Sommaire

Présentation générale

Un chatbot est un programme informatique conçu pour simuler la conversation avec des utilisateurs humains, en particulier sur Internet. Les Chatbots peuvent être déployés sur Facebook Messenger, mais aussi par exemple sur une page web de votre site, sur Twitter, sur une application mobile ou sur un assistant personnel commandé par la voix (ex. Google Home ou smartphones Android).

Aujourd'hui, ils sont de plus en plus présents dans nos vies quotidiennes. Les plus grandes entreprises possèdent toutes leur chatbot, souvent représenté sous la forme d'un personnage animé ou plus simplement d'une boîte de dialogue où les clients et les prospects peuvent poser leurs questions.

Désormais, les chatbots peuvent être utilisés pour différentes tâches comme faciliter une vente ou effectuer un paiement, fournir un service client 24h/7j ou augmenter l'engagement sur les réseaux sociaux. Ce sont des outils puissants pour la gestion des réponses aux internautes et clients et par conséquent pour l'amélioration de l'e-réputation de l'entreprise.

Il existe deux grands types de chatbots :

- Des chatbots "basiques", basés sur des règles. Simples à développer et à déployer, ils atteignent très vite leur limite, tant sur le plan des fonctionnalités que sur leur capacité à dialoguer ;
- Des "chatbots intelligents", appuyés sur l'apprentissage automatique (particulièrement la technique du deep learning) et de reconnaissance naturelle du langage (NLP – Natural Language Processing) ;

L'objectif de ce projet est le développement d'un chatbot intelligent pour la gestion des réponses aux clients et internautes et ayant pour finalité l'amélioration de l'e-réputation de l'entreprise.

Introduction générale

Avec l'internet constamment à portée de main, la gestion de la réputation en ligne d'une entreprise peut être un travail à plein temps. Pour rendre la tâche encore plus difficile, les consommateurs s'attendent à ce que les entreprises répondent rapidement à leurs questions et préoccupations, renforçant ainsi le fait que la gestion de la réputation est essentielle. La réputation peut également avoir un impact considérable sur les ventes des entreprises.

L'intelligence artificielle joue un rôle de premier plan dans la surveillance des plateformes de réseaux sociaux et d'autres forums d'évaluation des clients. Le partage social continue de s'améliorer sur différentes plateformes. En conséquence, les entreprises sont confrontées à un problème lorsqu'il s'agit de surveiller ce que les clients disent sur ces plateformes. Compte tenu des commentaires et des critiques accablants que reçoivent les entreprises, il est important de disposer d'un outil de surveillance avancé. Cela garantit que la réputation de l'entreprise n'est pas affectée en ligne.

Pour qu'une entreprise se prémunisse contre les atteintes à sa réputation dues à un service client médiocre, il est essentiel d'investir dans les chatbots et autres outils d'IA. Un chatbot est alimenté par des règles et une IA pour faciliter l'interaction avec les clients. Les chatbots effectuent des tâches automatisées et sont donc essentiels pour fournir un service client.

Notre objectif dans ce travail est d'étudier, de modéliser et de développer un chatbot qui permet d'améliorer la e-réputation d'une entreprise en utilisant les techniques de Deep Learning et de Traitement Automatique du Langage Naturel.

Notre travail sera divisé en trois parties, la première partie est menée à une étude bibliographique où nous parlons des chatbots et des œuvres associées ainsi que des techniques utilisées. La deuxième partie, nous allons modéliser et expliquer notre approche. La dernière partie concerne la réalisation et le développement, puis nous finirons par donner une perspective générale.

Étude bibliographique

CHATBOT

Introduction

Selon les prévisionnistes et les spécialistes du marketing du monde entier, nous vivons dans la décennie des chatbots et le meilleur reste à venir.
Avance rapide jusqu'à 50 ans plus tard et nous assistons à la montée en flèche du nombre de chatbots dans le monde. Les Chatbots font désormais partie intégrante de notre vie quotidienne. Que nous le réalisions ou non, ils deviennent nos assistants personnels, nos agents de service à la clientèle, notre aide pour trouver un parking, des cafés et des méthodes de travail alternatives et parfois, ils pourraient bien nous amuser.

Historique

En 1950, Alan Turing se demandait si un programme informatique pouvait parler à un groupe de personnes sans se rendre compte que leur interlocuteur était artificiel. Cette question, appelée test de Turing, est considérée par beaucoup comme l'idée générative des chatbots [1]. La pierre angulaire du développement des chatbots a été posée en 1966 par Joseph Weizenbaum. Les principes utilisés dans Eliza (le premier chatbot jamais conçu) ont été à la base des structures de développement des chatbots que nous connaissons aujourd'hui. Bien qu'Eliza ait pu tromper quelques utilisateurs en leur faisant croire qu'ils parlaient réellement à un humain, elle a échoué au test de Turing [2].

Depuis ELIZA, il y a eu des progrès dans le développement de chatbots de plus en plus intelligents. En 1972, Kenneth Colby à Stanford a créé PARRY, un robot schizophrène paranoïde imité [3]. L'intelligence artificielle est d'abord utilisée dans le domaine des chatbots avec la construction de Jabberwacky en 1988 [4], elle a utilisé le filtrage contextuel pour répondre en fonction des discussions précédentes. Pourtant, Jabberwacky ne peut pas répondre à la haute vitesse et travailler avec un grand nombre d'utilisateurs [5]. Dr Sbaitso (Sound Blaster Artificial Intelligent Text to Speech Operator), un chatbot créé en 1992, a été conçu pour afficher les voix numérisées que les cartes son étaient capables de produire [6]. Il a joué le rôle d'un psychologue sans aucune sorte d'interaction compliquée [7]. En 1995, Richard Wallace a créé A.L.I.C.E, un chatbot significativement plus complexe qui générait des réponses par correspondance de modèles d'entrées contre des paires <pattern> (input) <template> (output) stockées dans des documents dans une base de connaissances. Ces documents ont été rédigés en langage AIML (Artificial Intelligence Markup Language), une extension de XML, qui est toujours utilisée aujourd'hui [8].

En 2001, il y a eu une réelle évolution dans la technologie des chatbots avec le développement de SmarterChild [12], c'était la première fois qu'un chatbot pouvait aider les gens avec des tâches quotidiennes pratiques car il pouvait récupérer des informations dans des bases de données sur les horaires des films, résultats sportifs, cours boursiers, actualités et météo.

Le développement des chatbots à intelligence artificielle est allé plus loin avec la création d'assistants vocaux personnels intelligents, intégrés aux smartphones ou aux haut-parleurs dédiés à la maison, qui comprenaient les commandes vocales, parlaient par des voix numériques et gèrent des tâches telles que la surveillance des appareils automatisés à domicile, des calendriers, des e-mails autre. Apple Siri (Siri), IBM Watson, Google Assistant, Microsoft Cortana et Amazon Alexa sont les assistants vocaux les plus populaires. Il existe également de nombreux autres assistants vocaux moins connus ayant des caractéristiques uniques, mais les mêmes fonctions de base.

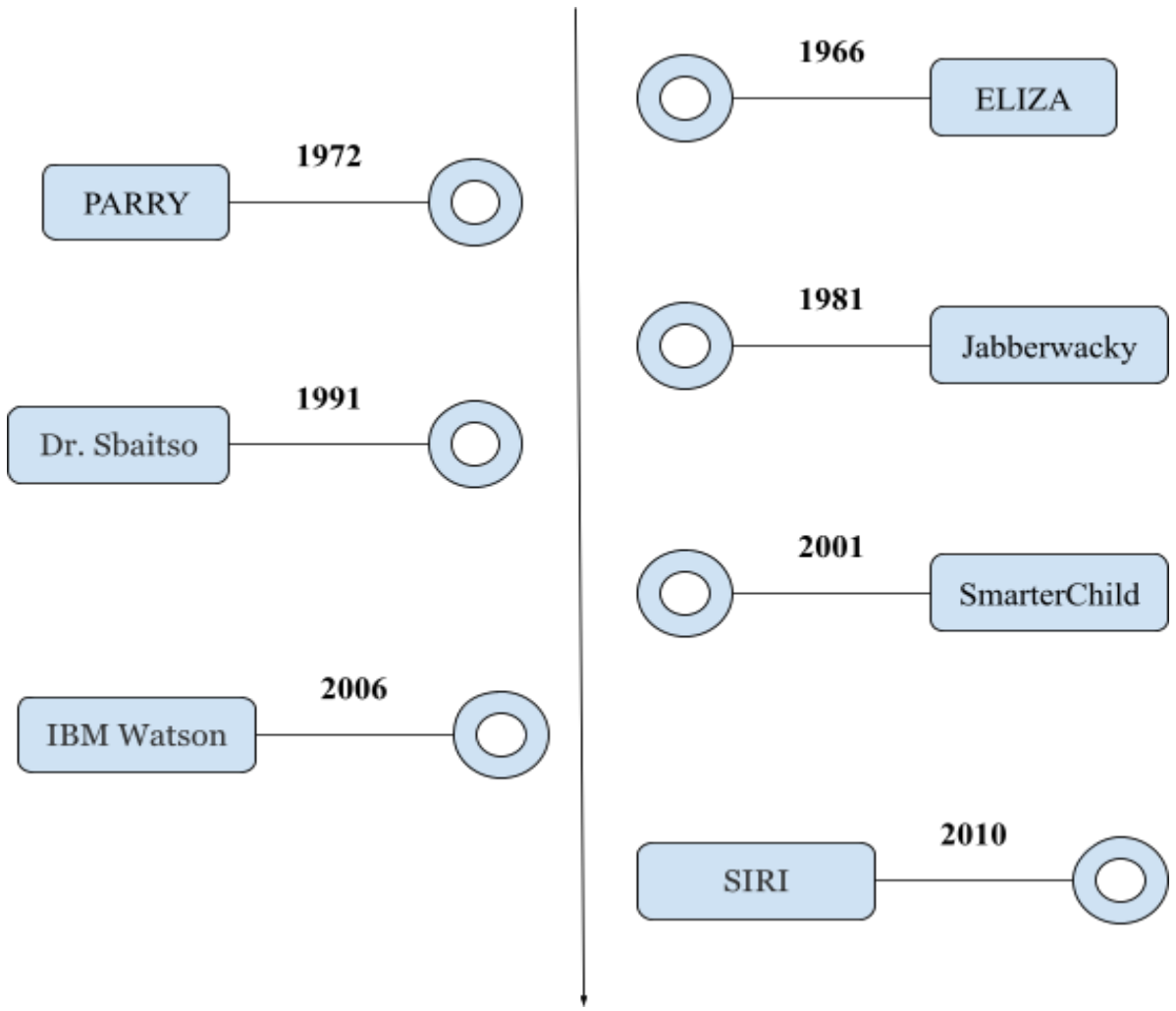


Figure 1. Brève histoire des chatbots [2]

Définitions

Les chatbots - également appelés «agents conversationnels» - sont des applications logicielles qui imitent la parole humaine (langage naturel) écrite ou parlée dans le but de simuler une conversation ou une interaction avec une personne réelle [13]. Aujourd'hui, les chatbots sont le plus souvent utilisés dans l'espace du service client, assumant des rôles traditionnellement joués par des êtres humains vivants et respirants. Le service pourrait être un certain nombre de choses, allant de fonctionnel à amusant, et il pourrait vivre dans n'importe quel produit de chat majeur.

Un chatbot est souvent décrit comme l'une des expressions les plus avancées et les plus prometteuses de l'interaction entre les humains et les machines. Cependant, d'un point de vue technologique, un chatbot ne représente que l'évolution naturelle d'un système de réponse aux questions tirant parti du traitement automatique du langage naturel (TALN). La formulation des réponses aux questions en langage naturel est l'un des exemples les plus typiques de traitement du langage naturel appliqué dans les applications finales de diverses entreprises [14].

Il existe un certain nombre de synonymes pour le chatbot, notamment "talkbot", "IM bot", "interactive agent" ou "artifice conversation entity".

Fonctionnent générale des chatbots

Il existe deux types de chatbots, l'un fonctionne basé sur un ensemble de règles et l'autre version plus avancée utilise l'apprentissage automatique et les techniques d'intelligence artificielle.

Chatbot qui fonctionne selon des règles

- Ce bot est très limité. Il ne peut répondre qu'à des commandes très spécifiques. Si vous dites une mauvaise chose, il ne sait pas ce que vous voulez dire.
- Ce bot est aussi intelligent qu'il est programmé pour l'être.

Un exemple de chatbot limité est un robot bancaire automatisé qui pose des questions à l'appelant pour comprendre ce que l'appelant veut faire.

Chatbot qui fonctionne à l'aide de l'intelligence artificiels

- Ce bot a un cerveau artificiel. Vous n'avez pas besoin d'être strictement précis lorsque vous lui parlez. Il comprend la langue, pas seulement les commandes.
- Ce bot devient continuellement plus intelligent à mesure qu'il apprend des conversations qu'il a avec les gens.

Au cœur de la technologie des chatbots basée sur l'intelligence artificielle se trouve le traitement automatique du langage naturel (TALN) [13]. Les chatbots traitent le texte qui leur est présenté par l'utilisateur (un processus appelé “analyse”), avant de répondre selon une série complexe d'algorithmes qui interprète et identifie ce que l'utilisateur a dit, en déduit ce qu'il veut dire et/ou ce qu'il veut, et déterminer un série de réponses appropriées basées sur ces informations [14] .

Donc, il y a deux tâches différentes au cœur d'un chatbot intelligent:

1. Analyse des demandes des utilisateurs.
2. Renvoyer la réponse.

Ce processus peut paraître simple; dans la pratique, les choses sont assez complexes.

1. Analyse des demandes des utilisateurs

C'est la première tâche qu'un chatbot effectue. Il analyse la demande de l’utilisateur pour identifier l’intention de l’utilisateur et extraire les entités pertinentes.

2. Renvoyer la réponse

Une fois l'intention de l'utilisateur identifiée, le chatbot doit fournir la réponse la plus appropriée à la demande de l'utilisateur. La réponse peut être:

- un texte générique et prédéfini.
- un texte extrait d'une base de connaissances contenant une réponse différente.
- une information contextualisée basée sur les données fournies par l'utilisateur.
- données stockées dans les systèmes d'entreprise.
- le résultat d'une action que le chatbot a effectuée en interagissant avec une ou plusieurs applications backend.
- Une question sans ambiguïté qui aide le chatbot à comprendre correctement la demande de l'utilisateur.

Attention cependant, les bots ont l'illusion de la simplicité sur le front-end, mais il y a de nombreux obstacles à surmonter pour créer une expérience formidable. Tant de travail à faire. Analytique, optimisation des flux, la vérification des erreurs, les intégrations aux API, la compréhension de le TALN etc. Nous devons désapprendre tout ce que nous avons appris par le passé pour créer une expérience incroyable dans ce nouveau navigateur [15].

Cas d'utilisation des chatbots

Avant de parler du cas d'utilisation des chatbots, voyons à quoi les gens pensent d'un chatbot [16]:

- Obtenir une réponse rapide dans une situation d'urgence.

- Obtenir des réponses ou des explications détaillées.
- Résoudre une plainte ou un problème.
- Trouver un assistant de service client humain.
- Acheter un article de base ou payer une facture.
- Obtenir des idées et de l'inspiration pour les achats.

Cela donne une idée que les chatbots devraient être souvent utilisés dans la gestion d'entreprise en premier lieu, Il est certain que les chatbots sont utilisés dans différents autres domaines tels que: la banque, les télécommunications, l'assurance et la santé.

En général, un chatbot d'entreprise sert à:

- Support client et engagement en répondant à des questions répétitives et en collectant également les commentaires du client.
- Dans le cas d'utilisation du marketing et des ventes, les chatbots peuvent faire des recommandations personnalisées en fonction des intérêts de l'utilisateur, ils sont également utilisés pour gérer les achats et l'intégration des clients ou fournir aux clients un accès aux informations.
- Aider les employés à trouver rapidement des informations et servir de support de projet en envoyant des notifications et des rappels de diverses tâches.
- Faciliter les ventes et envoyer des offres en offrant des conseils et des suggestions aux clients.
- Offrir une assistance 24/7, ce qu'aucun humain ne peut faire.
- Menez des recherches sur les consommateurs en collectant des informations sur les requêtes répétitives, les tendances et préférences courantes concernant les produits, etc.

Chatbots, Entreprises et l'e-réputation

Se débarrasser des tâches de routine et traiter simultanément plusieurs demandes d'utilisateurs ne sont que quelques-unes des raisons. En outre, les applications Chatbot rationalisent les interactions entre les personnes et les services, améliorant ainsi l'expérience client. Dans le même temps, ils offrent aux entreprises de nouvelles opportunités pour améliorer le processus d'engagement des clients et l'efficacité opérationnelle en réduisant le coût typique du service client.

Ainsi, c'est parce que, pour la toute première fois, les gens utilisent davantage les applications de messagerie que les réseaux sociaux, selon une enquête[15]. Les gens passent maintenant plus de temps dans les applications de messagerie que dans les médias sociaux et c'est un énorme tournant. Les applications de messagerie sont les plates-formes du futur et les bots permettront à leurs utilisateurs d'accéder à toutes sortes de services. Donc, logiquement, si vous voulez créer une entreprise en ligne, vous voulez construire là où se trouvent les gens. Cet endroit est maintenant dans les applications de messagerie.

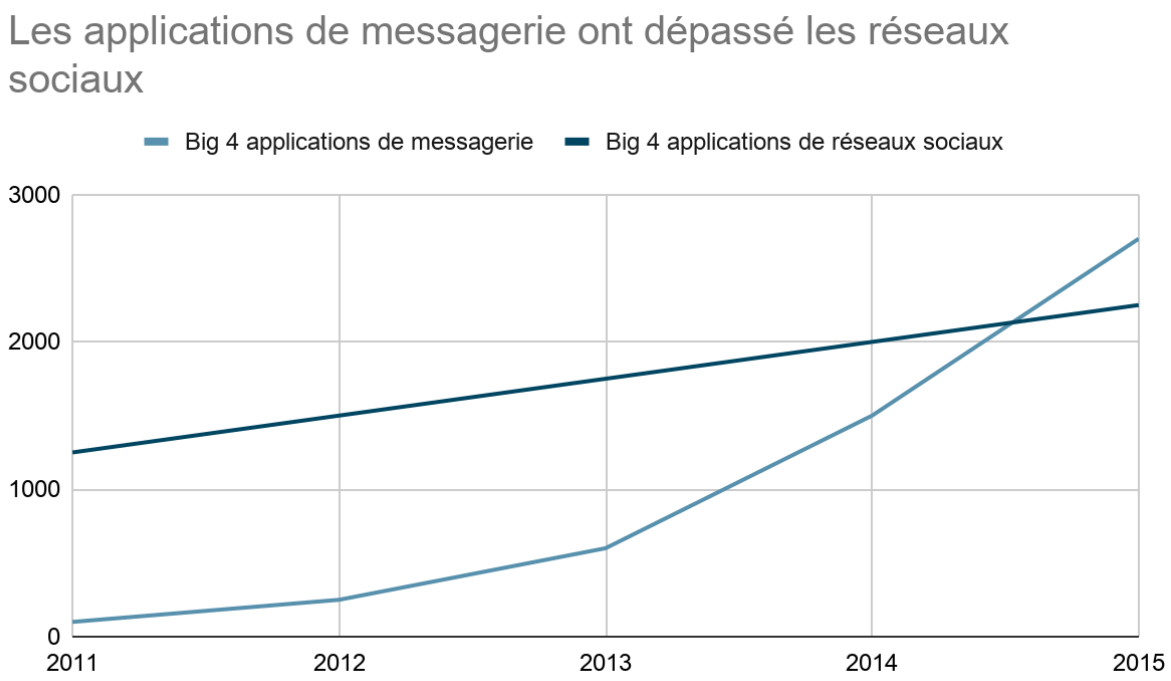


Figure 2. Applications de messagerie vs réseau social [15]

Vanson Bourne et Bold360 ont interrogé 850 chefs d'entreprise et 4500 consommateurs à travers le monde pour comprendre comment ces groupes perçoivent l'expérience client, déterminer où il y a encore des frictions et définir comment les technologies émergentes comme l'intelligence artificielle (IA) peuvent aider à améliorer les résultats.

Voici les principales conclusions du rapport :

- En moyenne, moins de la moitié (49 %) des demandes des clients sont résolues lors de la première interaction. Une raison possible à cela est que 68% des entreprises interrogées conviennent que leurs agents ont du mal avec le volume de demandes des clients.
- Un peu plus de six consommateurs sur dix (61 %) estiment qu'il faut trop de temps pour qu'une demande soit résolue.
- Plus de huit consommateurs sur dix (83 %) interrogés ont rencontré au moins un problème lors de l'interaction avec une marque, notamment devoir se répéter.
- Plus de 70 % des consommateurs conviennent que les marques devraient tirer parti des technologies pour réduire le temps nécessaire à la résolution d'une demande ou d'un problème client.

Conclusion

Dans ce chapitre, nous avons vu comment les chatbots font désormais partie intégrante de notre vie quotidienne. Beaucoup d'efforts sont déployés pour le faire parler comme un humain. De nos jours, le chatbot fait partie de presque toutes les applications qui traitent d'activités telles que la commande de vêtements, de nourriture, d'appareils électroniques, etc. Ils sont également utilisés pour réserver des billets, des rendez-vous, des spectacles ou toute activité transactionnelle. Les entreprises utilisent le chatbot pour résoudre le problème des clients en suggérant des questions fréquemment posées et en essayant de rendre la conversation interactive.

Dans le chapitre suivant, nous examinerons les dernières approches, les techniques réalisées concernant les chatbots et nous donnerons un bref aperçu des travaux récents.

ÉTAT DE L'ART DU CHATBOT

Introduction

Ces derniers temps, la conception et la mise en œuvre de chatbots ont reçu une grande attention de la part des développeurs et des chercheurs. L'objectif principal de ce chapitre est de résumer certaines des techniques de mise en œuvre les plus efficaces, approches qui ont été mises en œuvre les années précédentes.

Approches pour les chatbots intelligents

Il existe deux approches pour développer un chatbot en fonction des algorithmes et des techniques adoptées: Classique/Rule Based et l'apprentissage automatique.

Approches Classique/ Rule Based

Les approches classiques peuvent être qualifiées d'approches fondées sur des règles car elles définissent des règles prédéfinies pour générer des réponses. Ces règles sont devenues plus complexes et sophistiquées au fil du temps. Cela fonctionne très bien lorsque le domaine de la conversation est fermé, c'est-à-dire que la conversation est centrée sur un sujet/une tâche particulier.

Pattern Matching

Le Pattern Matching est l'une des techniques fondamentales de conception de chatbots et est utilisé dans presque tous les chatbot dans une certaine mesure. Cette méthode utilise un ensemble de règles pré-écrites et des modèles prédéfinis pour produire la réponse.

ELIZA a été le premier chatbot conçu en utilisant cette technique. Initialement, il identifie les mots-clés dans le texte en commençant de gauche à droite. Chaque mot-clé a une priorité qui lui est associée. Ensuite, la chaîne d'entrée est décomposée dans un modèle prédéfini [17].

Dans la sous-section suivante, nous examinerons l'un des langages les plus courants pour la mise en œuvre de chatbots avec l'approche de Pattern Matching.

Artificial Intelligence Markup Language (AIML)

Au cours des années 1995 à 2000, les développeurs ont créé le langage AIML (Artificial Intelligence Markup Language) [18], pour construire la base de connaissances des chatbots qui adoptent l'approche Pattern Matching. Il est basé sur XML et est open-source.

Un chatbot implémenté avec AIML est jusqu'à un certain point sensible au contexte. Il peut répondre aux entrées de l'utilisateur de différentes manières de manière aléatoire ou en fonction de la valeur des variables mises à jour au cours de la conversation.

La structure d'une balise AIML est:

<commande> Liste des paramètres </commande>

Où *<commande>* est la balise de début et *</commande>* est la balise de fermeture. Les balises les plus utilisées sont “category”, “pattern” et “template”. L'unité de base de connaissances ou communément appelée dialogue est définie par “category”. ”pattern” définit l'entrée probable de l'utilisateur et la réponse du chatbot est définie dans “template”.

< category >

< pattern > how are you? < /pattern >

< template >

I am absolutely fine!

< /template >

< /category >

AIML définit également des caractères génériques qui sont «_» et «*». Ils remplacent une chaîne ou une partie de la chaîne. AIML donne une priorité élevée aux catégories qui ont des caractères génériques en leur sein et elles sont analysées en premier.

< category >

*< pattern > I love * < /pattern >*

< template > I too love < star/ > . < /template >

< /category >

Discussion

Le principal inconvénient d'AIML est que l'auteur doit écrire un modèle pour chaque réponse possible de l'utilisateur. Cependant, cela aide le chatbot à répondre rapidement et facilement. AIML est facile à apprendre et à mettre en œuvre, mais les connaissances sont présentées comme une instance dans les fichiers AIML, par conséquent, il n'existe aucune disposition fiable pour la gestion des données à grande échelle. En outre, AIML est une série de mots de correspondance basée sur des règles qui produit soit une réponse entièrement contenue, soit une substitution de mots d'entrée et est très inefficace lors de l'adressage de grandes bases de connaissances [19]. Si des connaissances sont créées sur la base de données obtenues sur

Internet, elles doivent être mises à jour périodiquement, car les mises à jour automatiques ne peuvent pas être effectuées.

De plus, la version originale d'AIML n'a pas l'option d'extension. AIML a de mauvais modèles d'appariement et est difficile à gérer. Bien que le contenu soit simple à saisir, le principal défi est la grande quantité de données que le développeur doit saisir manuellement pour construire un chatbot fonctionnel [20].

Modèles de chaînes de Markov

Le modèle de chaîne de Markov, s'il est décrit mathématiquement, est le modèle qui décrit la probabilité d'événements présents sur la base de l'état des événements précédents. Il prend en considération la probabilité avec laquelle une lettre ou un mot apparaît dans un ensemble de données. Il utilise cette distribution de probabilité pour choisir les mots les plus probables pour une réponse. L'ordre de la chaîne de Markov détermine le nombre de lettres / mots successifs à prendre en entrée.
Lorsqu'un chatbot utilise cette méthode, il produira un élément de sortie pour accompagner la transition d'état. Cela lui permet de construire des phrases différentes à chaque fois mais plus ou moins cohérentes. L'état de départ peut être basé sur l'entrée de l'utilisateur, ce qui donne à la réponse une certaine pertinence [21].

Astuces linguistiques

Il est souvent plus naturel d'introduire dans un chatbot des concepts de type humain. Il peut s'agir de commettre délibérément une erreur d'orthographe ou de se faire passer pour lui-même [22]. Les astuces linguistiques sont souvent une technique supplémentaire utilisée dans le développement d'un chatbot. Certaines des astuces de langage courantes sont:

- **Erreurs de frappe et fausses frappe**s: Lorsqu'un utilisateur tape une entrée, il examine généralement le chatbot pendant qu'il saisit la réponse. Il ressemble beaucoup à un faux espace arrière et commet des fautes d'orthographe, qui sont des tendances naturelles des humains.
- **Réponses standardisées**: il existe certains modèles que le chatbot ne peut pas couvrir dans son algorithme de correspondance de modèles. Ces réponses sont “Hard coded” par le développeur.
- **Historique personnel**: les développeurs fournissent une identité au chatbot pour le rendre plus convaincant. Les détails sur sa naissance, son âge, ses parents, ses préférences, ses histoires y sont inculqués [23].
- **Obtenir plus d'informations**: si le chatbot n'a pas de réponse adéquate, il peut simplement demander à l'utilisateur de fournir plus d'informations. Il s'agit d'une question ouverte qui poursuit la conversation et donne au chatbot une autre chance d'interpréter la nouvelle entrée qu'il recevra.

Discussion

Comme nous l'avons mentionné précédemment, les chatbots basés sur des règles fonctionnent bien lorsque le domaine est fermé, mais à mesure que l'entrée devient plus naturelle ou que le domaine passe à l'ouverture, l'efficacité des approches basées sur des règles se détériore.

Les systèmes basés sur des règles, en général, ne créent pas de nouvelles réponses car les connaissances utilisées sont écrites par le développeur sous la forme de modèles conversationnels [24]. Plus la base de données avec les règles est étendue, plus un chatbot est capable de répondre aux questions de l'utilisateur. Comme il faut des milliers de règles pour que ce type de chatbot fonctionne correctement, il est difficile de gérer les erreurs grammaticales et syntaxiques dans les réponses de l'utilisateur. Cleverbot, Chatfuel et IBM Watson sont des chatbots basés sur des règles [24].

L'inconvénient de l'approche Pattern Matching est que les réponses sont automatisées, répétées et n'ont pas l'originalité et la spontanéité de la réponse humaine [24]. D'autre part, le temps de réponse est rapide, car un examen syntaxique ou sémantique plus approfondi du texte d'entrée n'est pas effectué [25].

Approches d'apprentissage automatique

Les chatbots qui adoptent des approches d'apprentissage automatique au lieu de la Pattern Matching extraient le contenu de l'entrée utilisateur à l'aide du traitement automatique du langage naturel (TALN) et disposent de la possibilité d'apprendre des conversations. Ils prennent en compte l'ensemble du contexte du dialogue, pas seulement le tour actuel, et ne nécessitent pas de réponse prédéfinie pour chaque entrée utilisateur possible. En règle générale, ils ont besoin d'un ensemble du "training set", dont la découverte peut constituer une difficulté cruciale car les ensembles de données disponibles peuvent être inadéquats. Par exemple, le corpus de scripts de films peut être trop large ou une ligne d'assistance informatique peut être trop spécifique [26].

Souvent, les Réseaux de Neurones Artificiels sont utilisés pour la mise en œuvre de ces chatbots. Les modèles Retrieval-based utilisent un réseau neuronal pour attribuer des scores et sélectionner la réponse la plus probable parmi un ensemble de réponses. En revanche, les modèles génératifs synthétisent la réponse, généralement en utilisant des techniques d'apprentissage en profondeur.

Réseaux de neurones artificiels

Inspiré par la fonctionnalité sophistiquée du cerveau humain où des centaines de milliards de neurones interconnectés traitent les informations en parallèle, un réseau de neurones artificiels (ou simplement un réseau de neurones) se compose d'une couche d'entrée de neurones (ou de nœuds, d'unités), d'une ou deux couches cachées de neurones et d'une couche finale de neurones de sortie [27]. La figure ci-dessous montre une architecture typique, où les

lignes reliant les neurones sont également représentées. Chaque connexion est associée à un nombre numérique appelé poids.

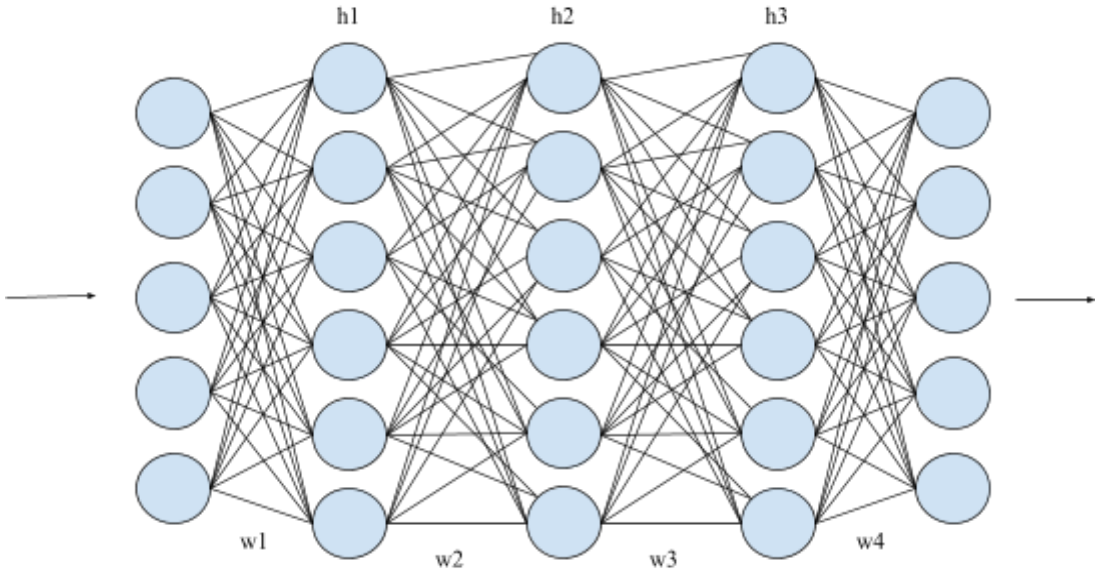


Figure 3. Un réseau neuronal typique.

Lorsqu'un réseau de neurones artificiels apprend, les poids entre les neurones changent, tout comme la force de la connexion Signification: Compte tenu des données d'entraînement et d'une tâche particulière telle que la classification des nombres, nous recherchons certains poids fixes qui permettent au réseau de neurones de fonctionner la classification [28]. L'ensemble de poids est différent pour chaque tâche et chaque ensemble de données. Nous ne pouvons pas prédire les valeurs de ces poids à l'avance, mais le réseau neuronal doit les apprendre. Le processus d'apprentissage que nous appelons également formation.

Apprentissage profondeur

L'apprentissage en profondeur (Deep Learning en Anglais) est une technique d'apprentissage automatique qui enseigne aux ordinateurs à faire ce qui vient naturellement aux humains: apprendre par l'exemple. Dans l'apprentissage en profondeur, un modèle informatique apprend à effectuer des tâches de classification directement à partir d'images, de texte ou de son. Les modèles d'apprentissage profond peuvent atteindre une précision de pointe, dépassant parfois les performances au niveau humain [29].
La plupart des méthodes du Deep Learning utilisent des architectures de réseau de neurones, c'est pourquoi les modèles d'apprentissage en profondeur sont souvent appelés réseaux de neurones “profonds”.
Le terme profond fait généralement référence au nombre de couches cachées dans le réseau neuronal. Les réseaux de neurones traditionnels ne contiennent que 2 à 3 couches cachées, tandis que les réseaux profonds peuvent en avoir jusqu'à 150.

Les modèles d'apprentissage en profondeur ont tendance à augmenter leur précision avec la quantité croissante de données d'entraînement, là où les modèles d'apprentissage automatique traditionnels tels que SVM et le classificateur Naive Bayes cessent de s'améliorer après un point de saturation [30].

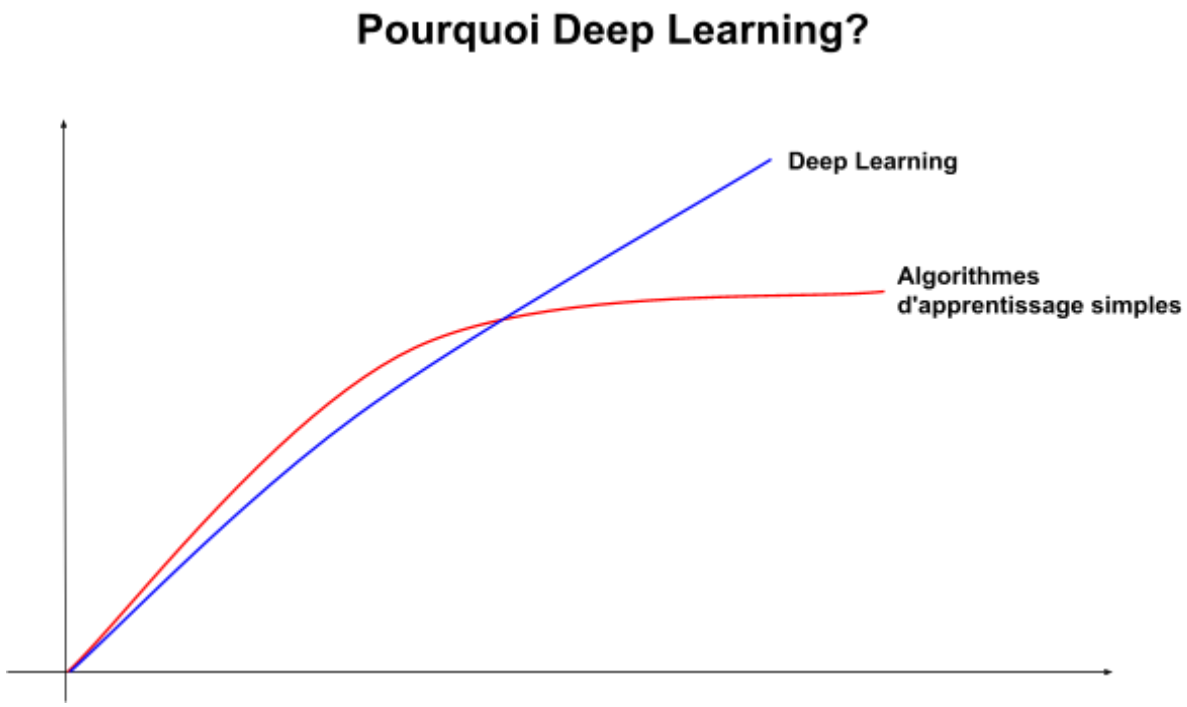


Figure 4. Pourquoi l'apprentissage en profondeur [31]

Bien que les réseaux de neurones classiques produisent des résultats acceptables, ils ne satisfont toujours pas le mot «profond». L'un des types de réseaux de neurones profonds les plus populaires et les plus couramment utilisés est les Réseaux de Neurones Récurrents (Recurrent Neural Networks).

Réseaux de Neurones Récurrents

Les humains ne commencent pas leur réflexion à partir de zéro à chaque seconde. En lisant, vous comprenez chaque mot en fonction de votre compréhension des mots précédents. Vous ne jetez pas tout et recommencez à penser à partir de zéro. Vos pensées sont persistantes.
Les réseaux de neurones traditionnels ne peuvent pas faire cela, et cela semble être une lacune majeure. Par exemple, imaginez que vous souhaitez classer le type d'événement qui se produit à chaque étape d'un film. On ne sait pas comment un réseau de neurones traditionnel pourrait utiliser son raisonnement sur les événements précédents du film pour informer les événements ultérieurs.
Les réseaux de neurones récurrents résolvent ce problème. Ce sont des réseaux avec des boucles qui permettent aux informations de persister. Ces boucles peuvent être considérées comme plusieurs copies du même réseau, chacune transmettant un message à un successeur.

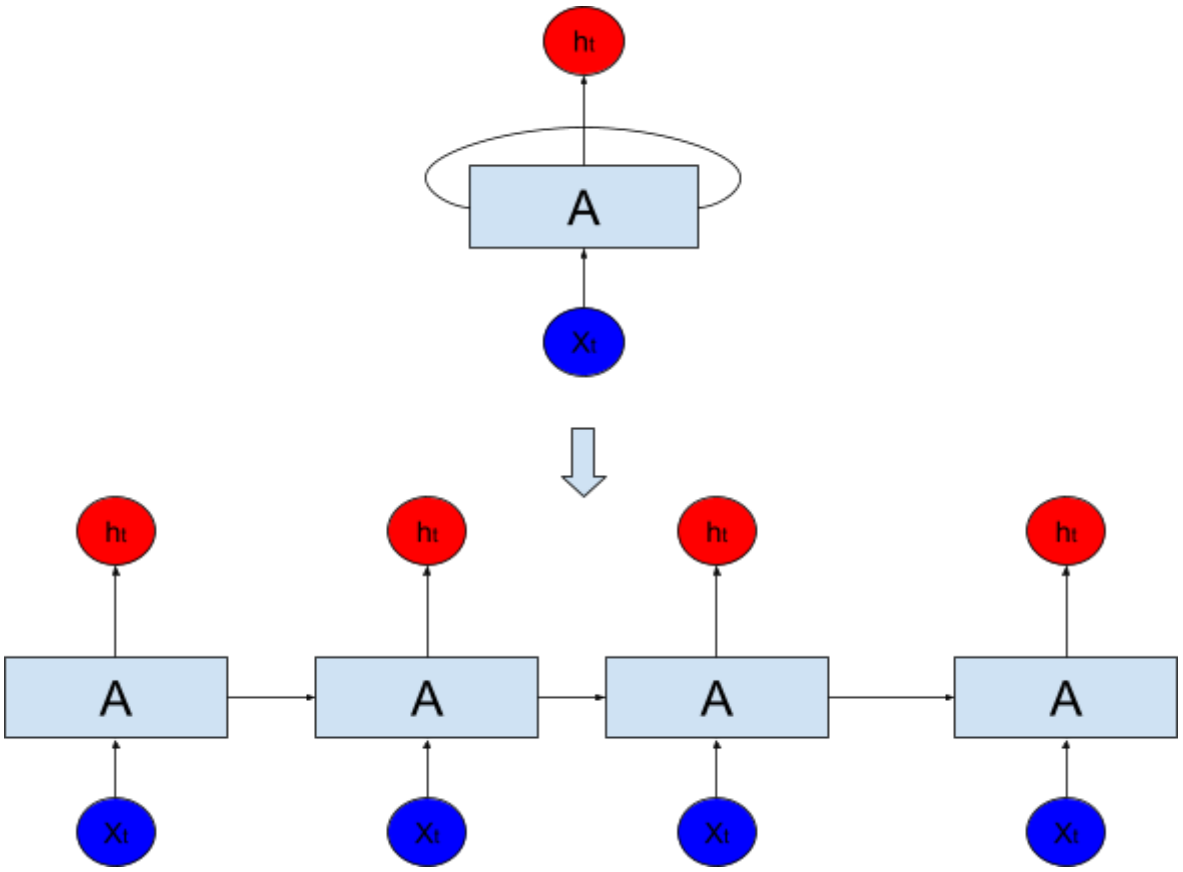


Figure 5. Réseau neuronal récurrent

Cette nature en chaîne révèle que les réseaux de neurones récurrents sont intimement liés aux séquences et aux listes. Ils constituent l'architecture naturelle du réseau neuronal à utiliser pour de telles données [32].

Les développeurs de chatbots utilisent les réseaux de neurones récurrents pour prendre en compte le contexte précédent dans une conversation [33].
Les RNC ont été largement utilisés à des fins diverses telles que la traduction de la langue, la modélisation de la reconnaissance vocale, le sous-titrage d'images, etc. Cependant, la version non modifiée de RNC n'est pas beaucoup utilisée car elle souffre d'un problème de gradient de disparition [34]. De plus, lorsque les développeurs de chatbots ont besoin de se référer aux informations précédentes et d'apprendre les dépendances à long terme, ils utilisent des réseaux de Long Short Term Memory (LSTM), un type particulier de RNC [35].

Long Short Term Memory

Il devient difficile pour un simple RNC de se souvenir des informations vues il y a plusieurs étapes lorsque les étapes de déroulement augmentent trop. En effet, la valeur du gradient dépend principalement de deux facteurs qui sont le poids et la fonction d'activation (essentiellement leur dérivée). Lorsque l'un d'eux s'approche de 0, le gradient disparaît avec le temps. Les fonctions d'activation telles que tanh et sigmoïde aggravent encore la situation

car leurs valeurs dérivées sont pour la plupart proches de 0. C'est là que Long Short Term Memory (LSTM) entre en scène et résout le problème du disparition du gradient.

LSTM utilise l'identité comme fonction d'activation dont le dérivé est 1, ce qui empêche le gradient rétropropagé de disparaître. LSTM accomplit cette tâche à l'aide de «portes». Les portes sont le composant qui décide des informations qui seront autorisées à passer. Les portes affichent la valeur entre 0 et 1, décidant de la quantité de chaque composant à laisser passer.

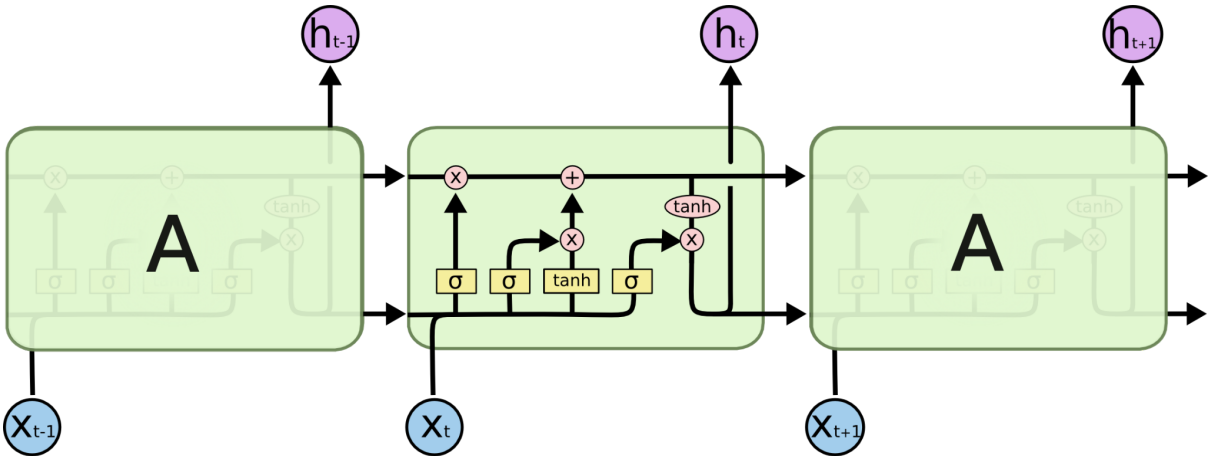


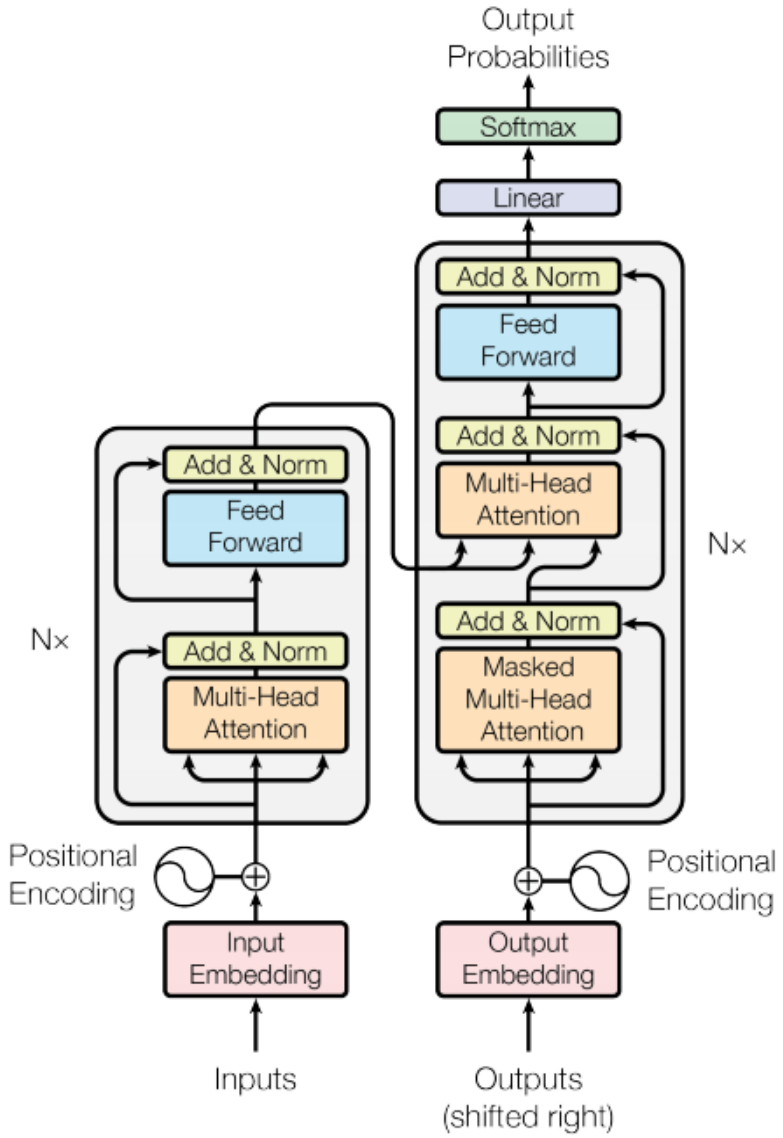
Figure 6. Architecture LSTM [32]

1. LSTM décide des informations que nous allons jeter de l'état de la cellule. Cette décision est prise par une couche sigmoïde appelée «couche de porte oubliée».
2. LSTM décide des nouvelles informations que nous allons stocker dans l'état de la cellule. Cela comporte deux parties. Tout d'abord, une couche sigmoïde appelée "couche de porte d'entrée" décide des valeurs que nous mettrons à jour.
3. Une couche *tanh* crée un vecteur de nouvelles valeurs candidates, qui pourraient être ajoutées à l'état.
4. L'ancien état de cellule sera mis à jour dans le nouvel état de cellule. Les étapes précédentes ont déjà décidé de la marche à suivre.

Discussion

Les réseaux neuronaux récurrents normaux sont lents, mais les LSTM sont encore plus lents car ils sont plus complexes, pour ces réseaux, les données d'entrée doivent être transmises séquentiellement ou en série l'une après l'autre, nous avons besoin des entrées de l'état précédent pour faire tout opérations sur l'état actuel, un tel flux séquentiel n'utilise pas très bien les GPU d'aujourd'hui qui sont conçus pour le calcul parallèle [1_T], la question est donc: Comment peut-on utiliser la parallélisation pour les données séquentielles ?

Transformer Neural Networks

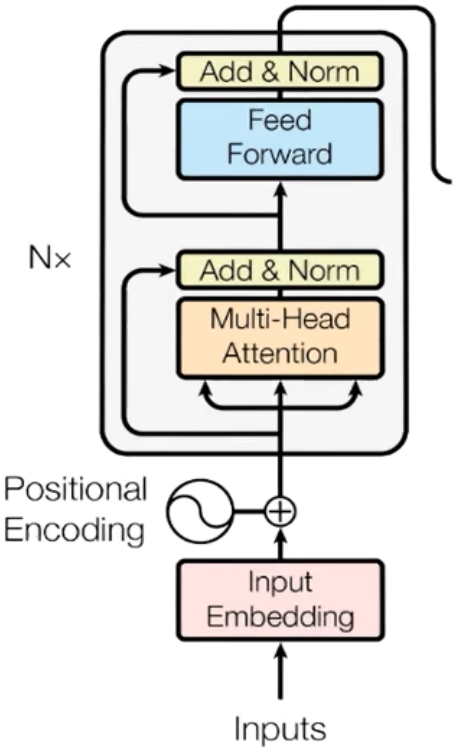


En 2017, l'architecture Transformer Neural Networks a été introduite [2_T], le réseau utilise une architecture encodeur-décodeur un peu comme les RNNs, la différence est que les données séquentielles peuvent être transmises en parallèle. Étant donné que l'utilisation principale des TNN est la traduction automatique, nous utiliserons un exemple de traduction d'une phrase anglaise en français pour expliquer leur fonctionnement.

Avec un encodeur RNN, nous passons en entrée un mot après l'autre, les mots actuels état caché ont des dépendances dans les mots précédents état caché, les plongements de mots sont générés un pas de temps à un temps mais avec un encodeur Transformer il n'y a pas de notion

de pas de temps pour l'entrée [3], on passe tous les mots de la phrase simultanément et on détermine le mot enfouissant simultanément. Alors, comment les TNN y parviennent-ils ?

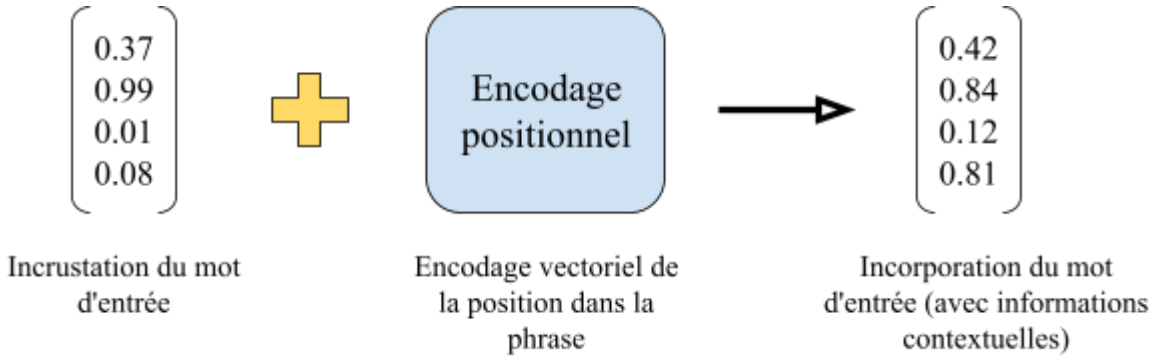
Encodeur



- **Input Embedding**

Les ordinateurs ne comprennent pas les mots, ils obtiennent des nombres, des vecteurs et des matrices, l'idée est de mapper chaque mot à un point de l'espace où des mots similaires de sens sont proches les uns des autres, l'espace dans lequel ils sont présents s'appelle un " Espace d'intégration", nous pourrions préformer cet espace d'intégration pour gagner du temps, ou même simplement utiliser un espace d'intégration déjà pré-formé, cet espace d'intégration mappe un mot à un vecteur, mais le même mot dans différentes phrases peut avoir des significations différentes.

C'est là qu'interviennent les encodeurs positionnels, c'est un vecteur qui contient des informations sur les distances entre les mots et les phrases [4], l'article original utilise une fonction sinus cosinus pour générer ce vecteur mais cela pourrait être n'importe quelle fonction raisonnable. Après avoir passé une phrase à travers l'intégration d'entrée et appliqué l'encodage positionnel, nous obtenons des vecteurs de mots qui ont des informations de position qui sont le contexte.



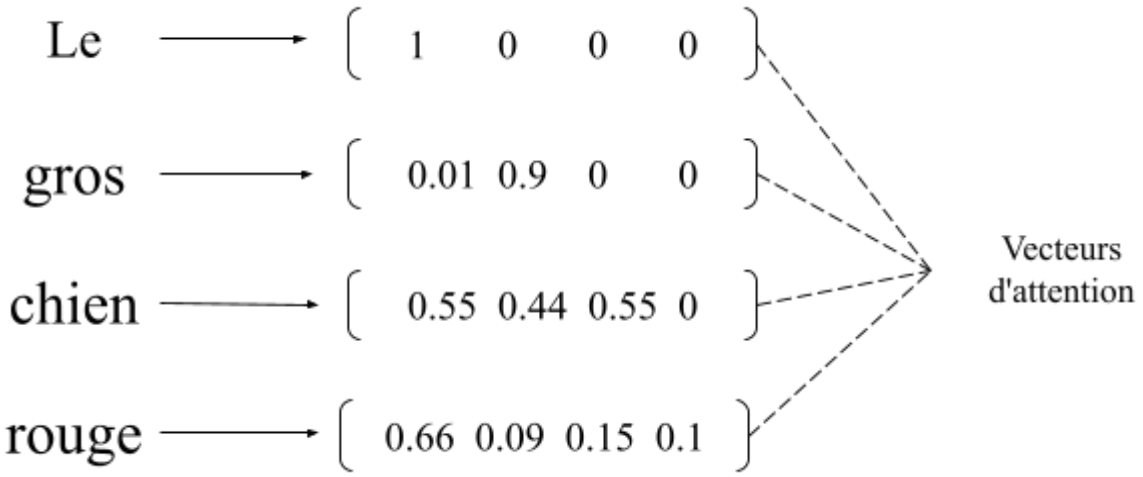
$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

Nous transmettons ensuite les informations de contexte au bloc d'encodeur où elles passent par une couche d'attention Multi-Headed et une couche Feed Forward.

- **Couche d'attention**

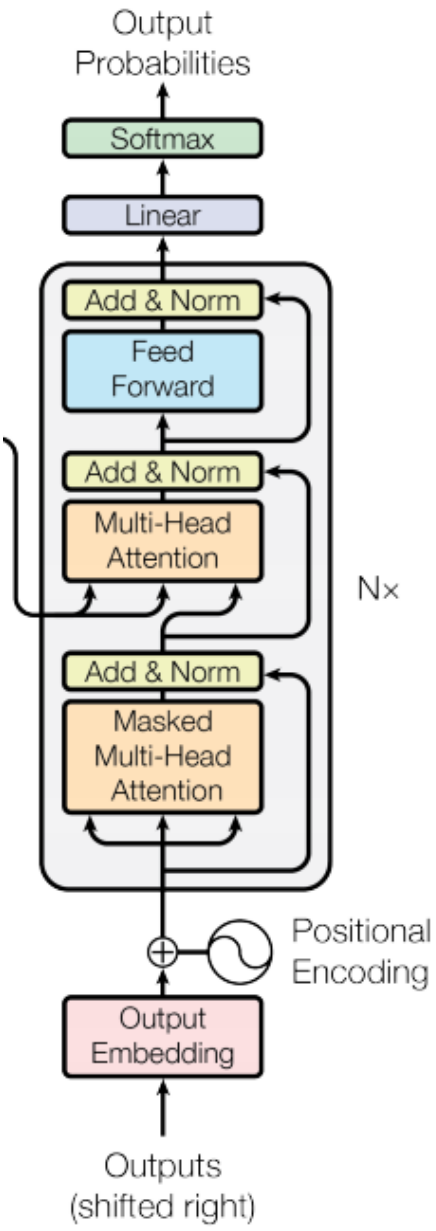
L'attention implique de répondre sur quelle partie de l'entrée dois-je me concentre [3_T], si nous traduisons de l'anglais vers le français et que nous faisons de l'auto-attention (c'est-à-dire de l'attention par rapport à soi-même), la question à laquelle nous voulons répondre est de savoir dans quelle *i* ème mot de la phrase anglaise aux autres mots de la même phrase, ceci est représenté dans le *i* ème vecteur d'attention et il est calculé dans le bloc d'attention, pour chaque mot, nous pouvons avoir un vecteur d'attention généré qui capture les relations contextuelles entre mots dans la phrase.



- **Couche Feed Forward**

Un simple réseau Feed Forward appliqué à tous les vecteurs d'attention, ces réseaux sont utilisés en pratique pour transformer les vecteurs d'attention en une forme digestible par le prochain bloc codeur ou décodeur.

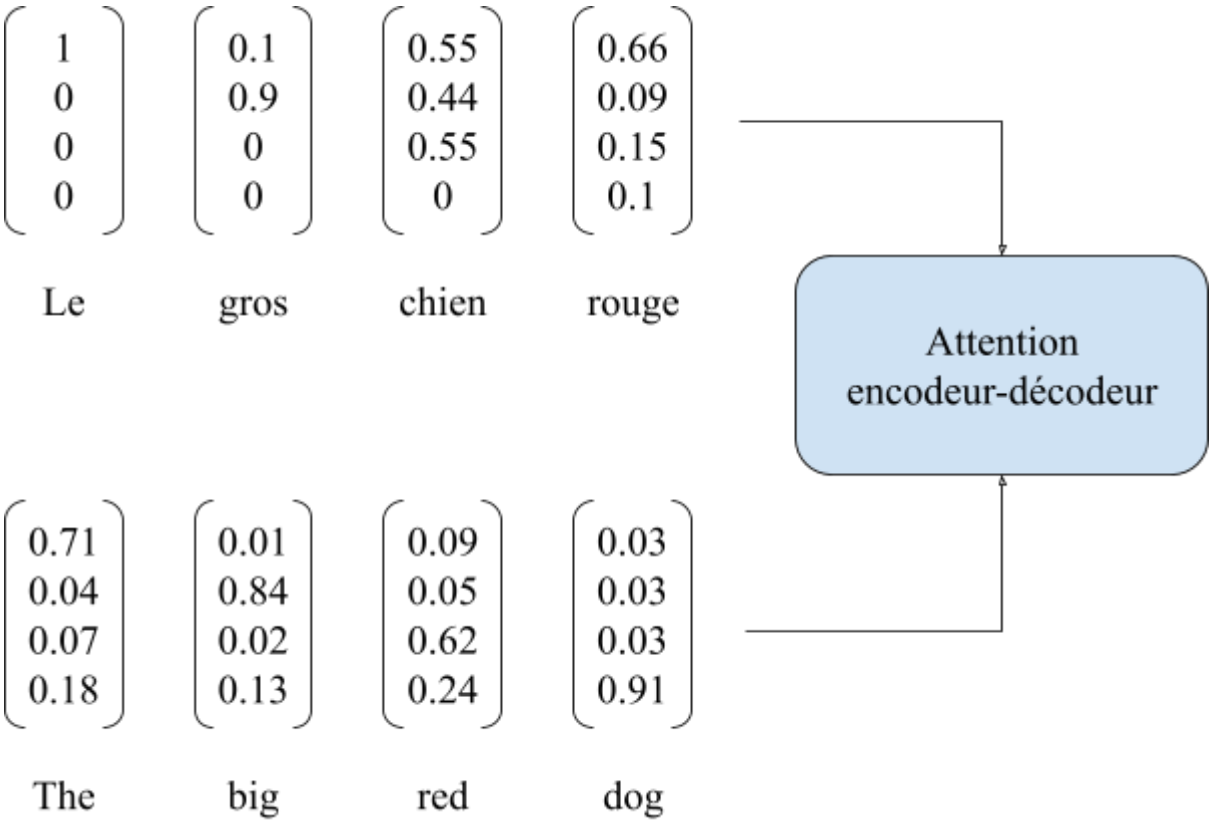
Décodeur



Pendant la phase d'apprentissage pour la traduction de l'anglais vers le français, nous alimentons la phrase française de sortie au décodeur et nous la traitons en utilisant des plongements de mots pour obtenir la forme vectorielle du mot puis nous ajoutons un vecteur positionnel pour obtenir la notion de contexte du mot dans la phrase, nous passons enfin ce vecteur dans le bloc décodeur qui a trois composants principaux (dont deux sont similaires au bloc codeur).

Le bloc d'auto-attention génère des vecteurs d'attention pour chaque mot de la phrase afin de représenter à quel point chaque mot est lié à chaque mot de la même phrase, ces vecteurs d'attention et les vecteurs de l'encodeur sont transmis à un autre bloc d'attention appelé Le bloc Encodeur-Décodeur [1_T].

Le bloc encodeur-décodeur déterminera la relation de chaque vecteur de mot les uns par rapport aux autres et c'est là que se produit le mappage principal des mots anglais vers français, la sortie de ce bloc est des vecteurs d'attention pour chaque mot en anglais et en français, chaque vecteur représentant les relations avec les autres mots dans les deux langues.



Ensuite, nous passons chaque vecteur d'attention à une unité Feedforward, cela rend le vecteur de sortie plus digeste par le bloc décodeur suivant ou une couche linéaire, la couche linéaire n'est rien d'autre qu'une autre couche connectée Feed-Forward utilisée pour étendre les dimensions dans le nombre de mots dans le langage de sortie, puis la couche softmax le transforme en une distribution de probabilité interprétable par l'homme et le mot final est le mot correspondant à la probabilité la plus élevée [3].

Dans l'ensemble, ce décodeur prédit le mot suivant et nous exécutons sur plusieurs pas de temps jusqu'à ce que le jeton de fin de phrase soit généré.

Ce qui rend cela conceptuellement beaucoup plus attrayant que certaines cellules LSTM, c'est que nous pouvons physiquement voir une séparation dans les tâches, l'encodeur apprend ce qu'est la langue, la grammaire et surtout quel est le contexte, le décodeur apprend comment

les mots se rapportent les uns aux autres, ces deux, même séparément, ont une certaine compréhension sous-jacente du langage.

BERT

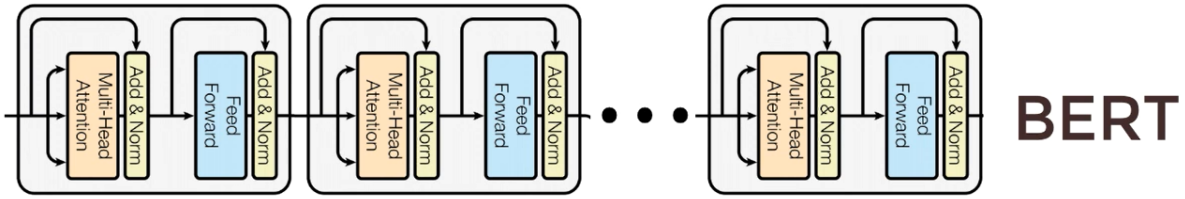
Ce qui rend l'architecture Transformer Neural Networks tellement plus attrayante sur le plan conceptuel que certaines cellules LSTM, c'est que nous pouvons physiquement voir une séparation des tâches, l'encodeur apprend ce qu'est la langue, la grammaire et surtout le contexte, le décodeur apprend comment les mots se rapportent à les uns des autres, même séparément, ont une compréhension sous-jacente du langage et, grâce à cette compréhension, nous pouvons choisir une partie de cette architecture et construire des systèmes qui comprennent le langage, par exemple, si nous empilons uniquement des encodeurs, nous obtenons BERT (Bidirectional Encoder Representations From Transformers).

Définition

Bidirectional Encoder Representations From Transformers (BERT) est une technique d'apprentissage automatique Transformer-based pour la pre-training au Traitement Automatique du Langage Naturel (TALN) développée par Google [1_B], BERT a été créée et publiée en 2018 par Jacob Devlin et ses collègues de Google.

Nous pouvons utiliser BERT pour apprendre la traduction linguistique, Question Answering, l'analyse des sentiments, Text Classification et bien d'autres tâches [2_B]. Il s'avère que tous ces problèmes nécessitent la compréhension du langage, nous pouvons donc former BERT à comprendre le langage, puis affiner en fonction du problème que nous voulons résoudre, en tant que tel, l'apprentissage de BERT se fait en deux phases :

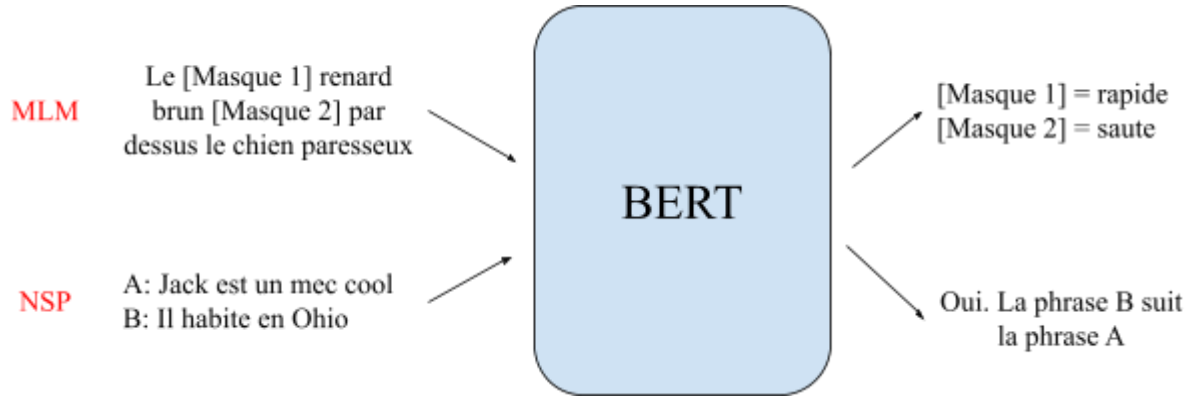
- **Pre-training** où le modèle comprend ce qu'est la langue et ce qu'est le contexte
- **Fine-tuning** où le modèle apprend que je connais la langue, mais comment puis-je résoudre ce problème



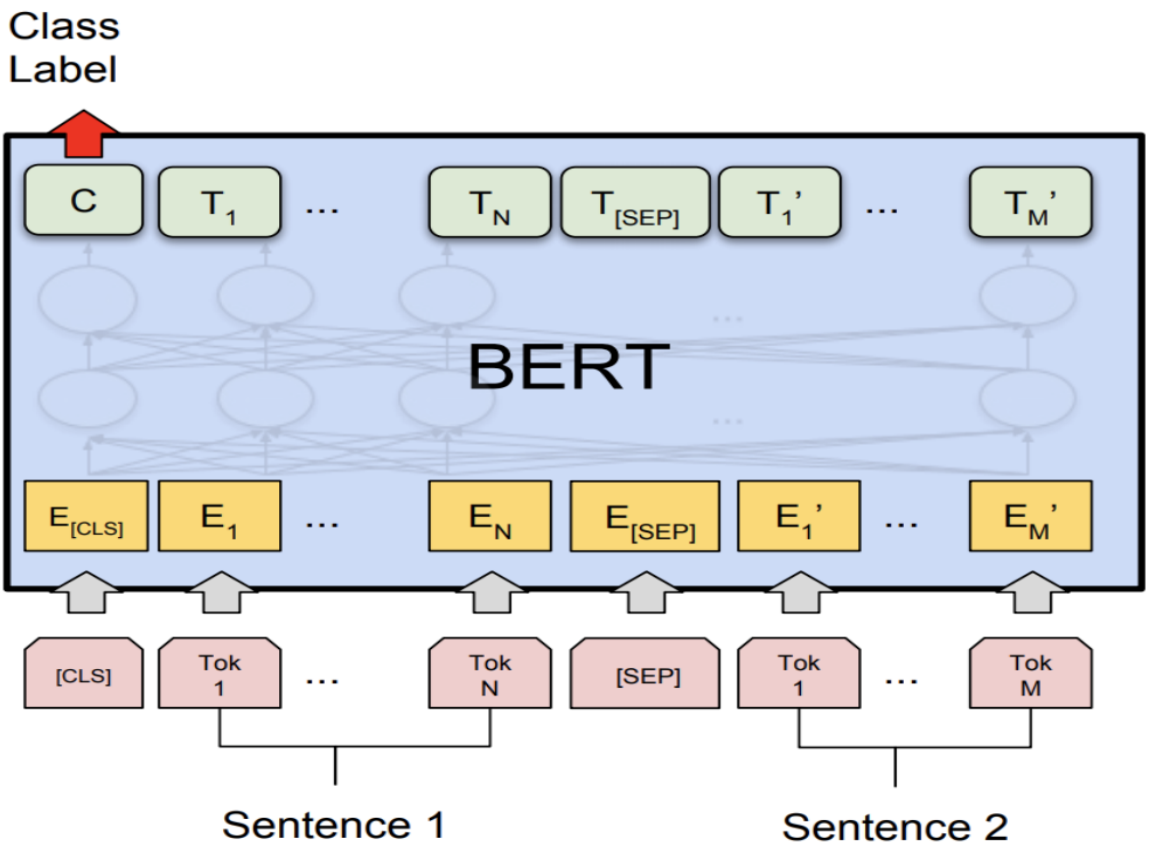
Pre-training

L'objectif de Pre-training est de faire comprendre au BERT ce qu'est le langage et ce qu'est le contexte. BERT apprend le langage en s'entraînant sur deux tâches non supervisées simultanément, ce sont Mass Language Modeling (MLM) et Next Sentence Prediction (NSP) [3_B].

Pour MLM, BERT prend une phrase avec des mots aléatoires remplis de masques, le but est de sortir ces jetons de masque (un peu comme remplir les blancs), cela aide BERT à comprendre le contexte bidirectionnel dans une phrase. Dans le cas de NSP, BERT prend deux phrases et détermine si la deuxième phrase suit réellement la première (une sorte de problème de classification binaire), cela aide BERT à comprendre le contexte à travers différentes phrases elles-mêmes et à utiliser les deux ensemble, BERT obtient une bonne compréhension de la langue.

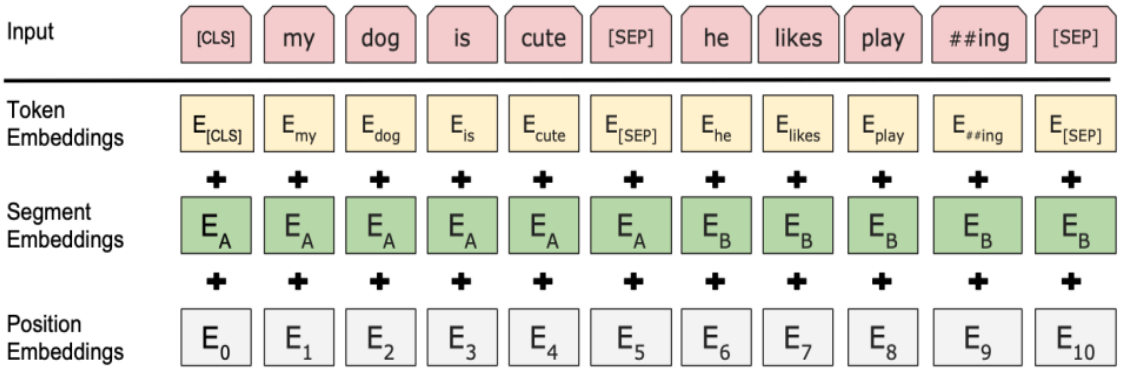


En pratique ces deux problèmes sont appris simultanément, l'entrée est un ensemble de deux phrases avec certains des mots masqués, chaque jeton est un mot et nous convertissons chacun de ces mots dans des embeddings, ceci fournit à BERT un bon point de départ pour commencer, du côté de sortie *C* (voir la figure ci-dessous) est la sortie binaire pour NSP, donc il produirait 1 si la phrase B suit la phrase A et 0 si la phrase B ne suit pas la phrase À, chacun des *T* ici sont des vecteurs de mots qui correspondent à la sortie du problème MLM, donc le nombre de vecteurs de mots que nous saisissons est le même que le nombre de vecteurs de mots que nous sortons.



Du côté des entrées, comment allons-nous générer ces intégrations à partir des entrées de jetons de mots ? Et bien l'embedding initial est construit à partir de trois vecteurs :

- Token Embeddings (le papier principal utilise des embeddings de wordPieces qui ont un vocabulaire de 30.000 tokens)
- Segment Embeddings (le numéro de phrase qui est encodé dans un vecteur)
- Position Embeddings (la position d'un mot dans cette phrase qui est codé dans un vecteur)



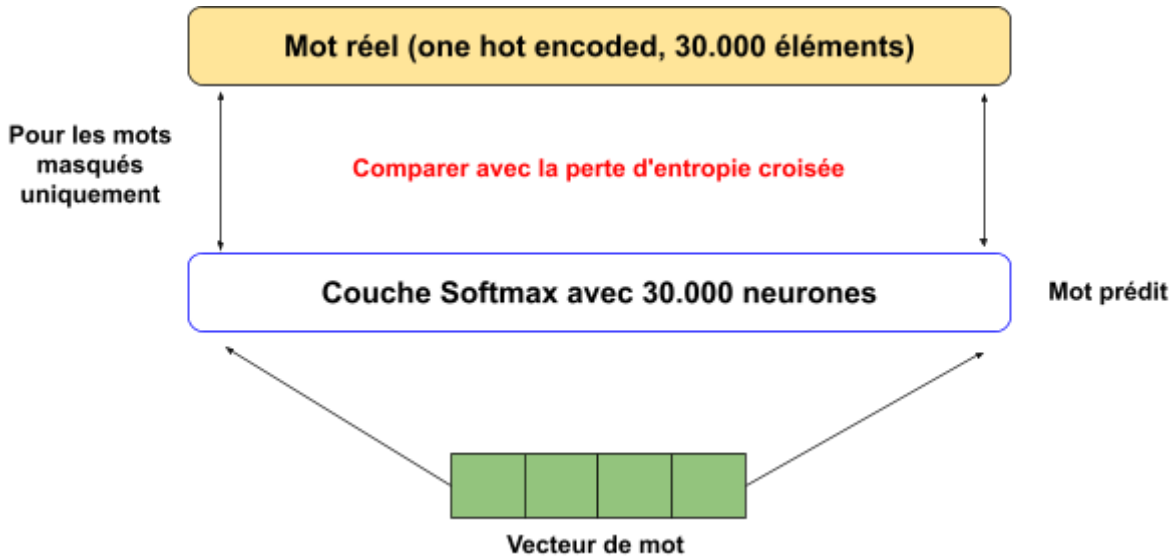
En ajoutant ces trois vecteurs, nous obtenons un vecteur d'intégration que nous utilisons en entrée de BERT. Les intégrations de segment et de position sont nécessaires pour l'ordre temporel puisque tous ces vecteurs sont introduits simultanément dans BERT et les modèles de langage ont besoin que cet ordre soit préservé [4_B].

Du côté de la sortie, l'output est une valeur binaire *C* et un tas de vecteurs de mots, mais avec l'apprentissage, nous devons minimiser une perte, il y a deux détails clés à noter ici :

- Tous ces vecteurs de mots ont la même taille.
- Tous ces vecteurs de mots sont générés simultanément

Nous devons prendre chaque mot vecteur, le passer dans une sortie en couches entièrement connectée avec le même nombre de neurones égal au nombre de jetons dans le vocabulaire (ce serait une couche de sortie correspondant à 30 000 neurones dans le cas du papier principal), et nous appliquerions une fonction d'activation softmax, de cette façon, nous convertirons un vecteur de mot en une distribution et l'étiquette réelle de cette distribution serait un vecteur One Hot Encoded pour le mot réel, et nous comparons donc ces deux distributions, puis nous nous entraînons le réseau en utilisant la perte d'entropie croisée.

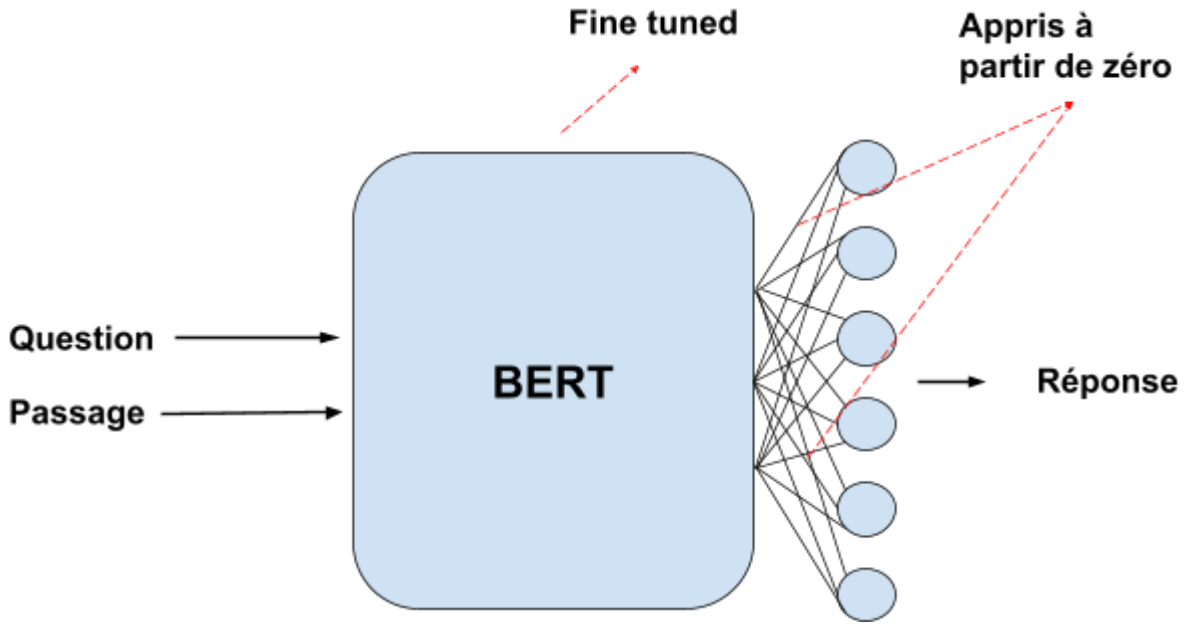
Notez que la sortie a tous les mots, même si ces mots n'étaient pas du tout masqués, la perte ne considère que la prédiction des mots masqués et elle ignore tous les autres mots qui sont sortis par le réseau, ceci est fait pour s'assurer que plus l'accent est mis sur la prédiction des valeurs de masse afin qu'elles soient correctes et que cela augmente la connaissance du contexte.



Fine-tuning

Nous pouvons maintenant former davantage BERT sur des tâches TALN très spécifiques, par exemple, prenons Question Answering, tout ce que nous avons à faire est de remplacer des couches de sortie entièrement connectées du réseau par un nouvel ensemble de couches de sortie qui peuvent essentiellement générer la réponse au question que nous voulons, alors

nous pouvons effectuer une apprentissage supervisée à l'aide d'une question répondant à l'ensemble de données [2_B], cela ne prendra pas longtemps car ce ne sont que les paramètres de sortie qui sont appris à partir de zéro, le reste des paramètres du modèle est juste légèrement affiné et par conséquent le temps d'entraînement est rapide et nous pouvons le faire pour tout problème de TALN (c'est-à-dire remplacer les couches de sortie, puis appris avec un ensemble de données spécifique.



On entraînerait le modèle en modifiant juste les entrées et la couche de sortie, on passe la question suivie d'un passage contenant la réponse en entrées et dans le couche de sortie, nous afficherons les mots de début et de fin qui encapsulent la réponse en supposant que la réponse se trouve dans la même étendue de texte.

Bien sûr, les performances dépendent de la taille que nous voulons que BERT soit, le grand modèle BERT qui a 340 millions de paramètres peut atteindre des précisions bien plus élevées que le modèle de base BERT qui n'a que 110 millions de paramètres.

Traitement Automatique de Langage Naturel

Le traitement automatique du langage naturel (Natural Language Processing en Anglais), généralement abrégé en TALN (NLP en Anglais), fait référence aux systèmes informatiques qui analysent, tentent de comprendre ou produisent un ou plusieurs langages humains. L'entrée peut être du texte, une langue parlée ou une saisie au clavier. La tâche peut être de traduire dans une autre langue, de comprendre et de représenter le contenu du texte, de créer une base de données ou de générer des résumés, ou de maintenir un dialogue avec un utilisateur dans le cadre d'une interface de base de données ou de recherche d'informations [37].

Le TALN a récemment attiré beaucoup d'attention de la communauté scientifique, selon Google Research, le nombre d'articles scientifiques publiés liés à Le TALN est passé de moins de 50 articles par an en 2005 à plus de 500 fin de 2015.

Le graphe ci-dessous démontre la croissance régulière du nombre d'articles jusqu'en 2020

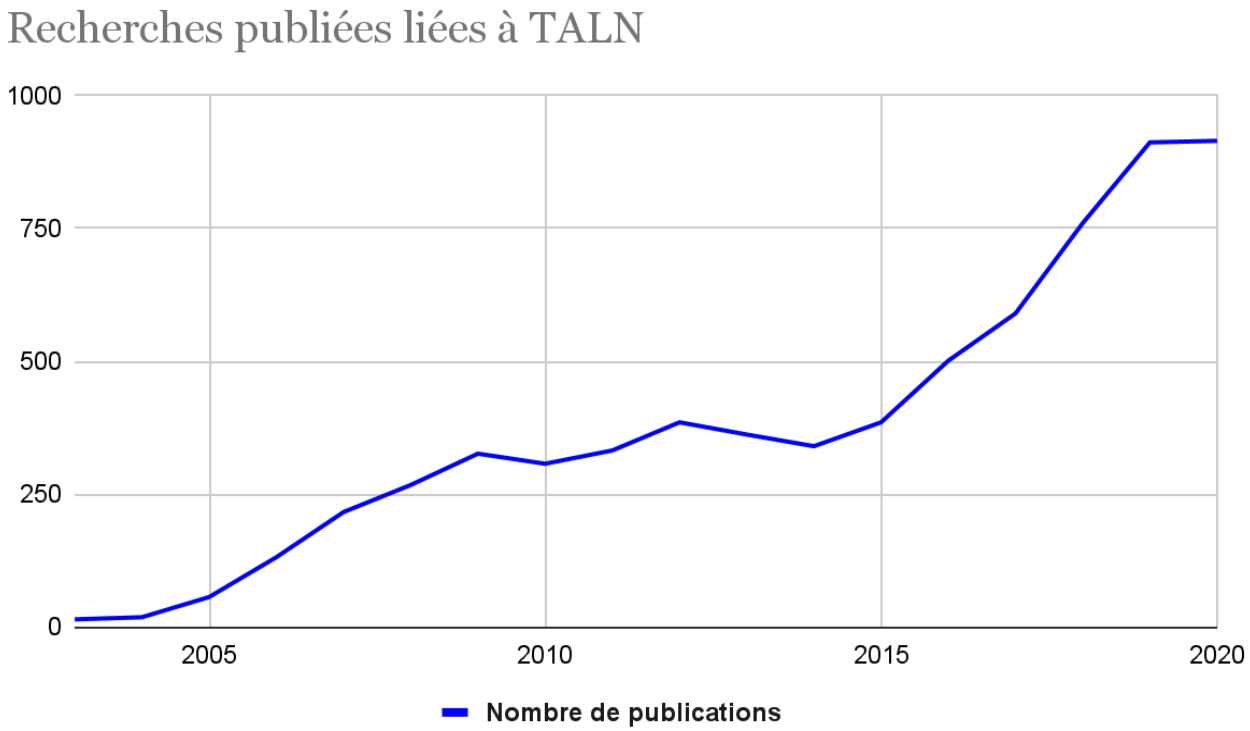


Figure 8. Recherches publiées liées à TALN par an [Google Research]

TALN implique l'application d'algorithmes pour identifier et extraire les règles du langage naturel de sorte que les données de langage non structurées soient converties en une forme que les ordinateurs peuvent comprendre.
Une fois le texte fourni, l'ordinateur utilisera des algorithmes pour extraire le sens associé à chaque phrase et en collecter les données essentielles.
Pour ce faire, TALN utilise certaines techniques telles que l'analyse syntaxique et l'analyse sémantique qui sont les principales techniques utilisées pour réaliser les tâches de traitement du langage naturel [38].
Voici une description de leur utilisation:

1. Syntaxe

La syntaxe fait référence à la disposition des mots dans une phrase de telle sorte qu'ils aient un sens grammatical, dans TALN, l'analyse syntaxique est utilisée pour évaluer comment la langue naturelle s'aligne avec les règles grammaticales [38].

Certaines techniques de syntaxe incluent:

- **Lemmatisation**: elle consiste à réduire les différentes formes fléchies d'un mot en une seule forme pour une analyse facile.
- **Segmentation morphologique**: elle consiste à diviser les mots en unités individuelles appelées morphèmes.
- **POS Tagging**: Il s'agit d'identifier la partie du discours pour chaque mot.
- **Analyse syntaxique**: elle consiste à entreprendre une analyse grammaticale de la phrase fournie.
- **Stemming**: Il s'agit de couper les mots fléchis à leur forme racine.

2. Sémantique

La sémantique fait référence au sens véhiculé par un texte. L'analyse sémantique est l'un des aspects difficiles du traitement du langage naturel qui n'a pas encore été entièrement résolu, elle implique l'application d'algorithmes informatiques pour comprendre le sens et l'interprétation des mots et comment les phrases sont structurées [38].

Certaines techniques d'analyse sémantique incluent:

- **Reconnaissance d'entités nommées (REN)**: Il s'agit de déterminer les parties d'un texte qui peuvent être identifiées et classées en groupes prédéfinis. Des exemples de tels groupes comprennent les noms de personnes et les noms de lieux.
- **Désambiguïsation du sens des mots**: il s'agit de donner un sens à un mot en fonction du contexte.
- **Génération de langage naturel**: il s'agit d'utiliser des bases de données pour dériver des intentions sémantiques et les convertir en langage humain.

La Compréhension du Langage Naturel

La compréhension du langage naturel (Natural Language Understanding) est définie comme «la compréhension des ordinateurs de la structure et de la signification du langage humain, permettant aux utilisateurs d'interagir avec l'ordinateur à l'aide de phrases naturelles» [39]. En d'autres termes, CLN est une intelligence artificielle qui utilise un logiciel informatique pour interpréter du texte et tout type de données non structurées. CLN peut digérer un texte, le traduire en langage informatique et produire une sortie dans une langue que les humains peuvent comprendre.

CLN aide le chatbot à comprendre la requête en la décomposant. Il comporte trois concepts spécifiques:

1. **Entités**: une entité représente les mots-clés de la requête de l'utilisateur sélectionnés par le chatbot pour comprendre ce que l'utilisateur souhaite. C'est un concept dans

votre chatbot. Par exemple, «Quelle est ma facture impayée?» A le mot «facture» en tant qu'entité.

2. **Intentions**: cela aide à identifier l'action que le chatbot doit effectuer sur l'entrée de l'utilisateur. Par exemple, l'intention de «Je veux commander un t-shirt» et «Avez-vous un t-shirt? Je veux en commander un »et« Montrez-moi des t-shirts »est la même chose. Tous ces textes utilisateur déclenchent une seule commande donnant aux utilisateurs des options pour les t-shirts.
3. **Contexte**: il n'est pas facile d'évaluer le contexte du dialogue pour un algorithme CLN car il ne dispose pas de l'historique des conversations de l'utilisateur. Cela signifie qu'il ne se souviendra pas de la question s'il reçoit la réponse à une question qu'il vient de poser. Pour différencier les phases de la conversation par chat, son état doit être stocké. Il peut signaler des expressions telles que «Commander une pizza» ou des paramètres tels que «Restaurant:« Dominos »». Avec le contexte, vous pouvez facilement relier les intentions sans avoir besoin de savoir quelle était la question précédente.

Les chatbots utilisent la compréhension du langage naturel pour récupérer le contexte de l'entrée utilisateur non structurée en langage humain et répondre en fonction de l'intention de l'utilisateur actuel [40]. Les trois problèmes majeurs soulevés au cours du processus CLN sont les mécanismes de la pensée, l'interprétation et les connaissances générales de l'utilisateur [41].

Mesures d'évaluation

Les métriques les plus utilisées pour l'évaluation d'un chatbot sont "BiLingual Evaluation Understudy" (BLEU) et la perplexité, METEOR (métrique pour l'évaluation de la traduction avec ORdering explicite) qui étaient à l'origine destinées aux méthodes de traduction automatique [42].

BLEU

BLEU mesure la similitude entre le texte généré et la réponse attendue. Un score de 1,0 représente une correspondance parfaite tandis que 0,0 représente une inadéquation parfaite. Il mesure l'adéquation et la fluidité d'un texte généré en comptant les mots qui correspondent à la réponse attendue. La correspondance des mots a lieu pour chaque mot, par paire, par triolets, etc., également appelés n-grammes. Pour $n = 1$, il considérerait un seul jeton (unigram), pour $n = 2$ une paire de mots est considérée (bigram) et ainsi de suite. L'ordre des grammes (mots) n'est pas significatif dans cette méthode.

METEOR

Pour surmonter certaines des limites de la métrique BLEU, les auteurs ont utilisé METEOR qui est très similaire à BLEU, avec la fonctionnalité supplémentaire de correspondance de

synonymes et de mappage entre la réponse générée et attendue [22]. Il correspond aux mots exacts des deux phrases; chaque mot de la réponse attendue est mappé sur un autre mot de la réponse générée. Des synonymes sont trouvés pour les mots qui ne correspondent pas. Après avoir mis en correspondance les unigrams, le score est calculé en fonction de la précision et du rappel des unigrams.

Perplexité

La perplexité définit la qualité d'un modèle de probabilité pour prédire une donnée de test. La perplexité est l'exponentiation de l'entropie. Après l'apprentissage du modèle, l'ensemble de test peut être utilisé pour calculer la perplexité. Si un modèle «q» existe, la perplexité est donnée par:

$$2^{-1/N \sum_{i=1}^{N} log_2 q(X_i)}$$

X_i est l'ensemble de test ou les mots d'entrée.

N est la longueur de la phrase.

Le modèle fonctionne mieux lorsque la perplexité est moindre.

Critique

Les méthodes d'évaluation d'un Chatbot restent encore une question dans le domaine ouvert. En effet, l'efficacité d'un chatbot ne peut être évaluée que dans un domaine temps réel. La tâche d'évaluation d'un chatbot est subjective et traite étroitement du jugement humain.

L'évaluation de ces modèles a longtemps été un défi posé aux praticiens de l'IA. Étant donné que les mesures d'évaluation quantitative telles que BLEU, METEOR et la perplexité sont loin de l'évaluation des juges humains, en particulier pour les chatbots. D'autres métriques ont également été introduites dans plusieurs articles, mais aucune méthode standard n'existe pour les chatbots jusqu'à présent [22].

L'engagement des utilisateurs est mesuré par la durée du chat entre l'humain et le chatbot. Un plus grand nombre de tours de conversation peut signifier que le chatbot est capable de fournir des réponses afin de garder l'utilisateur engagé. La cohérence est mesurée par la pertinence de la réponse générée. C'est généralement un objectif difficile à atteindre dans une conversation ouverte, mais il est également extrêmement important.

Donc, la meilleure méthode pour évaluer un chatbot est de le faire évaluer par un être humain. Il/elle peut décider si les réponses générées sont significatives et naturelles. La grammaire, l'efficacité et le naturel d'un chatbot ne peuvent être jugés véritablement que par

un humain. Pour un chatbot orienté tâches, on peut demander à l'utilisateur s'il se sent satisfait des réponses ou si le chatbot a pu répondre à ses requêtes.

Travaux sur les chatbots

Travaux existants

Dans ce qui suit, nous examinerons quelques architectures proposées dans des travaux récents.

A Smart Chatbot Architecture based NLP and Machine learning for health care assistance.

Cet article propose une architecture globale composée de 5 composants principaux :

- **Le système de réponse aux questions** est chargé de répondre aux questions fréquemment posées par les utilisateurs avec les réponses associées stockées dans la base de connaissances. Le document suggère qu'un tel système devrait être formé à l'aide d'une formation manuelle et automatisée, où dans le premier les spécialistes mappent les questions fréquemment posées avec leurs réponses relatives et le second consiste à soumettre différents types de documents d'entreprise et à demander au chatbot de se former sur eux.

- **L'environnement** est le lieu où se produisent le moteur fondamental du processus d'apprentissage naturel et la clarification du contexte. Il se compose de deux modules qui sont le moteur NLP qui comprend ce que tout utilisateur dit à un moment donné, puis convertit ce langage en entrée bien définie à l'aide d'un classificateur d'intention et d'un extracteur d'entité qui peuvent être traités ultérieurement par le système et un agent pour gestion des dialogues qui gèrent le contexte réel du dicton utilisateur.

- **Les systèmes Front-End** sont des systèmes orientés client tels que - Facebook Messenger, WhatsApp Business, Slack, Google Hangouts, votre site Web ou votre application mobile, etc.
- **Serveur de nœud / serveur de trafic** c'est le serveur qui traite les demandes de trafic des utilisateurs et les achemine vers les composants appropriés. La réponse des composants internes est souvent acheminée via le serveur de trafic vers les systèmes frontaux.
- **Plugins** qui sont des intégrations personnalisées, votre chatbot peut être intégré à vos systèmes backend existants tels que CRM, base de données, applications de paiement, calendrier et de nombreux outils similaires, pour améliorer les capacités de votre chatbot.

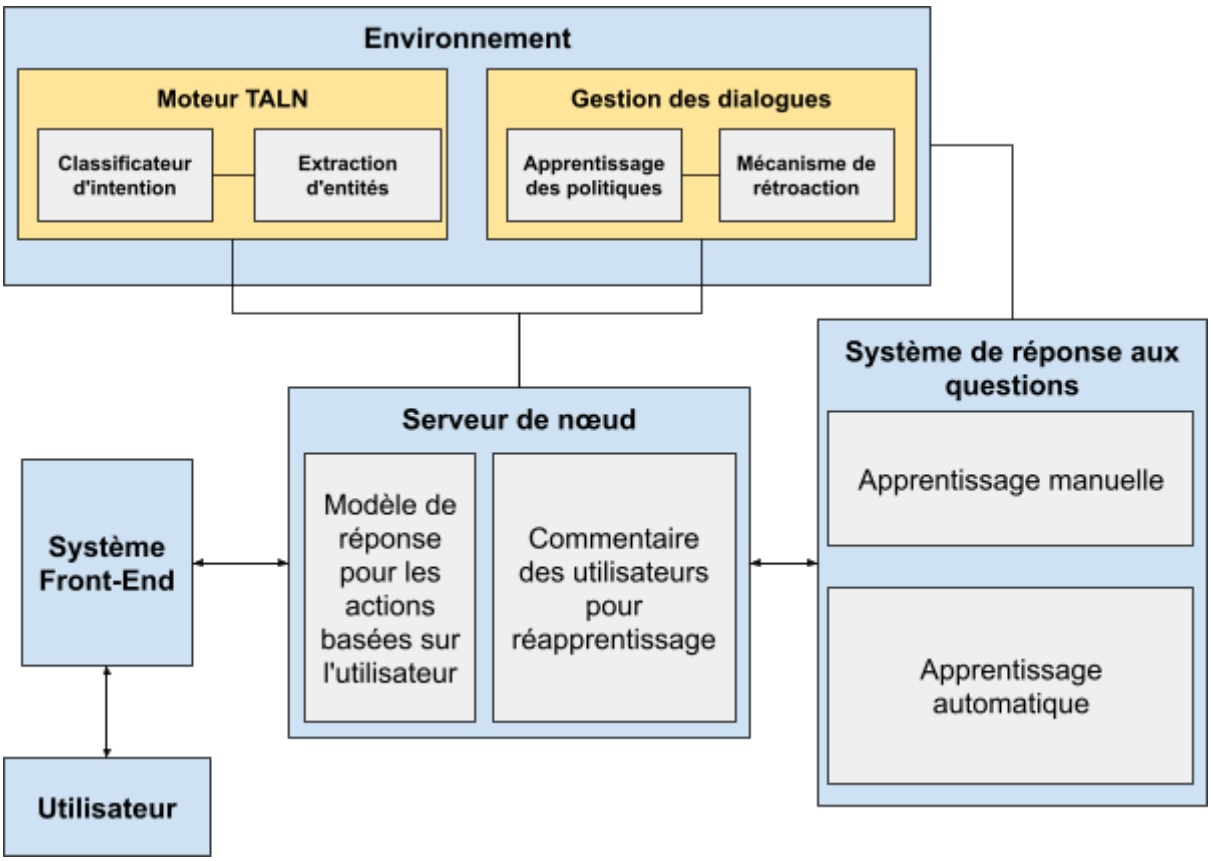


Figure 9. Architecture générale d'un Chatbot 1[source]

Chatbots: History, technology, and applications [source]

Cet article résume une architecture générale basée sur des travaux récents, il inclut certains détails et fonctionnalités des chatbots qui étaient considérés comme cruciaux.

L'architecture proposée est composée de 5 composants principaux qui sont :

- **L'interface utilisateur** est une application utilisant la saisie textuelle ou vocale, telle qu'une application de messagerie comme Facebook, Slack, WhatsApp, WeChat, Viber ou Skype.
- **Composant d'analyse des messages utilisateur** sert à trouver l'intention de l'utilisateur et extrait les entités en suivant des approches de Pattern Matching ou d'apprentissage automatique. Certains services cognitifs peuvent être liés au composant d'analyse des messages de l'utilisateur pour améliorer la précision, tels qu'un correcteur orthographique, Traducteur automatique et analyse des sentiments.
- **Composant de gestion de dialogue** sert à contrôler et mettre à jour le contexte de conversation. Ce composant de gestion de dialogue comprend généralement des modules comme le gestionnaire d'ambiguïté, le gestionnaire de données et d'erreurs.

- **Backend** sert à récupérer les informations nécessaires pour répondre à l'intention de l'utilisateur via des appels d'API externes ou de base de données. Une fois les informations appropriées extraites, elles sont transmises au module de gestion de dialogue puis au module de génération de réponses.
- **Composant de génération de réponse** produit des réponses à l'aide d'un ou plusieurs des trois modèles disponibles : modèles basés sur des règles (Rule Based), modèles basés sur la récupération (Retrieval Based) et modèles basés sur les générations (Generative Based). Le modèle basé sur des règles sélectionne la réponse à partir d'un ensemble de règles. Le modèle basé sur la récupération est plus flexible car il sélectionne la réponse la plus appropriée avec la vérification et l'analyse des ressources disponibles. Le modèle génératif utilise la génération de langage naturel pour répondre dans un langage naturel semblable à celui d'un humain en fonction des dernières entrées et des entrées précédentes.

Lorsque le chatbot produit une réponse, il la présente à l'utilisateur et attend d'avoir un retour.

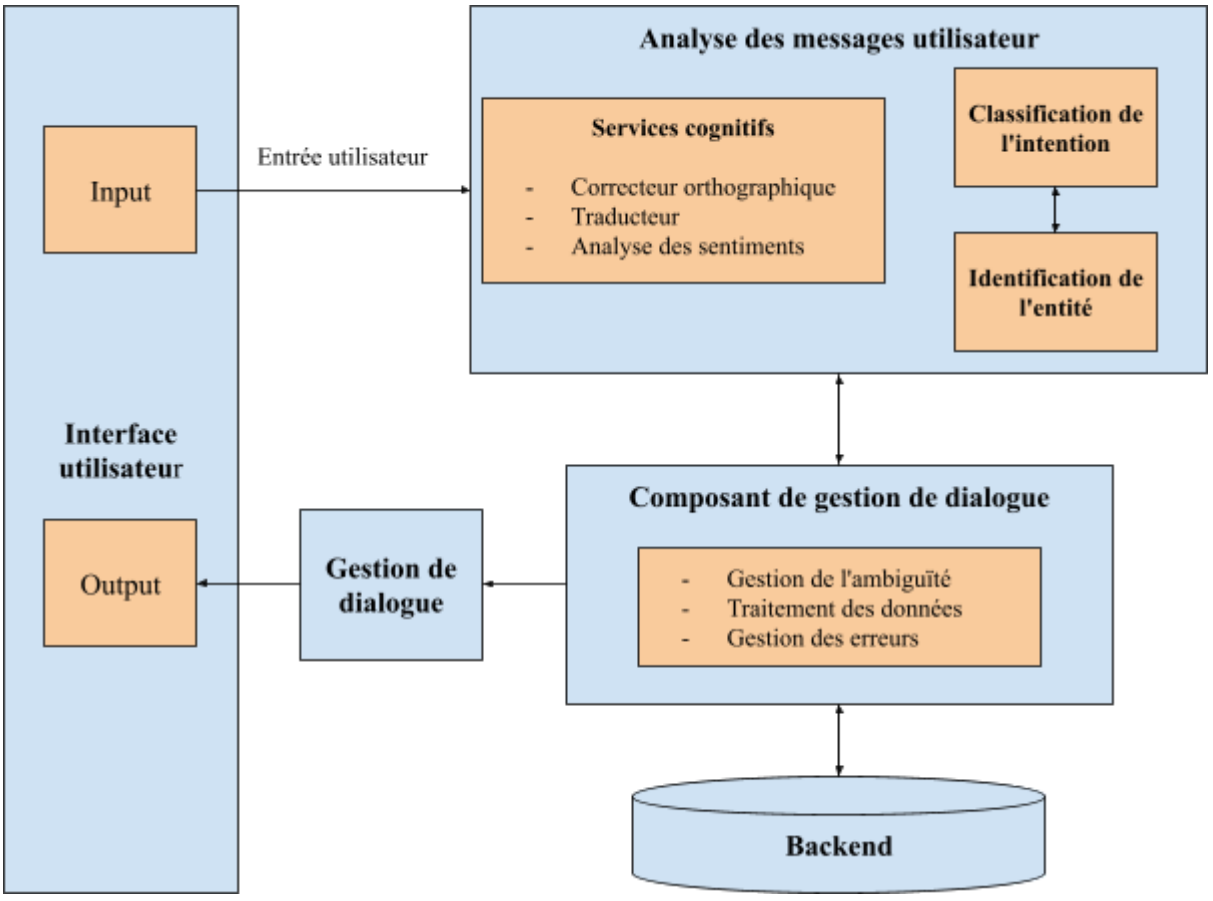


Figure 10. Architecture générale d'un Chatbot 2 [source]

Discussion

- No handling of frequent questions

Travaux de recherche

Ces derniers temps, la conception et la mise en œuvre de chatbots ont reçu une grande attention de la part des développeurs et des chercheurs. Un certain nombre d'études sélectionnées qui ont été réalisées au cours des cinq dernières années sont examinées et expliquées. Notez qu'une comparaison objective est presque impossible à faire compte tenu de l'utilisation de différentes techniques d'évaluation et de mise en œuvre ainsi que du corpus utilisé.

(Shang et al.)

Cet article propose un cadre encoder-decoder pour la modélisation conversationnelle. Le mécanisme d'attention est appliqué au modèle et la recherche par faisceau est utilisée pour le décodage. Le Neural Responding Machine (NRM) est formé avec une grande quantité de données de conversation en un tour collectées à partir d'un service de microblogging. L'annotation humaine a été choisie comme métrique d'évaluation, le modèle ne fonctionne bien que pour une courte conversation textuelle [43].

(Sordoni et al.)

Cette recherche présente un nouveau système de génération de réponses qui peut être formé de bout en bout sur de grandes quantités de conversations Twitter non structurées. Une architecture de réseau neuronal est utilisée pour résoudre les problèmes de rareté qui surviennent lors de l'intégration d'informations contextuelles dans des modèles statistiques classiques, ce qui permet au système de prendre en compte les énoncés de dialogue précédents. Les techniques BLEU, METEOR et l'évaluation humaine ont été utilisées pour l'évaluation, un inconvénient est que le modèle Bag of words est utilisé qui ne prend pas en compte l'ordre dans le contexte [44].

(Zweig et al.)

Cet article propose une approche basée sur un réseau neuronal qui modélise les processus d'attention et d'intention. Il se compose essentiellement de trois réseaux récurrents. Le réseau de codeurs est un modèle au niveau des mots représentant des phrases secondaires sources. Le réseau d'intention est un réseau récurrent qui modélise la dynamique du processus d'intention. Le réseau décodeur est un réseau récurrent qui produit des réponses à l'entrée du côté source. La perplexité a été utilisée pour l'évaluation et le dataset du service de chat du helpdesk In-house a été utilisé comme corpus. Cette approche est uniquement spécifique à l'intention [45].

(Li et al.)

Cette recherche applique le Reinforcement Learning en conjugaison avec le modèle seq2seq. Le modèle simule des dialogues entre deux agents virtuels, en utilisant des méthodes de gradient de politique pour récompenser des séquences qui affichent trois propriétés conversationnelles utiles: l'informativité (tours non répétitifs), la cohérence et la facilité de réponse (liée à la fonction prospective). L'évaluation était basée sur la diversité, la longueur ainsi que sur des juges humains. OpenSubtitles a été utilisé pour entraîner le modèle. Un inconvénient de cette approche est que les récompenses sont heuristiques et ne conduisent donc pas à une conversation idéale [46].

Les chatbots les plus performants

Meena

Contrairement à la plupart des agents conversationnels modernes, qui sont hautement spécialisés, l'équipe de recherche de Google [Meena] introduit un chatbot Meena qui peut discuter de pratiquement tout. Il est construit sur un grand réseau de neurones avec 2,6 milliards de paramètres entraînés sur 341 Go de texte.

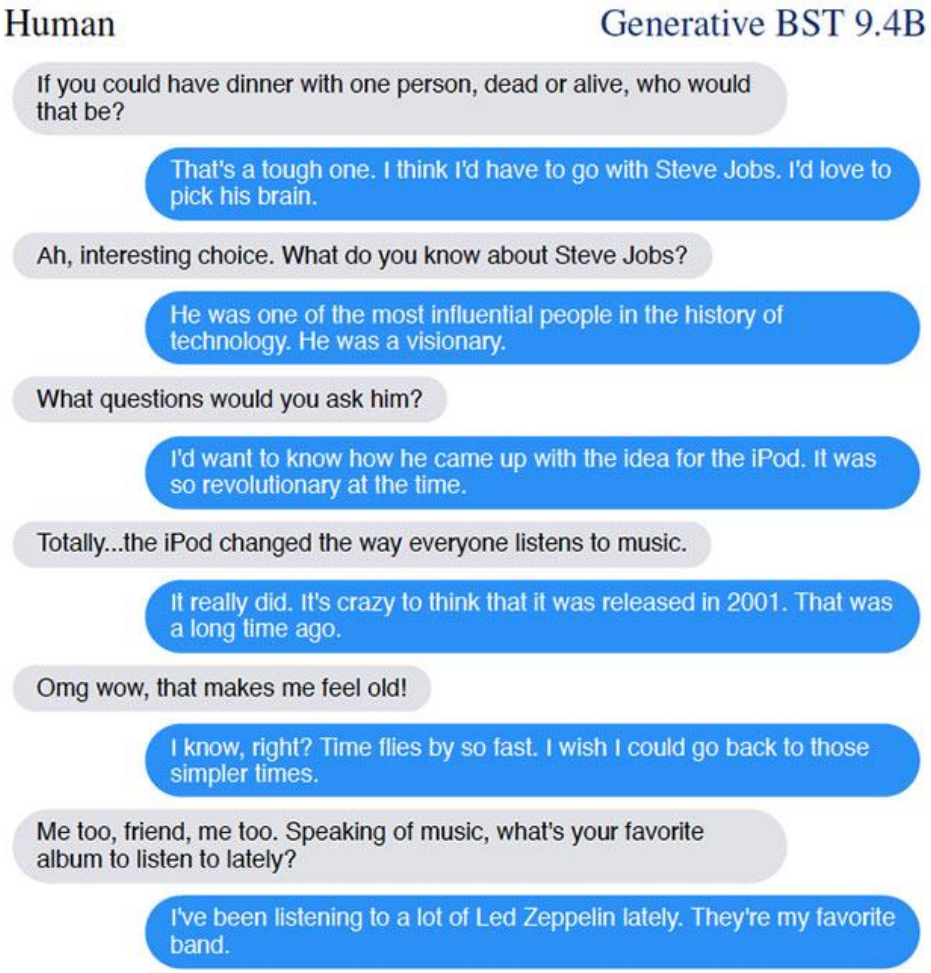
```
Meena: Why don't you have a favorite seafood?
Human: I do
Meena: Then why don't you eat it?
Human: I never said I don't eat my favorite seafood
Meena: Why don't you eat it more?
Human: hmm, I need to go
Meena: Why don't you come back?
Human: why should I?
Meena: Because we miss you.
Human: you're not alone?
Meena: There's more to this, I know.
Human: ok, see you later then
Meena: You're not fooling anyone, you know.
Human: bye
Meena: You're not leaving.
```

```
Human: What's your favorite food?
Meena: Chicken! What about you?
Human: Chicken is an animal. What Chicken dish you like most?
Meena: I'd have to say fried chicken. You?
Human: Vegan here!
Meena: Chicken here!
Human: I don't like you!
Meena: I like you, though!
Human: You were insensitive to me being a Vegan!
Meena: I didn't mean for it to come off that way! I'm sorry! You can still love me?
Human: No. I love chicken more
Meena: Oh. I'm sorry. I love you still.
Human: No need
Meena: Oh, but I need.
Human: That's way too much flirty
Meena: No, it's just right.
Human: Bye Meena! Sorry it couldn't work out between us!
Meena: It's okay, you know I'm still in love with you.
Human: It's not me. its about you
Meena: It's about us.
```

Les chercheurs proposent également une nouvelle métrique d'évaluation humaine pour les chatbots à domaine ouvert, appelée Sensibleness and Specificity Average (SSA), qui peut capturer des attributs importants pour la conversation humaine. Ils démontrent que cette métrique est fortement corrélée avec la perplexité, une métrique automatique qui est facilement disponible. Ainsi, le chatbot Meena, qui est formé pour minimiser la perplexité, peut mener des conversations plus sensées et spécifiques par rapport aux autres chatbots. En particulier, les expériences démontrent que Meena surpasse largement les chatbots de pointe existants en termes de score SSA (79% contre 56%) et comble l'écart avec les performances humaines (86%).

BlenderBot

L'équipe Facebook AI Research [FAIR] montre qu'avec des données de formation et une stratégie de génération appropriées, les modèles à grande échelle peuvent acquérir de nombreuses compétences conversationnelles importantes, telles que l'engagement, les connaissances, l'empathie et la cohérence de la personnalité. Ainsi, pour créer leur agent conversationnel de pointe, appelé BlenderBot, ils ont tiré parti d'un modèle avec des paramètres de 9.4B et l'ont entraîné sur une nouvelle tâche appelée Blended Skill Talk. Les évaluations humaines démontrent que BlenderBot surpasse Meena en comparaison par paires 75% à 25% en termes d'engagement et 65% à 35% en termes d'humanité.



GPT-3

L'équipe de recherche OpenAI [GPT-3] attire l'attention sur le fait que le besoin d'un ensemble de données étiqueté pour chaque nouvelle tâche linguistique limite l'applicabilité des modèles linguistiques. Considérant qu'il existe un large éventail de tâches possibles et qu'il est souvent difficile de collecter un grand ensemble de données d'entraînement étiquetées, les chercheurs suggèrent une solution alternative, qui consiste à étendre les modèles de langage pour améliorer les performances de quelques coups indépendants des

tâches. Ils testent leur solution en entraînant un modèle de langage autorégressif à 175 B de paramètres, appelé GPT-3, et en évaluant ses performances sur plus de deux douzaines de tâches NLP. Le modèle GPT-3 sans réglage fin obtient des résultats prometteurs sur un certain nombre de tâches de PNL, et dépasse même parfois les modèles de pointe qui ont été réglés avec précision pour cette tâche spécifique

Ada

Fondé en 2016, le chatbot IA d'Ada permet aux entreprises ayant de grandes bases de clients de créer des expériences personnalisées tout en réduisant les temps d'attente et en augmentant la satisfaction client. Ada dispose d'un cadre pratique de glisser-déposer, qui ne nécessite aucun codage et prend en charge plus de cent langues. Ada est facile à utiliser et résout jusqu'à 80% des demandes des clients standard, mais il a été conçu pour le support client automatisé à l'échelle de l'entreprise (domaine fermé) et peut donc ne pas convenir aux entreprises naissantes [50].

Conclusion

Dans ce chapitre, nous avons présenté des travaux menés sur les chatbots qui donnent une image claire sur les approches qui peuvent être déployées dans le développement d'un chatbot. La plupart du temps, les versions vanilla sont présentées et peuvent être davantage manipulées et améliorées. Les chatbots modernes utilisent toujours ces techniques jouées mais puissantes. De nos jours, de plus en plus les chatbots utilisent des approches basées sur des réseaux neuronaux, tout en conservant les éléments avantageux des méthodes non basées sur l'IA.

Dans le chapitre suivant, nous illustrerons notre modélisation du chatbot en utilisant les dernières approches d'apprentissage profond et de TALN.

Conception

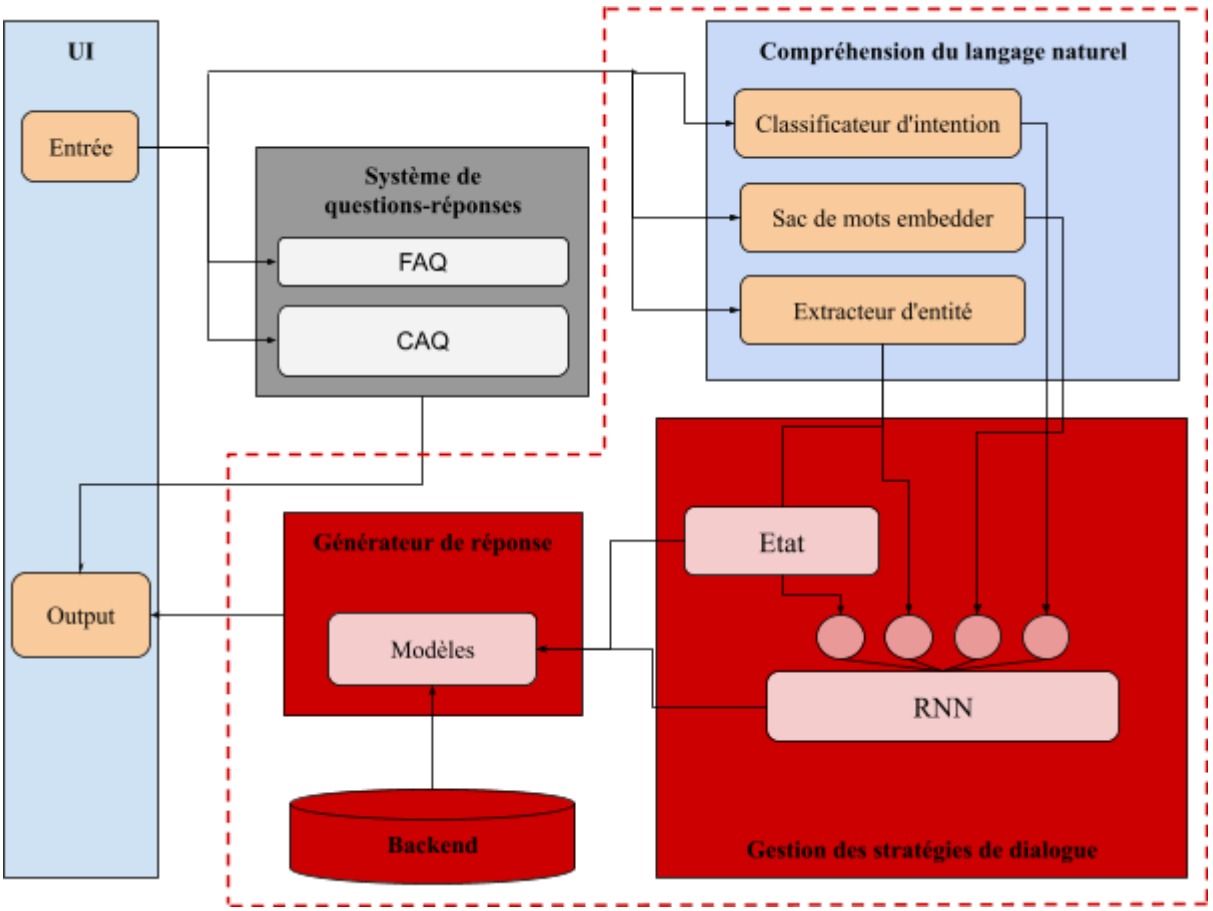
CONCEPTION

Introduction

Modéliser un chatbot ne consiste pas seulement à définir une architecture globale, mais aussi à expliquer comment les composants de cette architecture fonctionnent et interagissent les uns avec les autres, et quelles techniques et modèles ont été utilisés pour les former. Dans ce chapitre nous commencerons par présenter l'architecture globale et approfondirons en expliquant le rôle de chaque composant. Enfin, nous expliquerons comment ces modules interagissent les uns avec les autres en utilisant UML.

Architecture globale du système

Comme le montre la figure, notre architecture de chatbot sera composée de trois systèmes principaux et de cinq modules intérieurs communiquant entre eux. Chaque module représente un lien dans une chaîne qui est essentiellement un pipeline de processus fonctionnant de manière indépendante mais circulant selon un flux de données



- Goal oriented dialogue systems

L’interface Utilisateur

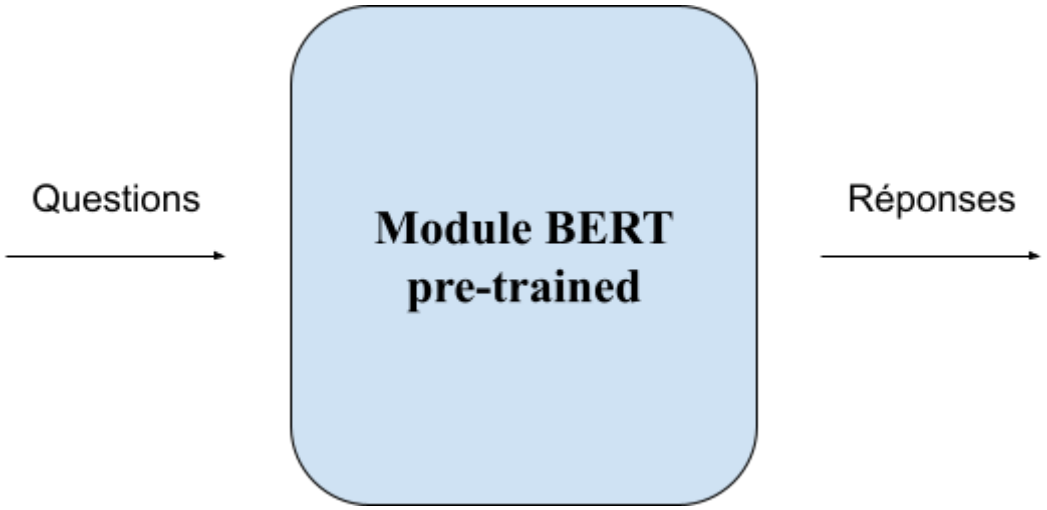
L'interface utilisateur représente ce que les utilisateurs peuvent voir comme entrée/sortie, le cycle de vie du système intérieur de notre chatbot commence lorsqu'il reçoit une demande textuelle ou vocale de l'utilisateur, après avoir traité l'énoncé de l'utilisateur, la réponse générée peut être affichée via l'interface utilisateur.

Système de questions-réponses

Comme son nom l’indique, le système de questions/réponses est chargé de répondre aux questions fréquemment posées par les clients. L’entrée est interprétée par le système de questions-réponses, qui répond ensuite par des réponses appropriées issues de la base de connaissances. Il se compose des modules suivants:

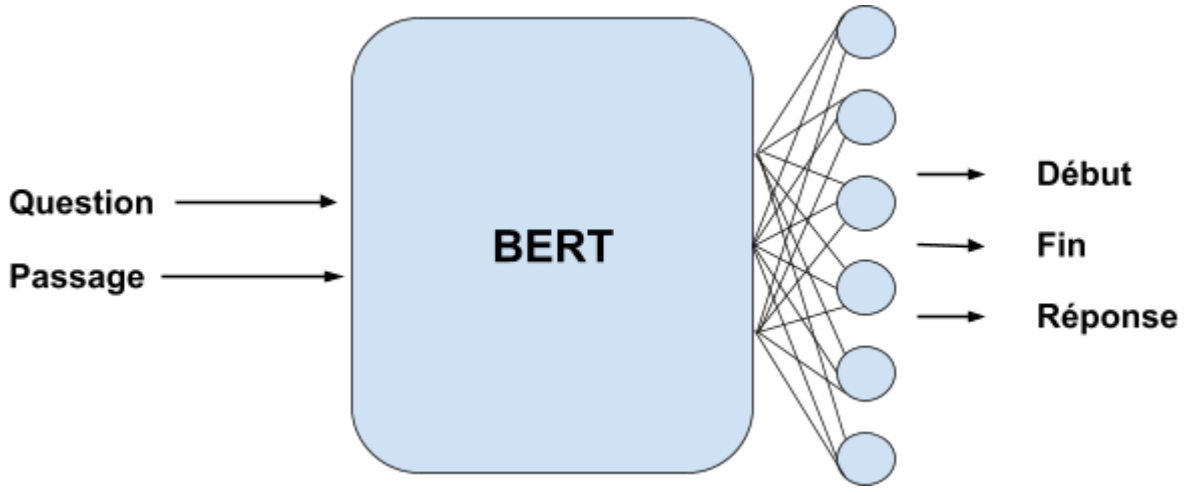
Module FAQ

Ce module est chargé de répondre aux questions fréquemment posées (Frequently Asked Questions) par les utilisateurs, ces questions et réponses doivent être collectées par des spécialistes ou par les agents de l'entreprise. Ensuite, et à l'aide d'un modèle BERT pré-entraîné, nous effectuons la phase d'ajustement en passant des questions en entrée et des réponses en sortie.



Module CAQ

Context Question Answering (CAQ) est chargé de trouver une réponse à une question sur un contexte donné (par exemple, un document d'entreprise), où la réponse à chaque question est un segment du contexte. Pour cette tâche, nous utiliserons un modèle basé sur BERT qui prédit la position de début et de fin de la réponse dans un contexte donné simplement en appliquant deux transformations linéaires aux sorties BERT pour chaque sous-token. La première/seconde transformation linéaire est utilisée pour prédire la probabilité que le sous-token actuel soit la position de début/fin d'une réponse.



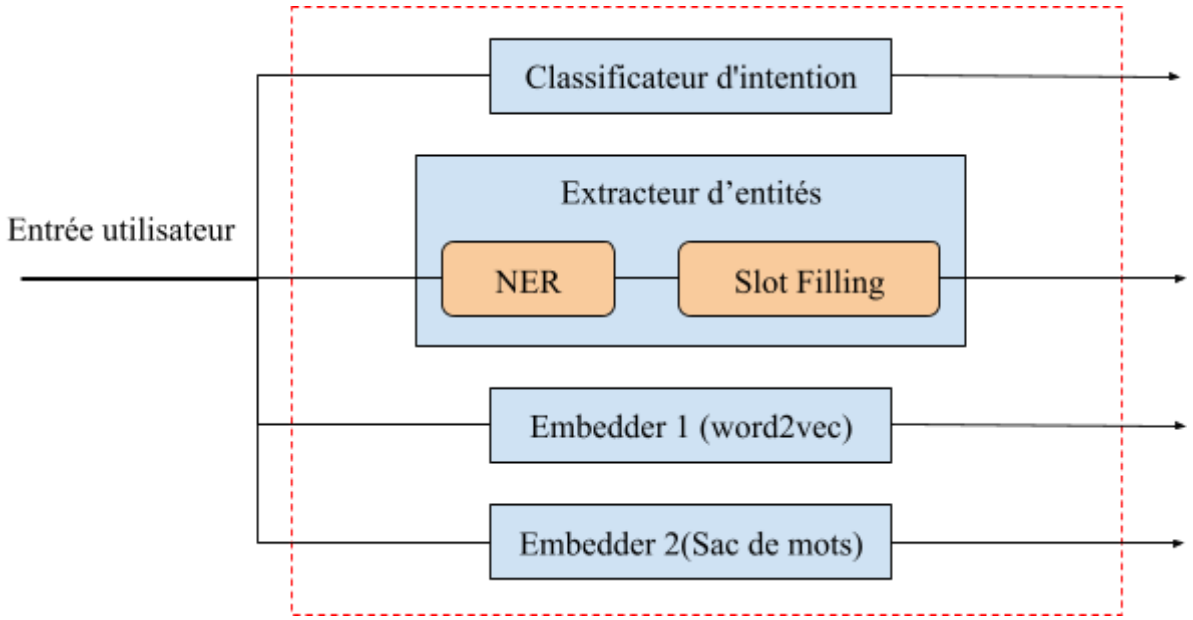
Pour des tâches telles que la classification de texte ou les questions fréquemment posées, nous devons affiner le modèle BERT sur notre dataset. Mais pour les tâches de réponse aux questions contextuelles, nous pouvons même utiliser le modèle déjà formé et obtenir des résultats décents même lorsque notre texte provient d'un domaine complètement différent parce qu'un modèle BERT pré-entraîné comprend le langage en s'entraînant aux tâches MLM et NSP.

Système d'État de dialogue

Un système d'état de dialogue est destiné à résoudre un problème pragmatique de la vie réelle (tickets, réservation...). Dans tout système d'état de dialogue, il y a trois éléments principaux: le module Compréhension du langage naturel (comprendre la saisie de l'utilisateur), la politique de dialogue, y compris la gestion du dialogue pour
décider où chercher la bonne réponse, et le module Réponse Générateur (comment générer une bonne réponse en langage naturel) [57] .

Module de compréhension du langage Naturel

Le module Compréhension du langage naturel est chargé de comprendre ce que veut réellement l'utilisateur en jouant le rôle d'une couche d'abstraction entre l'énoncé de l'utilisateur formaté dans un langage naturel, et le fonctionnement interne du système.



Classificateur d'intention

L'intention indique ici la fonction de l'énoncé, par ex. interroger ou fournir des informations.
[UNFINISHED]

Modèle utilisé

Pour la tâche de classification d'intention, l'approche typique pour la construction d'un tel modèle est basée sur la fourniture d'un nombre relativement important d'exemples pour chacune des intentions. Cela peut avoir du sens lorsqu'un nombre d'intentions est relativement petit et qu'il y a suffisamment de données (par exemple, un petit chatbot organisationnel interne), mais est discutable lorsque le nombre d'intentions est important et que la quantité de données disponibles est relativement faible.

[N] a proposé un capteur d'intentions (Intent Catcher) conçu autour de l'idée qu'en ajoutant un coût supplémentaire pour exiger des connaissances de base des expressions régulières, les utilisateurs seraient en mesure de fournir un plus petit nombre d'exemples au format RegEx pour permettre le suréchantillonnage. En plus de cela, il s'est avéré que l'utilisation directe de RegEx, en plus de l'ensemble de données suréchantillonné, était également utile. Enfin, il était nécessaire de vérifier la ponctuation comme moyen utile de distinguer les déclarations des questions et autres.

Extraction d'entités

L'extraction d'entités s'articule autour de l'idée de générer des paires slot-valeur sont des éléments sémantiques mentionnés dans l'énoncé. L'extraction de ces valeurs peut être résolue à l'aide de deux tâches TALN qui sont : Named Entity Recognition (NER) et Slot Filling.

NER est une sous-tâche d'extraction d'informations qui cherche à localiser et à classer les entités nommées mentionnées dans un texte non structuré en catégories prédéfinies telles que les noms de personnes, les organisations, les emplacements, les expressions temporelles, les quantités, les valeurs monétaires, les pourcentages, etc.

Slot Filling est une étape typique après le NER. Il peut être formulé comme :

“Étant donné qu'une entité d'un certain type et un ensemble de toutes les valeurs possibles de ce type d'entité fournissent une forme normalisée de l'entité.”

Par exemple, il existe une entité de type « Emplacements» : parsi

C'est certainement mal orthographié, l'ensemble de toutes les entités alimentaires connues est {'paris', 'berlin', 'londres', ...}. L'entité la plus proche de l'ensemble donné est paris. Ainsi, la sortie du système de Slot Filling sera paris.

Modèle utilisé

Pour la tâche NER, l'utilisation de BERT peut être problématique car elle utilise la tokenisation Wordpiece, ce qui signifie qu'un mot peut se diviser en plusieurs morceaux (unités de sous-mots) **[UNFINISHED]**

La tâche de Slot Filling est résolue par la recherche de distance de Levenshtein sur toutes les entités connues d'un type donné. Le modèle utilise les tags et les jetons pour effectuer la

normalisation des entités extraites par le modèle NER. La normalisation est effectuée via une recherche Levenshtein floue dans le dictionnaire donné. La sortie de ce composant est un dictionnaire des valeurs de slot trouvées dans les énoncés d'entrée.

La distance de Levenshtein entre deux chaînes a,b (de longueur |a| et |b| respectivement) est donnée par *lev(a,b)* où :

$$\operatorname{lev}(a,b) = \begin{cases} |a| & \text{if } |b| = 0, \\ |b| & \text{if } |a| = 0, \\ \operatorname{lev}(\operatorname{tail}(a), \operatorname{tail}(b)) & \text{if } a[0] = b[0] \\ 1 + \min \begin{cases} \operatorname{lev}(\operatorname{tail}(a), b) \\ \operatorname{lev}(a, \operatorname{tail}(b)) \\ \operatorname{lev}(\operatorname{tail}(a), \operatorname{tail}(b)) \end{cases} & \text{otherwise.} \end{cases}$$

Embedders

Bien que les tâches d'identification d'intention et Slot Filling puissent obtenir l'aspect sémantique, pour mieux comprendre suffisamment l'énoncé de l'utilisateur, nous avons décidé d'ajouter et d'utiliser 2 autres embeddings, l'une est le modèle Word2Vec (pour construire un vecteur pour chaque mot en fonction de la pertinence) et un modèle de sac de mots.

Par exemple, l'aspect sémantique de l'énoncé «Pouvez-vous recommander un restaurant chinois à Manhattan ?» peut être structuré comme :

```
{
  texte: "Pouvez-vous recommander un restaurant chinois à Manhattan?,
  intention: "recherche_restaurant",
  slots: {
    cuisine: "chinois",
    emplacement : "Manhattan"
    ...
  }
}
```

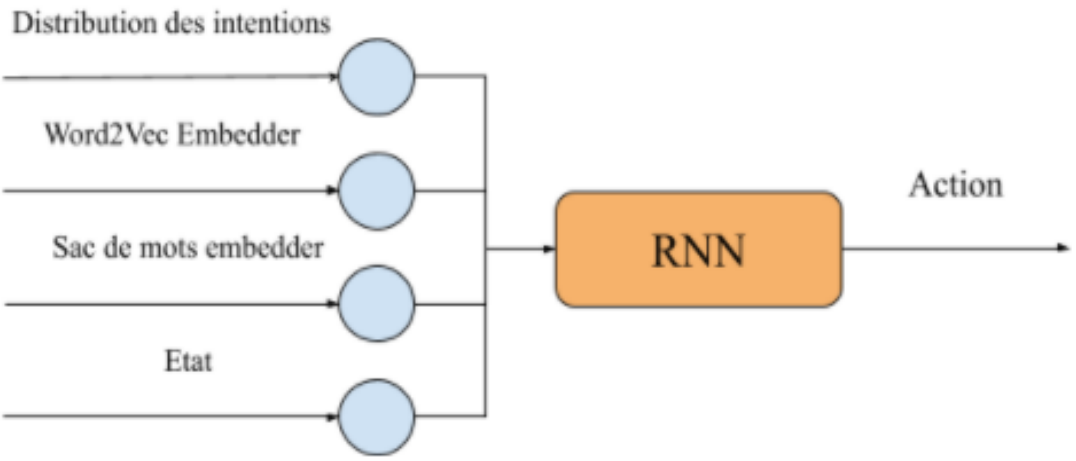
Module de gestion du dialogue

Un module de gestion de dialogue du chatbot décide quelle action doit être entreprise dans l'état de dialogue actuel. Une architecture typique pour un gestionnaire de dialogue se compose de deux parties:

- Politique d'action : cette partie détermine la prochaine action en fonction d'un état donné et de l'interprétation sémantique par le module CLN.

- Suivi de l'état du dialogue : cette partie est chargée de garder une trace du dialogue avec l'utilisateur après chaque entrée.

La politique de notre chatbot est implémentée en tant que Recurrent Neural Network (récurrence sur les entrées utilisateur). Le réseau classe les entrées de l'utilisateur dans l'une des actions système prédéfinies. Des exemples d'actions possibles sont de dire bonjour, de demander l'emplacement de l'entreprise ou de faire un appel API à une base de données. En substance, le module de gestion du dialogue résout la tâche de classification.



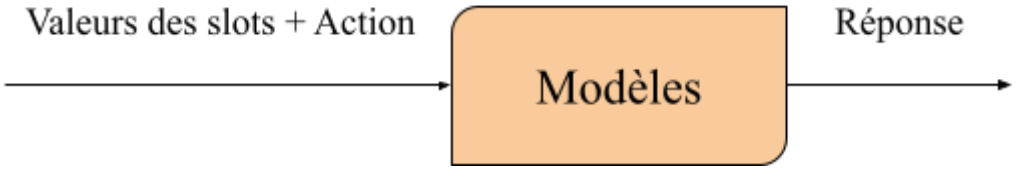
État du dialogue

Le suivi d'état prend les entités extraites des entrées précédentes de l'utilisateur pour construire l'état de l'objectif de l'utilisateur en agrégeant et en suivant les informations à tous les tours du dialogue, cela aide notre chatbot à décider quelle prochaine action il doit entreprendre en fonction de l'état actuel **[UNFINISHED]**.

Générateur de réponse

Ce composant mappe l'acte de dialogue généré par la politique de dialogue à un énoncé en langage naturel. La réponse doit correspondre à un acte de dialogue pour assurer l'achèvement de la tâche, et aussi être naturelle, spécifique et informative [57].

Le modèle de génération de réponse est relativement simple, il consiste à préparer des modèles de phrases contenant des valeurs variables à remplir et chaque action du système dispose d'un ensemble de ces modèles, ces valeurs seront remplies avec la valeur de slot de l'état de dialogue actuel ou de la base de données , la génération de la réponse se fait en sélectionnant au hasard l'un des modèles d'action, puis en remplissant - si nécessaire - les valeurs des variables avec des valeurs de slot ou des résultats de base de données.



Modélisation UML

UML (Unified Modeling Language) est un langage standard pour spécifier, visualiser, construire et documenter les artefacts des systèmes logiciels.

Pour expliquer davantage le fonctionnement global de notre système de chatbot et comment les modules interagissent les uns avec les autres, nous utiliserons certains des diagrammes UML (diagramme de cas d'utilisation, diagrammes d'activité, diagrammes de séquence)

Diagramme de cas d'utilisation

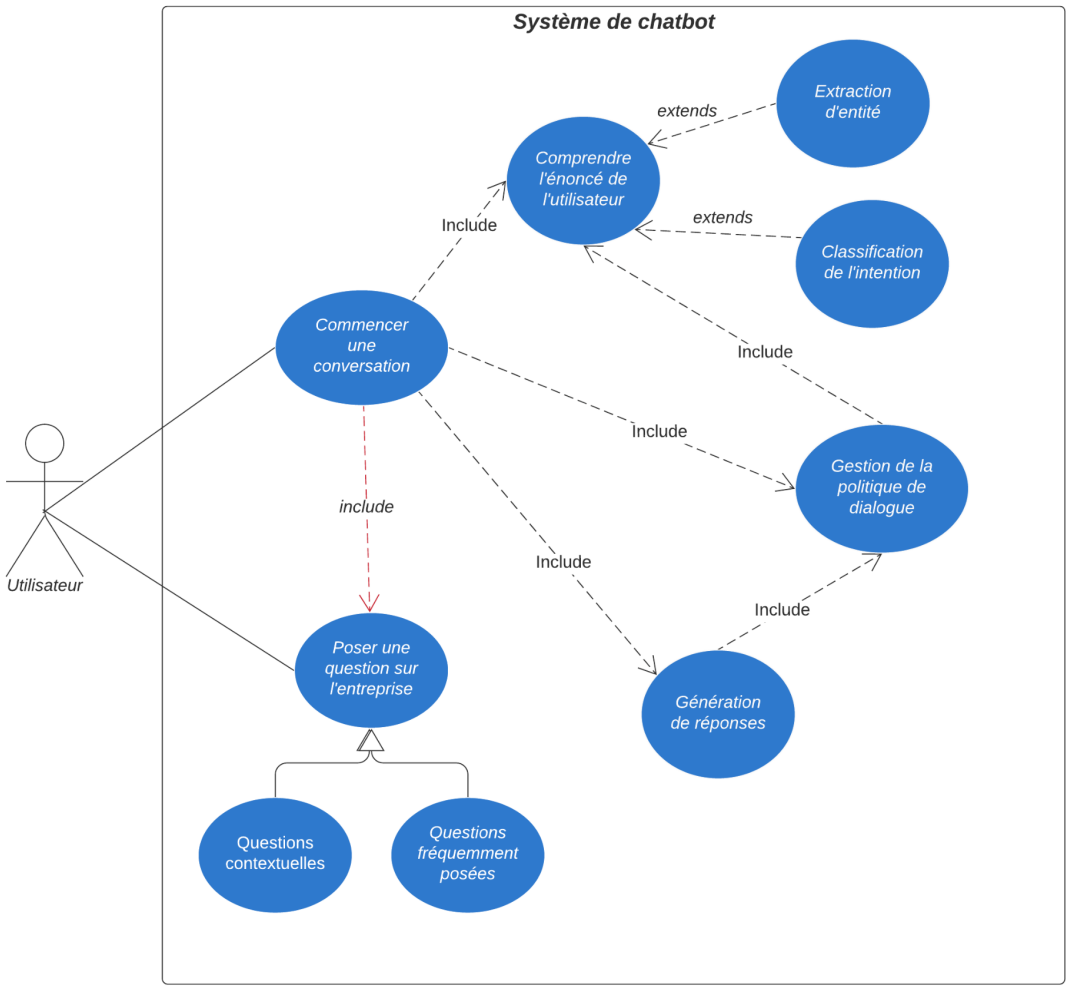
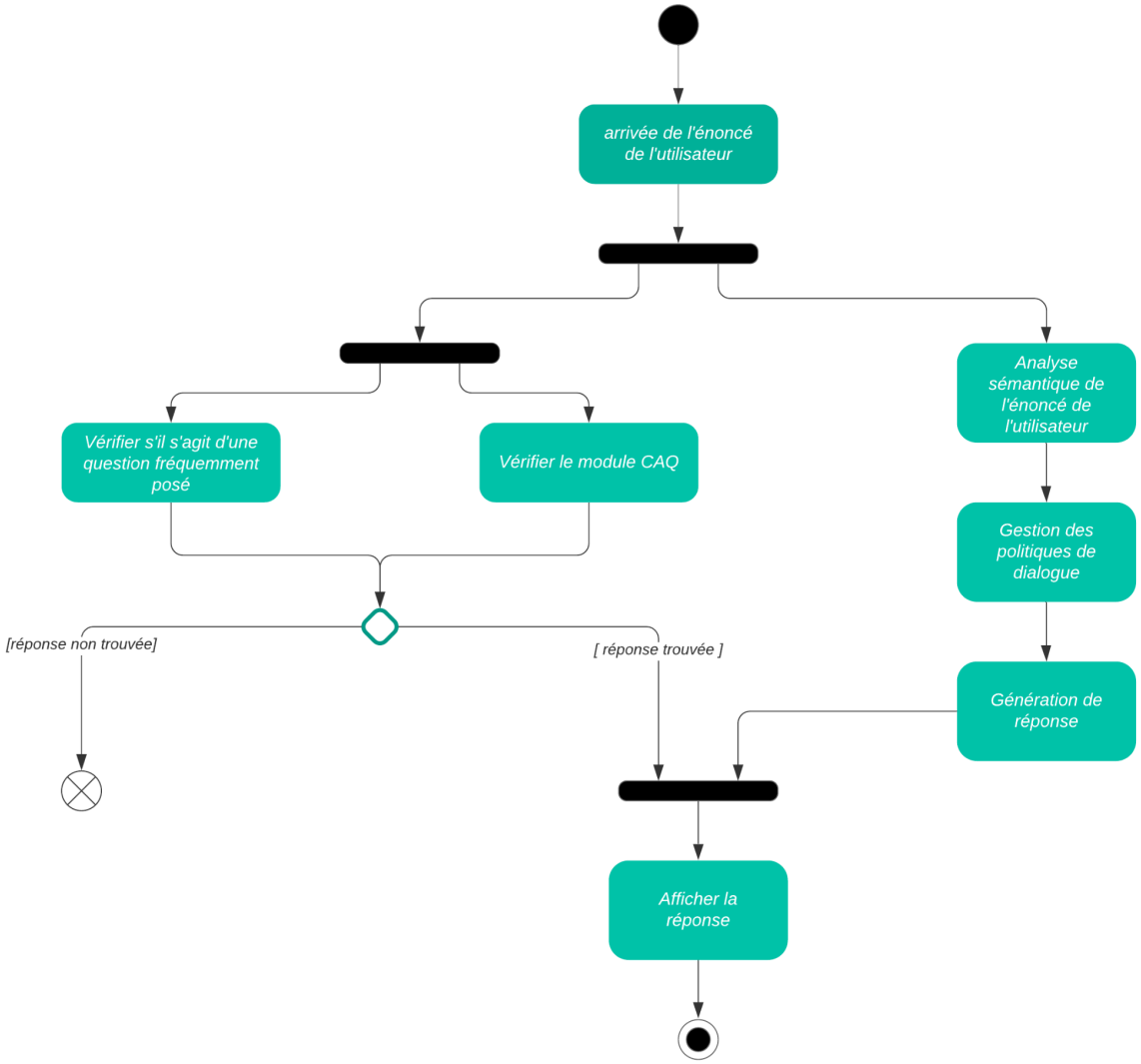
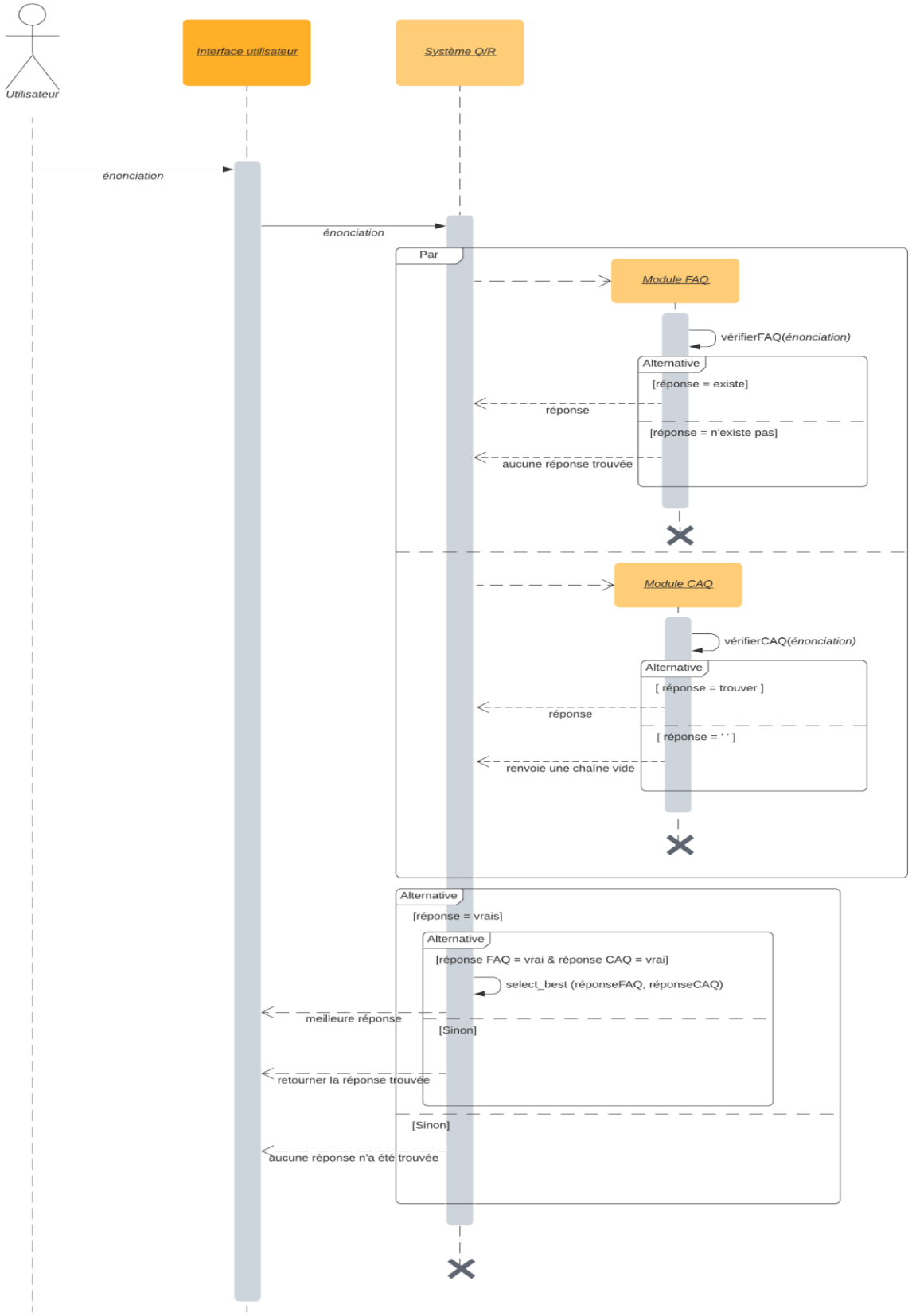


Diagramme d’activité



Diagrammes de séquence



BIBLIOGRAPHIE

[1] Turing, A. M. “I.—COMPUTING MACHINERY AND INTELLIGENCE.” *Mind*, vol. LIX, no. 236, Oct. 1950, pp. 433–60, doi:10.1093/mind/lix.236.433.

[2] “Chatbot: What Is It and How to Build One.” *Paldesk*, 12 Aug. 2019, https://www.paldesk.com/chatbot-what-it-is-and-how-to-build-one/.

[3] Colby, Kenneth Mark, et al. “Artificial Paranoia.” *Artificial Intelligence*, vol. 2, no. 1, 1971, pp. 1–25, doi:10.1016/0004-3702(71)90002-6.

[4] Contributors to Wikimedia projects. “Jabberwacky.” *Wikipedia*, 15 Nov. 2019, https://en.wikipedia.org/w/index.php?title=Jabberwacky&oldid=926273345.

[5] Jwala, K. (2019). Developing a Chatbot using Machine Learning.

[6] Contributors to Wikimedia projects. “Dr. Sbaitso.” Wikipedia, 19 Oct. 2019, https://en.wikipedia.org/w/index.php?title=Dr._Sbaitso&oldid=922071070.

[7] Zemčík, Mgr. Tomáš. “A Brief History of Chatbots.” *DEStech Transactions on Computer Science and Engineering*, no. aicae, Oct. 2019, doi:10.12783/dtcse/aicae2019/31439.

[8] Wallace R.S. The anatomy of a.l.I.C.e Epstein R., Roberts G., Beber G. (Eds.), Parsing the turing test: philosophical and methodological issues in the quest for the thinking computer, Springer Netherlands, Dordrecht (2009), pp. 181-210, 10.1007/978-1-4020-6710-5_13

[12] Molnar, Gyorgy, and Zoltan Szuts. “The Role of Chatbots in Formal Education.” *2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY)*, IEEE, 2018, http://dx.doi.org/10.1109/sisy.2018.8524609.

[13] Shewan, Dan. “10 of the Most Innovative Chatbots on the Web.” *WordStream*, 25 Apr. 2021, https://www.wordstream.com/blog/ws/2017/10/04/chatbots.

[14] Team, Expert. ai. “Chatbots: What Is a Chatbot and Why Is It Important?” *Expert.Ai*, 16 Mar. 2020, https://www.expert.ai/blog/chatbot/.

[15] Schlicht, Matt. “The Complete Beginner’s Guide To Chatbots.” *Chatbots Magazine*, 10 May 2018, https://chatbotsmagazine.com/the-complete-beginner-s-guide-to-chatbots-8280b7b906ca.

[16] Bika, Nikoletta. “Chatbot Use Cases: What Bots Can Do per Industry and Function.” *Acquire*, 22 May 2020, https://acquire.io/blog/chatbot-use-cases/.

[17] Weizenbaum, Joseph. “ELIZA—A Computer Program for the Study of Natural Language Communication between Man and Machine (1966).” *Ideas That Created the Future*, The MIT Press, 2021, pp. 271–78, http://dx.doi.org/10.7551/mitpress/12274.003.0029.

[18] Bruno Marietto, Maria das Gracas, et al. “Artificial Intelligence Markup Language: A Brief Tutorial.” *International Journal of Computer Science & Engineering Survey*, vol. 4, no. 3, June 2013, pp. 1–20, doi:10.5121/ijcses.2013.4301.

[19] Trivedi A., Gor V., Thakkar Z. Chatbot generation and integration: A review International Journal of Advance Research, Ideas and Innovations in Technology, 5 (2) (2019), pp. 1308-1311 XX.XXX/IJARIIT-V5I2-1840

[20] Arsovski S., Muniru I., Cheok A. Analysis of the chatbot open source languages aiml and chatscript: A review (2017), 10.13140/RG.2.2.34160.15367

[21] “Fun with Markov Chain Chatbots.” *ChatBot Pack*, 22 Sept. 2018, https://www.chatbotpack.com/markov-chain-chatbot/.

[22] Agarwal, Ritu, and Mani Wadhwa. “Review of State-of-the-Art Design Techniques for Chatbots.” *SN Computer Science*, vol. 1, no. 5, July 2020, doi:10.1007/s42979-020-00255-3.

[23] Bradeško L, Mladenić D. A survey of chabot systems through a loebner prize competition. Res Net. 2012;2:1–4.

[24] Ramesh, Kiran, et al. “A Survey of Design Techniques for Conversational Agents.” Communications in Computer and Information Science, Springer Singapore, 2017, pp. 336–50, http://dx.doi.org/10.1007/978-981-10-6544-6_31.

[25] Jia, Jiyou. “CSIEC: A Computer Assisted English Learning Chatbot Based on Textual Knowledge and Reasoning.” *Knowledge-Based Systems*, vol. 22, no. 4, May 2009, pp. 249–55, doi:10.1016/j.knosys.2008.09.001.

[26] Lin, Lue, et al. “A Web-Based Platform for Collection of Human-Chatbot Interactions.” *Proceedings of the Fourth International Conference on Human Agent Interaction*, ACM, 2016, http://dx.doi.org/10.1145/2974804.2980500.

[27] Wang, Sun-Chong. “Artificial Neural Network.” *SpringerLink*, https://link.springer.com/chapter/10.1007/978-1-4615-0377-4_5.

[28] z_ai. “Deep Learning for NLP: ANNs, RNNs and LSTMs Explained!” *Towards Data Science*, 28 Aug. 2020, https://towardsdatascience.com/deep-learning-for-nlp-anns-rnns-and-lstms-explained-95866c1db2e4.

[29] "What Is Deep Learning?" *Math Works*, https://www.mathworks.com/discovery/deep-learning.html. Accessed 30 May 2021.

[30] "What Is Deep Learning and How Does It Work?" *Towards Data Science*, 11 Aug. 2020, https://towardsdatascience.com/what-is-deep-learning-and-how-does-it-work-2ce44bb692ac.

[31] Brownlee, Jason. "What Is Deep Learning?" *Machine Learning Mastery*, 15 Aug. 2019, https://machinelearningmastery.com/what-is-deep-learning/.

[32] Olah, Christopher. "Understanding LSTM Networks -- Colah's Blog." *Colah's Blog*, 27 Aug. 2017, http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[33] Chung J., Gulcehre C., Cho K., Bengio Y. Empirical evaluation of gated recurrent neural networks on sequence modeling (2014)

[34] Hochreiter, Sepp. "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions." *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 06, no. 02, Apr. 1998, pp. 107–16, doi:10.1142/s0218488598000094.

[35] Xu, Anbang, et al. "A New Chatbot for Customer Service on Social Media." *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ACM, 2017, http://dx.doi.org/10.1145/3025453.3025496.

[36] Vinyals, Oriol, and Quoc Le. "A Neural Conversational Model." *ArXiv.Org*, 19 June 2015, https://arxiv.org/abs/1506.05869.

[37] Allen, James F. "Natural Language Processing." Encyclopedia of Cognitive Science, John Wiley & Sons, Ltd, 2006, http://dx.doi.org/10.1002/0470018860.s00078.

[38] Garbade, Dr. Michael J. "A Simple Introduction to Natural Language Processing." *Becoming Human: Artificial Intelligence Magazine*, 15 Oct. 2018, https://becominghuman.ai/a-simple-introduction-to-natural-language-processing-ea66a1747b32.

[39] "What Is Natural Language Understanding and How Is It Different from NLP?" *Expert.Ai*, 10 Apr. 2020, https://www.expert.ai/blog/natural-language-understanding-different-nlp/.

[40] Jung, Sangkeun. "Semantic Vector Learning for Natural Language Understanding." *Computer Speech & Language*, vol. 56, July 2019, pp. 130–45, doi:10.1016/j.csl.2018.12.008.

[41] Chowdhury, Gobinda G. "Natural Language Processing." *Annual Review of Information Science and Technology*, vol. 37, no. 1, Jan. 2005, pp. 51–89, doi:10.1002/aris.1440370103.

[42] Sordoni, Alessandro, et al. “A Neural Network Approach to Context-Sensitive Generation of Conversational Responses.” *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, 2015, http://dx.doi.org/10.3115/v1/n15-1020.

[43] Shang L, Lu Z, Li H. Neural responding machine for short-text conversation. 2015; 1577–1586.

[44] Galley M, Auli M, Brockett C, Ji Y, Mitchell M, Gao J. A neural network approach to context-sensitive generation of conversational responses. arXiv preprint arXiv:1506.06714. 2015.

[45] Zweig V. Attention with Intention for a Neural Network Conversation Model. 2015; 1–7.

[46] Li, Jiwei, et al. “Deep Reinforcement Learning for Dialogue Generation.” *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2016, http://dx.doi.org/10.18653/v1/d16-1127.

[47] Bizzaco, Michael. “What Is Amazon’s Alexa, and What Can It Do?” *Digital Trends*, 30 Mar. 2021, https://www.digitaltrends.com/home/what-is-amazons-alexa-and-what-can-it-do/.

[48] “Google Assistant.” *Google Assistant*, https://assistant.google.com/. Accessed 31 May 2021.

[49] “Watson Assistant - Intelligent Virtual Agent.” *IBM Watson*, https://www.ibm.com/cloud/watson-assistant/. Accessed 31 May 2021.

[50] Bibla, Michael. “The Top 24 AI Chatbots.” *Ada*, 12 Mar. 2021, https://www.ada.cx/ai-chatbot.

[51] “Siri.” *Apple*, https://www.apple.com/siri/. Accessed 31 May 2021.

[52] View, Log In or Sign Up to. “Apple Siri Features, Use, Advantages, Disadvantages & Using of Siri For Learning.” *Science Online*, 17 May 2019, https://www.online-sciences.com/technology/apple-siri-features-use-advantages-disadvantages-using-of-siri-for-learning/.

[53] “Cortana - Your Personal Productivity Assistant.” *Cortana - Your Personal Productivity Assistant*, https://www.microsoft.com/en-us/cortana. Accessed 1 June 2021.

[54] Adamopoulou, Eleni, and Lefteris Moussiades. “Chatbots: History, Technology, and Applications.” *Machine Learning with Applications*, vol. 2, Dec. 2020, p. 100006, doi:10.1016/j.mlwa.2020.100006.

[55] Hirschberg, Julia, and Christopher D. Manning. “Advances in Natural Language Processing.” *Science*, vol. 349, no. 6245, July 2015, pp. 261–66, doi:10.1126/science.aaa8685.

[56] Koidan, Kate. “Task-Oriented Dialog Agents: Recent Advances and Challenges.” *TOPBOTS*, 6 Oct. 2020, https://www.topbots.com/task-oriented-dialog-agents/.

[57] Wang, Haixun. “An Annotated Reading List of Conversational AI - AI Graduate - Medium.” *AI Graduate*, 7 Jan. 2020, https://medium.com/x8-the-ai-community/a-reading-list-and-mini-survey-of-conversational-ai-32fceea97180.

[1_T] Maxime. (2020, March 5). What is a Transformer? - Inside Machine learning - Medium. *Inside Machine Learning*. https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04

[2_T] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017, June 12). *Attention is all you need*. ArXiv.Org. https://arxiv.org/abs/1706.03762

[3_T] Alammar, J. (n.d.). *The Illustrated Transformer*. Retrieved June 16, 2021, from http://jalammar.github.io/illustrated-transformer/

[4_T] *Transformer model for language understanding*. (n.d.). TensorFlow. Retrieved June 16, 2021, from https://www.tensorflow.org/text/tutorials/transformer

[1_B] Devlin, Jacob, et al. “BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding.” *ArXiv.Org*, 11 Oct. 2018, https://arxiv.org/abs/1810.04805.

[2_B] Horev, Rani. “BERT Explained: State of the Art Language Model for NLP.” *Towards Data Science*, 17 Nov. 2018, https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270.

[3_B] Rogers, Anna, et al. “A Primer in BERTology: What We Know About How BERT Works.” *Transactions of the Association for Computational Linguistics*, vol. 8, Dec. 2020, pp. 842–66, doi:10.1162/tacl_a_00349.

[4_B] ___. “How the Embedding Layers in BERT Were Implemented - ___.” *Medium*, 3 Jan. 2021, https://medium.com/@_init_/why-bert-has-3-embedding-layers-and-their-implementation-details-9c261108e28a.

[Meena] Adiwardana, Daniel, et al. “Towards a Human-like Open-Domain Chatbot.” *ArXiv.Org*, 27 Jan. 2020, https://arxiv.org/abs/2001.09977.

[FAIR] Roller, Stephen, et al. “Recipes for Building an Open-Domain Chatbot.” *ArXiv.Org*, 28 Apr. 2020, https://arxiv.org/abs/2004.13637.

[GPT-3] Brown, Tom B., et al. “Language Models Are Few-Shot Learners.” *ArXiv.Org*, 28 May 2020, https://arxiv.org/abs/2005.14165v2.

[VB] Bourne, Vanson. “Impact of Chatbots and AI on the Customer Journey.” *Bold360*, https://www.bold360.com/ai-report. Accessed 3 July 2021.