fung, « Knowledge Graph-Driven Conversational Agents », p. 12.

[23] B. Hixon, P. Clark, et H. Hajishirzi, « Learning Knowledge Graphs for Question Answering through Conversational Dialog », in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Denver, Colorado, 2015, p. 851-861, doi: 10.3115/v1/N15-1086.

[24] S. D. Emidio, G. Temperman, et B. D. Lièvre, « Manier l'intelligence artificielle sans coder : création de chatbots éducatifs », p. 4.

[25] C. Smith, M. Cavazza, D. Charlton, L. Zhang, M. Turunen, et J. Hakulinen, « Integrating Planning and Dialogue in a Lifestyle Agent », in *Intelligent Virtual Agents*, Berlin, Heidelberg, 2008, p. 146-153, doi: 10.1007/978-3-540-85483-8_15.

[26] A. Venkatesh *et al.*, « On Evaluating and Comparing Conversational Agents », p. 10.

[27] N. Radziwill et M. Benton, « Evaluating Quality of Chatbots and Intelligent Conversational Agents », p. 21.

[28] L. Bradeško et D. Mladenić, « A Survey of Chabot Systems through a Loebner Prize Competition », p. 4.

[29] K. Isbister et P. Doyle, « Design and Evaluation of Embodied Conversational Agents: A Proposed Taxonomy », p. 6.

[30] M. A. Walker, R. Passonneau, et J. E. Boland, « Quantitative and qualitative evaluation of Darpa Communicator spoken dialogue systems », in *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, Toulouse, France, juill. 2001, p. 515–522, doi: 10.3115/1073012.1073078.

[31] L. Dybkjær, N. O. Bernsen, et W. Minker, « Evaluation and usability of multimodal spoken language dialogue systems », *Speech Communication*, vol. 43, n° 1, p. 33-54, juin 2004, doi: 10.1016/j.specom.2004.02.001.

[32] O. S. Goh, C. Ardil, W. Wong, et C. C. Fung, « A Black-box Approach for Response Quality Evaluation of Conversational Agent Systems », p. 9, 2007.

[33] M. A. Walker, D. J. Litman, C. A. Kamm, et A. Abella, « PARADISE: A Framework for Evaluating Spoken Dialogue Agents », p. 10.

[34] J. Bradley, D. Benyon, O. Mival, et N. Webb, « Wizard of Oz Experiments and Companion Dialogues », présenté à Proceedings of HCI 2010, sept. 2010, doi: 10.14236/ewic/HCI2010.16.

[35] H. Wei, Y. Zhao, et J. Ke, « Building Chatbot with Emotions », p. 8.

[36] C.-W. Lee, Y.-S. Wang, T.-Y. Hsu, K.-Y. Chen, H.-Y. Lee, et L. Lee, « Scalable Sentiment for Sequence-to-sequence Chatbot Response with Performance Analysis », *arXiv:1804.02504 [cs]*, avr. 2018, [En ligne]. Disponible sur: http://arxiv.org/abs/1804.02504.

[37] S. Leggeri, A. Esposito, et L. Iocchi, « Task-oriented Conversational Agent Self-learning Based on Sentiment Analysis », p. 12.

[38] H. Zhou, M. Huang, T. Zhang, X. Zhu, et B. Liu, « Emotional Chatting Machine: Emotional Conversation Generation with Internal and External Memory »,

arXiv:1704.01074 [cs], mai 2018. [En ligne]. Disponible sur: http://arxiv.org/abs/1704.01074.

[39] A. D'Andrea, F. Ferri, P. Grifoni, et T. Guzzo, « Approaches, Tools and Applications for Sentiment Analysis Implementation », *IJCA*, vol. 125, nº 3, p. 26-33, sept. 2015, doi: 10.5120/ijca2015905866.

[40] H. Chen et D. Zimbra, « AI and Opinion Mining », *IEEE Intelligent Systems*, vol. 25, nº 3, p. 74-80, mai 2010, doi: 10.1109/MIS.2010.75.

[41] B. Pang et L. Lee, « Opinion mining and sentiment analysis », p. 94.

[42] B. Liu, « Sentiment Analysis and Opinion Mining », p. 168.

[43] H. Saif, *Semantic Sentiment Analysis in Social Streams*. IOS Press, 2017.

[44] B. Agarwal et N. Mittal, « Semantic Orientation-Based Approach for Sentiment Analysis », 2016, p. 77-88.

[45] B. Azzeddine, A. Harbaoui, et H. Ben Ghezala, « Sentiment Analysis Approaches based on Granularity Levels », janv. 2018, p. 324-331, doi: 10.5220/0007187603240331.

[46] K. Schouten et F. Frasincar, « Survey on Aspect-Level Sentiment Analysis », *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, nº 3, p. 813-830, mars 2016, doi: 10.1109/TKDE.2015.2485209.

[47] S. B. Moralwar et S. N. Deshmukh, « Different Approaches of Sentiment Analysis », vol. 3, p. 6, 2015.

[48] « Sentiment analysis algorithms and applications: A survey | Elsevier Enhanced Reader ».
https://reader.elsevier.com/reader/sd/pii/S2090447914000550?token=2628C73C9F13FECB412AE816D9DA4AB057F4EDE28A0AED267F250129AD66B89CE5F011065FD14BFC6F0BA7A053714DCC.

[49] A. L. Samuel, « Some Studies in Machine Learning Using the Game of Checkers », *IBM Journal of Research and Development*, vol. 3, nº 3, p. 210-229, juill. 1959, doi: 10.1147/rd.33.0210.

[50] B. Agarwal et N. Mittal, « Machine Learning Approach for Sentiment Analysis », in *Prominent Feature Extraction for Sentiment Analysis*, B. Agarwal et N. Mittal, Éd. Cham: Springer International Publishing, 2016, p. 21-45.

[51] V. A. et S. S. Sonawane, « Sentiment Analysis of Twitter Data: A Survey of Techniques », *IJCA*, vol. 139, nº 11, p. 5-15, avr. 2016, doi: 10.5120/ijca2016908625.

[52] A. Suresh et C. Bharathi, « Sentiment Classifi cation using Decision Tree Based Feature Selection », *International Journal of Control Theory and Applications*, vol. 9, p. 419-425, janv. 2016.

[53] T. Mullen et N. Collier, « Sentiment analysis using support vector machines with diverse information sources », p. 7.

[54] T. Phienthrakul, B. Kijsirikul, H. Takamura, et M. Okumura, « Sentiment Classification with Support Vector Machines and Multiple Kernel Functions », in *Neural Information Processing*, Berlin, Heidelberg, 2009, p. 583-592, doi: 10.1007/978-3-642-10684-2_65.

[55] D. Sharma, M. Sabharwal, V. Goyal, et M. Vij, « Sentiment Analysis Techniques for Social Media Data: A Review », in *First International Conference on Sustainable Technologies for Computational Intelligence*, Singapore, 2020, p. 75-90, doi: 10.1007/978-981-15-0029-9_7.

[56] O. Irsoy et C. Cardie, « Opinion Mining with Deep Recurrent Neural Networks », in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014, p. 720-728, doi: 10.3115/v1/D14-1080.

[57] K. Suppala et N. Rao, « Sentiment Analysis Using Naïve Bayes Classifier », vol. 8, n° 8, p. 6, 2019.

[58] L. Gutiérrez, J. Bekios-Calfa, et B. Keith Norambuena, « A Review on Bayesian Networks for Sentiment Analysis: Proceedings of the 7th International Conference on Software Process Improvement (CIMPS 2018) », 2019, p. 111-120.

[59] M. S. Usha et I. D. Karthikeyan, « Analysis of sentiments using unsupervised learning techniques », févr. 2013, p. 241-245, doi: 10.1109/ICICES.2013.6508203.

[60] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, et M. Stede, « Lexicon-Based Methods for Sentiment Analysis », *Computational Linguistics*, vol. 37, n° 2, p. 267-307, juin 2011, doi: 10.1162/COLI_a_00049.

[61] H. Han, Y. Zhang, J. Zhang, J. Yang, et X. Zou, « Improving the performance of lexicon-based review sentiment analysis method by reducing additional introduced sentiment bias », *PLOS ONE*, vol. 13, n° 8, p. e0202523, août 2018, doi: 10.1371/journal.pone.0202523.

[62] S. Mohammad, « Challenges in Sentiment Analysis », 2017, p. 61-83.

[63] D. I. H. Farias et P. Rosso, « Chapter 7 - Irony, Sarcasm, and Sentiment Analysis », in *Sentiment Analysis in Social Networks*, F. A. Pozzi, E. Fersini, E. Messina, et B. Liu, Éd. Boston: Morgan Kaufmann, 2017, p. 113-128.

[64] D. G. Maynard et M. A. Greenwood, « Who cares about sarcastic tweets? Investigating the impact of sarcasm on sentiment analysis », p. 7.

[65] A. Joshi, P. Bhattacharyya, et M. J. Carman, « Automatic Sarcasm Detection: A Survey », *ACM Comput. Surv.*, vol. 50, nº 5, p. 73:1–73:22, sept. 2017, doi: 10.1145/3124420.

[66] L. Jia, C. Yu, et W. Meng, « The effect of negation on sentiment analysis and retrieval effectiveness », in *Proceedings of the 18th ACM conference on Information and knowledge management*, Hong Kong, China, nov. 2009, p. 1827–1830, doi: 10.1145/1645953.1646241.

[67] C. Sumanth et D. Inkpen, « How much does word sense disambiguation help in sentiment analysis of micropost data? », in *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Lisboa, Portugal, 2015, p. 115-121, doi: 10.18653/v1/W15-2916.

[68] U. Farooq, T. P. Dhamala, A. Nongaillard, Y. Ouzrout, et M. A. Qadir, « A word sense disambiguation method for feature level sentiment analysis », in *2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, déc. 2015, p. 1-8, doi: 10.1109/SKIMA.2015.7399988.

[69] R. Nithya, « Need for Anaphoric Resolution towards Sentiment Analysis-A Case Study with Scarlet Pimpernel (Novel) », *International Journal of Education and Management Engineering*, vol. 9, p. 37-50, janv. 2019, doi: 10.5815/ijeme.2019.01.04.

[70] V. Stoyanov et C. Cardie, « Partially supervised coreference resolution for opinion summarization through structured rule learning », in *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing - EMNLP '06*, Sydney, Australia, 2006, p. 336, doi: 10.3115/1610075.1610123.

[71] X. Ding et B. Liu, « Resolving Object and Attribute Coreference in Opinion Mining », p. 9.

[72] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

[73] E. Alpaydin, *Introduction to Machine Learning*. MIT Press, 2020.

[74] L. Bottou, F. E. Curtis, et J. Nocedal, « Optimization Methods for Large-Scale Machine Learning », *SIAM Rev.*, vol. 60, nº 2, p. 223-311, janv. 2018, doi: 10.1137/16M1080173.

[75] R. Caruana et A. Niculescu-Mizil, « An empirical comparison of supervised learning algorithms », in *Proceedings of the 23rd international conference on Machine learning*, New York, NY, USA, juin 2006, p. 161–168, doi: 10.1145/1143844.1143865.

[76] W.-Y. Loh, « Classification and Regression Tree Methods », in *Wiley StatsRef: Statistics Reference Online*, American Cancer Society, 2014.

[77] H. B. Barlow, « Unsupervised Learning », *Neural Computation*, vol. 1, nº 3, p. 295-311, sept. 1989, doi: 10.1162/neco.1989.1.3.295.

[78] Z. Ghahramani, « Unsupervised Learning », in *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*, O. Bousquet, U. von Luxburg, et G. Rätsch, Éd. Berlin, Heidelberg: Springer, 2004, p. 72-112.

[79] X. Zhu et A. B. Goldberg, « Introduction to Semi-Supervised Learning », *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 3, nº 1, p. 1-130, janv. 2009, doi: 10.2200/S00196ED1V01Y200906AIM006.

[80] X. (Jerry) Zhu, « Semi-Supervised Learning Literature Survey », University of Wisconsin-Madison Department of Computer Sciences, Technical Report, 2005. [En ligne]. Disponible sur: https://minds.wisconsin.edu/handle/1793/60444.

[81] L. P. Kaelbling, M. L. Littman, et A. W. Moore, « Reinforcement Learning: A Survey », *Journal of Artificial Intelligence Research*, vol. 4, p. 237-285, mai 1996, doi: 10.1613/jair.301.

[82] M. Wiering et M. van Otterlo, Éd., *Reinforcement learning: state-of-the-art*. Heidelberg ; New York: Springer, 2012.

[83] P. Dayan et B. W. Balleine, « Reward, Motivation, and Reinforcement Learning », *Neuron*, vol. 36, nº 2, p. 285-298, oct. 2002, doi: 10.1016/S0896-6273(02)00963-7.

[84] L. Deng et D. Yu, « Deep Learning: Methods and Applications », *Found. Trends Signal Process.*, vol. 7, nº 3–4, p. 197–387, juin 2014, doi: 10.1561/2000000039.

[85] J. Schmidhuber, « Deep learning in neural networks: An overview », *Neural Networks*, vol. 61, p. 85-117, janv. 2015, doi: 10.1016/j.neunet.2014.09.003.

[86] H. D. Block, « The Perceptron: A Model for Brain Functioning. I », *Rev. Mod. Phys.*, vol. 34, nº 1, p. 123-135, janv. 1962, doi: 10.1103/RevModPhys.34.123.

[87] I. Goodfellow, Y. Bengio, et A. Courville, *Deep Learning*. MIT Press, 2016.

[88] D. Svozil, V. Kvasnicka, et J. Pospichal, « Introduction to multi-layer feed-forward neural networks », *Chemometrics and Intelligent Laboratory Systems*, vol. 39, nº 1, p. 43-62, nov. 1997, doi: 10.1016/S0169-7439(97)00061-0.

[89] « Structure and function of the feed-forward loop network motif | PNAS ». https://www.pnas.org/content/100/21/11980.short.

[90] R. Hecht-nielsen, « III.3 - Theory of the Backpropagation Neural Network**Based on “nonindent” by Robert Hecht-Nielsen, which appeared in Proceedings of the International Joint Conference on Neural Networks 1, 593–611, June 1989. © 1989 IEEE. », in *Neural Networks for Perception*, H. Wechsler, Éd. Academic Press, 1992, p. 65-93.

[91] Y. C. (Ph D.) et D. E. Rumelhart, *Backpropagation: Theory, Architectures, and Applications*. Psychology Press, 1995.

[92] S. Du, J. Lee, H. Li, L. Wang, et X. Zhai, « Gradient Descent Finds Global Minima of Deep Neural Networks », in *International Conference on Machine Learning*, mai 2019, p. 1675-1685. [En ligne]. Disponible sur: http://proceedings.mlr.press/v97/du19c.html.

[93] S. Hochreiter, A. S. Younger, et P. R. Conwell, « Learning to Learn Using Gradient Descent », in *Artificial Neural Networks — ICANN 2001*, Berlin, Heidelberg, 2001, p. 87-94, doi: 10.1007/3-540-44668-0_13.

[94] W. H. Lopez Pinaya, S. Vieira, R. Garcia-Dias, et A. Mechelli, « Chapter 10 - Convolutional neural networks », in *Machine Learning*, A. Mechelli et S. Vieira, Éd. Academic Press, 2020, p. 173-191.

[95] R. Yamashita, M. Nishio, R. K. G. Do, et K. Togashi, « Convolutional neural networks: an overview and application in radiology », *Insights Imaging*, vol. 9, n° 4, p. 611-629, août 2018, doi: 10.1007/s13244-018-0639-9.

[96] *Recurrent Neural Networks*. .

[97] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, et S. Valaee, « Recent Advances in Recurrent Neural Networks », *arXiv:1801.01078 [cs]*, févr. 2018. [En ligne]. Disponible sur: http://arxiv.org/abs/1801.01078.

[98] S. Hochreiter et J. Schmidhuber, « Long Short-term Memory », *Neural computation*, vol. 9, p. 1735-80, déc. 1997, doi: 10.1162/neco.1997.9.8.1735.

[99] A. Graves, « Long Short-Term Memory », in *Supervised Sequence Labelling with Recurrent Neural Networks*, A. Graves, Éd. Berlin, Heidelberg: Springer, 2012, p. 37-45.

[100] L. Lu, X. Zhang, K. Cho, et S. Renals, « A Study of the Recurrent Neural Network Encoder-Decoder for Large Vocabulary Speech Recognition », p. 5.

[101] K. Cho *et al.*, « Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation », *arXiv:1406.1078 [cs, stat]*, sept. 2014. [En ligne]. Disponible sur: http://arxiv.org/abs/1406.1078.

[102] I. Sutskever, O. Vinyals, et Q. V. Le, « Sequence to Sequence Learning with Neural Networks », in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani,

M. Welling, C. Cortes, N. D. Lawrence, et K. Q. Weinberger, Éd. Curran Associates, Inc., 2014, p. 3104–3112.

[103] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, et Y. Bengio, « Attention-Based Models for Speech Recognition », in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, et R. Garnett, Éd. Curran Associates, Inc., 2015, p. 577–585.

[104] Y. Wang et M. Huang, « Attention-based LSTM for Aspect-level Sentiment Classification », p. 10.

[105] J. Pennington, R. Socher, et C. Manning, « Glove: Global Vectors for Word Representation », p. 12.

[106] D. P. Kingma et J. Ba, « Adam: A Method for Stochastic Optimization », déc. 2014. [En ligne]. Disponible sur: https://arxiv.org/abs/1412.6980v9.

[107] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, et R. Salakhutdinov, « Dropout: A Simple Way to Prevent Neural Networks from Overfitting », p. 30.

[108] D. Stathakis, « How many hidden layers and nodes? », *International Journal of Remote Sensing*, vol. 30, nº 8, p. 2133-2147, avr. 2009, doi: 10.1080/01431160802549278.

[109] İ. Tarımer, A. Çoban, et A. E. Kocaman, « Sentiment Analysis on IMDB Movie Comments and Twitter Data by Machine Learning and Vector Space Techniques », p. 8.

[110] A. Patel et A. K. Tiwari, « Sentiment Analysis by using Recurrent Neural Network », Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 3349572, févr. 2019. doi: 10.2139/ssrn.3349572.