République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Chadli Bendjedid
Faculté des Sciences et de la Technologie
Département d'Informatique

الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي
جامعة الشاذلي بن جديد
كلية العلوم و التكنولوجيا
قسم الاعلام الالي

جامعة الشاذلي بن جديد
UNIVERSITE CHADLI BENDJEDID

MEMOIRE

Présenté par

ABBACI HOUDA

Pour l'obtention de diplôme de

MASTER

Filière : Informatique

Spécialité : Systèmes Informatiques Intelligents

Thème

Elicitation des Préférences dans les Systèmes de Recommandation Conversationnels

Soutenu le : 30/06/2022

Devant le Jury composé de :

Qualité	Nom et Prénom	Grade	Université
Président	Mr. Chemmam. C	MAB	Chadli Bendjedid El-Tarf
Rapporteur	**Mme. Maatallah. M**	**MCB**	**Chadli Bendjedid El-Tarf**
Examinateur	Mme. Makhlouf. A	MCB	Chadli Bendjedid El-Tarf

Année Universitaire : 2021/2022

Remerciements

Mes remerciements les plus sincères vont à Mon Mari ***M^r^ CHABBI TAREK*** *pour son soutien moral et physique et son aide dans mon travail par ces précieux conseils, réponse et recommandations*

A mon encadreur, ***M^e^. MAATALLAH*** *j'adresse mon plus profonde reconnaissance pour son bon encadrement et pour les conseils fructueux qu'il n'a cessé de ma prodiguer.*

Je tiens à remercier ***M^e^. KADI SONIA*** *La directrice de la direction de la poste et de télécommunication de la Wilaya de Taref. Une personne qui m'inspire le plus grand respect.*

Je deviens chaque bribe de ma connaissance à mes enseignants au département informatique qui ont si bien mené leur noble quête d'enseigner les bases d'Informatique « Intelligence Artificielle ». Je les remercie non seulement pour le savoir qu'ils nous ont transmis, mais aussi pour la fierté et l'ambition que leurs personnes nous aspirent.

Que messieurs les membres du jury trouvent ici l'expression de notre reconnaissance pour avoir accepté d'évaluer mon travail.

Et toutes les personnes qui ont contribué de près ou de loin au bon déroulement de ce travail.

Un petit bout de chemin certes, mais un grand enrichissement.

Dédicaces

A mon mari TAREK

A mes très chers parents, source de vie, d'amour et d'affection

A mes chers enfants, Douaa, Mohamed Yahia et Youcef source de joie et de bonheur

A toute ma famille, source d'espoir et de motivation

A vous cher lecteur

Résumé

Les Systèmes de Recommandation Conversationnels (SRC) constituent ces dernières années l'axe de recherche le plus visé dans le domaine de la recommandation. Cependant, le point fort de ces systèmes est leur capacité à prendre en considération les préférences des utilisateurs, en temps réels, à travers leur conversation. Cela nécessite l'utilisation des techniques du Traitement Automatique du Langage Naturel (TALN) afin de pouvoir éliciter les préférences des utilisateurs envers les produits pour enrichir leurs profils. Les développements récents de l'apprentissage automatique dans le TALN, nous rapprochent de la possibilité d'exprimer nos besoins spécifiques en langage naturel et obtenir une réponse à nos questions à la machine.

Nous nous intéressons dans ce travail aux méthodes d'élicitation des préférences, en particulier l'analyse des sentiments, à l'aide des techniques du TALN. Pour explorer cette voie, nous avons adapté et étendu nos techniques de modélisation basées sur les transformateurs BERT et LSTM, qui sont les plus récents modèles basés sur l'apprentissage en profondeur dédiés à la tâche d'analyse des sentiments à partir des conversations. Deux modèles BERT ont été utilisés pour déduire le sentiment des utilisateurs. Les performances des modèles ont été testées sur deux ensembles de données différents ; le premier contient des commentaires sur les films, et l'autre des commentaires sur les applications Playstore. Les résultats obtenus étaient très satisfaisants, et montrent que les modèles BERT donnent des résultats intéressants en termes de performances avec le modèle LSTM.

Mots-clés: Système de Recommandation Conversationnel, Elicitation de Préférence, Analyse de Sentiment, BERT, Traitement Automatique du Langage Naturel, Deep Learning, Apprentissage Automatique.

Abstract

Conversational Recommender Systems (CRS) have been the most targeted area of research in the field of recommendation in recent years. However, the strong point of these systems is their ability to take into account the users preferences, in real time, through their conversation. This requires the use of Automatic Natural Language Processing (ANLP) techniques in order to be able to elicit user preferences towards products to enrich their profiles. Recent developments on machine learning in NLP bring us closer to the possibility of expressing our specific needs in natural language and getting our questions answered by the machine. We are interested in this work by the preference elicitation methods, especially sentiment analysis using NLP techniques. To explore this road, we have adapted and extended our modeling techniques based on BERT and LSTM transformers, which are the most recent deep learning based models dedicated to the sentiment analysis task from conversations. Two BERT models were used to infer user sentiment. The performance of the models was tested on two different Datasets; the first one contains comments on movies, and the other Playstore applications comments. The results obtained were

very satisfactory, and show that the BERT models give interesting results in terms of performance with the LSTM model.

Keywords: Conversational Recommendation System, Preference Elicitation, Sentiment Analysis, BERT, Natural Language Processing, Deep Learning, Machine Learning.

الملخص

أنظمة التوصية للمحادثة اصبحت أكثر مجالات البحث المستهدفة في مجال التوصية في السنوات الأخيرة. ومع ذلك ، فإن النقطة القوية لهذه الأنظمة هي قدرتها على مراعاة تفضيلات المستخدمين ، في الوقت الفعلي ، من خلال محادثتهم. يتطلب هذا استخدام تقنيات المعالجة الالية للغات الطبيعية حتى تتمكن من إثارة تفضيلات المستخدم تجاه المنتجات لإثراء ملفات التعريف الخاصة بهم. تقربنا التطورات الأخيرة في التعلم الآلي في البرمجة اللغوية العصبية من إمكانية التعبير عن احتياجاتنا الخاصة بلغة طبيعية والحصول على إجابات لأسئلتنا بواسطة الآلة.نحن مهتمون في هذا العمل بطرق استنباط التفضيل وتحليل المشاعر باستخدام تقنيات البرمجة اللغوية العصبية. لاستكشاف هذا المسار ، قمنا بتكييف وتوسيع تقنيات النمذجة لدينا استنادًا إلى محولات BERT و LSTM ، وهي أحدث النماذج القائمة على التعلم العميق والمخصصة لمهمة تحليل المشاعر من المحادثات. تم استخدام نموذجين من BERT لاستنتاج شعور المستخدم. تم اختبار أداء النماذج على مجموعتي بيانات مختلفتين ، مجموعة بيانات تحتوي على تعليقات على الأفلام ، وأخرى على تطبيقات Playstore. كانت النتائج التي تم الحصول عليها مرضية للغاية ، وتبين أن نماذج BERT تعطي نتائج جيدة من حيث الأداء مع نموذج LSTM

الكمات المفتاحية : نظام توصية المحادثة ، استنباط التفضيلات ، تحليل المشاعر ، BERT ، معالجة اللغة الطبيعية ، التعلم العميق ، التعلم الآلي.

Table des matières

Liste des figures

Liste des tableaux

Liste des acronymes

CRS	SRC	*Conversational Recommender System ; Système de Recommandation Converssationnel;*
NLP	TALN	*Naturel Language Processing; Traitement Automatique du Longuage Naturel*
BERT	BERT	*Bidirectional Encoder Représentation from Transformers; Représentation d'encodeurs bidirectionnels à partir de transformateurs*
GRU	URF	*Gated Recurrent Unit; Unité récurrente fermée*
MF	FM	*Matrix Factorization; Factorisation Matricielle*
AMT	AMT	*Amazon Mechanical Turk;*
RNN	RNN	*Neuronal Net work Récurrent ; Réseaux de neurone récurent*
VAE	VAE	*Variational AutoEncoder Model; Modèle d'AutoEncodeur Variationnel*
MAB	MAB	*Multi Armed Bandit ; Bandit Multi Bras*
UCB	UCB	*Upper Conversational Bandis ; Algorithme de borne supérieur de confiance*
CNN	CNN	*Convolutif Net work Neuronal; Réseaux de neurone Convolutionnel*
SAUR	SAUR	*Système Ask User Response; Système Question Reponse*
CHATBOT	CHATBOT	*Chatbot ; ChatRobot*
LSTM	LSTM	*Long-Short-Term Memory; Memoire à Long Court Terme ;*
GNN	GNN	*Graphique Neuronal Net work ; Réseaux de Neurone Graphique*
DRL	DRL	*Deep Reinforcement Learning; Apprentissage par renforcement profond*
RL	RL	*Reinforcement Learning; Renforcement profond*
POMDP	POMDP	*Partially Observed Markov Decision Process : Processus Decisional de Markov Partiellement Observée*
DQN	DQN	*Deep Q-Network ;*
LSTM	LSTM	*Long-Short-Term Memory; Memoire à Long Court Terme ;*
GNN	GNN	*Graphique Neuronal Net work ; Réseaux de Neurone Graphique*
DRL	DRL	*Deep Reinforcement Learning; Apprentissage par renforcement profond*
RL	RL	*Reinforcement Learning; Renforcement profond*
POMDP	POMDP	*Partially Observed Markov Decision Process : Processus Decisional de Markov Partiellement Observée*
DQN	DQN	*Deep Q-Network ;*
DDPG	DDPG	*Deep Deterministic Political Gradient; Gradient Politique Déterministe Profond*
SAUR	SAUR	*System Ask User Response ; Système Question Réponse*
CRM	CRM	*Conversational recommendation model; Modèle de recommandation conversationnel*
Google AI	Google AI	*Google Artificial Intelligence ; Intelligence Artificielle de Google*
BERT-ANA-SENT	BERT-ANA-SENT	*Représentation d'encodeurs bidirectionnels à partir de transformateurs-Analyse des Sentiments*
TPU Cloud	TPU Cloud	*Tensor Processing Unit Cloud ; Nuage d'Unité de Traitement de Tenseur*
MLM	MLM	*Masque Language Model*
CE-NCF	CE-NCF	*filtrage collaboratif neuronal explicatif*
IMDB	IMDB	*Internet Movie Database*

1. Introduction Générale

L'essor du Web ces dernières années et l'augmentation massive de la quantité de données disponibles sur le Web signifient que les utilisateurs sont souvent confrontés à de longues recherches qui prennent des heures et qui finalement n'aboutissent pas à des résultats inattendus. Ce problème a conduit à l'émergence d'un domaine de recherche consacré aux **Systèmes de Recommandation (SRs).** Les solutions proposées consistent généralement à analyser le comportement des utilisateurs sur le site pour en déduire leurs préférences et leurs centres d'intérêt, puis à recommander des ressources susceptibles de répondre à leurs besoins, évitant ainsi le fardeau de longues recherches sur le Web.

2. Contexte du projet et Problématique

Traditionnellement, les recommandations utilisaient un modèle unique basé sur les actions passées des utilisateurs. Cependant, ces dernières années de nouveaux types de recommandataires ont été apparus appelés les *Systèmes de Recommandation Conversationnels* (SRC), qui permettent d'obtenir progressivement les préférences de l'utilisateur en effectuant un dialogue utilisateur-système, ce qui aide à considérer ses préférences en temps réel. Par exemple, les systèmes peuvent demander quelle est la préférence de l'utilisateur pour une caractéristique associée aux items. Dans de tels systèmes, il est important de concevoir une conversation efficace, qui minimise le nombre de questions posées tout en maximisant les informations de préférence obtenues.

Le dialogue entre l'utilisateur et le système nécessite automatiquement un traitement du langage naturel (Natural Language Processing NLP) pour analyser et traiter les commentaires fournis par l'utilisateur afin d'en éliciter ses préférences et pouvoir lui générer des recommandations pertinente.

Toutefois, l'élicitation des préférences est un défi majeur pour les Systèmes de Recommandation Conversationnels (SRC), qui trouve sa solution dans l'application des techniques du Traitement Automatique du Langage Naturel (TALN), car il est difficile de déduire la bonne préférence envers un produit à partir d'un commentaire écrit.

Dans ce travail, on vise l'utilisation du Modèle de langage BERT (Bidirectional Encoder representations from Transformers) [4] pour le traitement efficace des commentaires de l'utilisateur afin d'en extraire ses préférences.

3. Motivations

Le Traitement automatique du Langage Naturel (**Natural Language Processing NLP**) est un sous domaine de l'apprentissage automatique qui a pour objectif de donner à des programmes informatiques la capacité de comprendre et traiter le langage humain.

En octobre 2018, la filière d'intelligence artificielle de Google (Google AI) a publié un modèle de Deep Learning pré-entraîné, appelé **BERT** (Bidirectional Encoder Representations from Transformers) [W10], capable de résoudre plusieurs problématiques de NLP et qui offre les meilleurs résultats actuellement. Google a même annoncé que ce modèle dépasse les performances humaines sur certaines tâches, ce qui est une grande étape pour tout algorithme de l'apprentissage automatique.

4. Objectifs

Dans ce travail on vise l'utilisation des techniques du traitement automatique du langage naturel (NLP), en se basant sur le modèle BERT, dans le domaine de la recommandation, plus précisément la recommandation conversationnelle, afin d'aider l'utilisateur à surmonter le problème de choix et d'aide à la décision pour des recommandations automatiques. En plus de :

- L'utilisation et l'application des techniques de l'NLP dans le domaine de la recommandation conversationnelle pour l'analyse et l'extraction des préférences utilisateur.
- Application de plusieurs Modèles BERT dans les Systèmes de Recommandation Conversationnels.
- Comparaison des résultats avec différents jeux de données.

L'objectif de notre travail est, donc, d'utiliser une nouvelle méthode pour l'élicitation et l'extraction de préférences de l'utilisateur, en utilisant l'analyse de sentiments comme méthode afin de comprendre les choix et réactions des utilisateurs envers un produit afin de développer un système de recommandation conversationnel adapté aux besoins des utilisateurs.

Ce système permettra de récupérer les opinions et les critiques des utilisateurs laissés à propos d'un produit, en prenant leurs avis sous forme quantitative ou alors sous forme textuelle. L'analyse des sentiments sera utilisée pour déduire la note correspondante à une opinion donnée sous forme textuelle, qui sera utilisée, prochainement, pour personnaliser les recommandations en fonction des préférences des utilisateurs.

5. Contenu du mémoire

Ce mémoire est organisé en trois chapitres :

Ce mémoire est organisé en trois chapitres :

- Dans **le premier chapitre**, nous présentons un état de l'art sur les Systèmes de Recommandations Conversationnels, à travers une illustration et présentation des différentes approches, méthodes et techniques de l'élicitation des préférences dans les SRC. De plus, les principaux concepts du modèle Bert de Google sont présentés afin de comprendre son fonctionnement.
- **Le deuxième chapitre** présente la conception et les différentes architectures de nos modèles ainsi que l'explication de l'algorithme en détail.
- **Le troisième chapitre**, présente l'implémentation de notre système proposé et les différentes expériences menées sur les différents modèles avec leurs analyses et résultats obtenus.

Chapitre 1 : Etat de L'art

1.1 Introduction

Les systèmes de recommandation sont parmi les réussites les plus visibles de l'IA dans la pratique. Typiquement, la tâche principale de tels systèmes est de diriger les utilisateurs vers des éléments d'intérêt potentiels, par exemple, dans le contexte d'un site de commerce électronique. De ce fait, ils aident non seulement les utilisateurs dans des situations de surcharge d'informations, mais ils peuvent également contribuer de manière significative au succès commercial des prestataires de services.

En règle générale, le système sous-jacent surveille le comportement de ses utilisateurs au fil du temps, puis présente un ensemble personnalisé de recommandations dans des situations de navigation prédéfinies, par exemple lorsqu'un utilisateur se connecte au service.

Bien qu'une telle approche soit courante et utile dans divers domaines, elle peut présenter un certain nombre de limitations potentielles. Il existe, par exemple, un certain nombre de scénarios d'applications, dans lesquels les préférences de l'utilisateur ne peuvent pas être estimées de manière fiable à partir de leurs interactions passées. De plus, ce qu'il faut inclure dans l'ensemble de recommandations peut dépendre fortement du contexte, et il peut être difficile de déterminer automatiquement la situation ou les besoins actuels de l'utilisateur. Les utilisateurs peuvent également construire leurs préférences uniquement pendant le processus de décision, lorsqu'ils prennent conscience de l'espace des options. Dans certains cas, ils peuvent également se renseigner sur le domaine et les options disponibles uniquement lors de l'interaction avec le recommandeur.

Les Systèmes de Recommandation sont des applications logicielles qui aident les utilisateurs à trouver des éléments d'intérêt dans des situations de surcharge d'informations. La recherche actuelle suppose souvent un paradigme d'interaction unique, où les préférences des utilisateurs sont estimées sur la base du comportement observé dans le passé et où la présentation d'une liste classée de suggestions est la principale forme unidirectionnelle d'interaction avec l'utilisateur.

Les systèmes de Recommandation Conversationnels (SRC) adoptent une approche différente et prennent en charge un ensemble plus riche d'interactions. Ces interactions peuvent, par exemple,

contribuer à améliorer le processus d'élicitation des préférences ou permettre à l'utilisateur de poser des questions sur les recommandations et de donner son avis.

Dans ce chapitre, nous présentons en détails les SRCs, leur définition et les approches utilisées, où on a visé celles basées sur l'élicitation des préférences avec un état de l'art détaillé et un tableau comparatif à la fin. Ensuite, nous présentons BERT et nous expliquons son principe de fonctionnement.

1.2 Les Systèmes de Recommandation Conversationnels

Un système de recommandation conversationnel peut être définit comme un système qui peut susciter les préférences dynamiques des utilisateurs et prendre des mesures en fonction de leurs besoins actuels grâce à des interactions multi-tours en temps réel. Cette définition met en évidence une propriété des SRC : *interactions multi-tours*.

Par une définition étroite, la conversation signifie des dialogues à plusieurs tours sous la forme d'un langage naturel écrit ou parlé. Les SRCs peuvent être vus initialement comme une réponse donnée aux utilisateurs ayant des difficultés à prendre une décision dans le cadre d'utilisation d'un système de recherche d'information classique.

L'intérêt pour les SRC a, considérablement, augmenté au cours des dernières années.[5] Cette évolution est principalement due aux progrès significatifs dans le domaine du traitement automatique du langage naturel, l'émergence de nouveaux assistants à commande vocale utilisés à domicile et l'utilisation accrue de la technologie des chatbots[1].

L'idée générale de tels systèmes, est qu'ils prennent en charge un dialogue multi-tours orienté tâche avec leurs utilisateurs. Au cours d'un tel dialogue, le système peut obtenir les préférences détaillées et actuelles de l'utilisateur, fournir des explications pour les suggestions d'articles ou traiter les commentaires des utilisateurs sur les suggestions faites. Compte tenu du potentiel important de tels systèmes, la recherche sur les SRC a déjà une certaine tradition. A la fin des années 1970, *Rich* a imaginé un agent de bibliothèque informatisé qui propose des suggestions de lecture aux utilisateurs en leur posant des questions de manière interactive, en langage naturel, sur leur personnalité et leurs préférences.

[1] **Chatbot** mot anglais de chat,bavardage , et bot , robot
Programme informatique basé sur l'intelligence artificielle, capable de répondre en temps réel aux questions d'un internaute, faisant ainsi office de conseiller virtuel recommandation officiel : agent conversationnel (d'après l'Arousse)

Outre, des interfaces basées sur le traitement du langage naturel (NLP), une variété d'interfaces utilisateur basées sur des formulaires ont été proposées au fil des années. L'une des premières approches d'interaction dans les SRC basés sur de telles interfaces est appelée « *Critiquing* »[5], qui a été proposé comme moyen de reformulation des requêtes dans le domaine des bases de données en 1982. Dans ces approches, les utilisateurs reçoivent une recommandation bientôt dans le dialogue et peuvent appliquez des critiques prédéfinies sur les recommandations, [5]**.**

Les approches basées sur des formulaires peuvent généralement être attrayantes car les actions disponibles pour les utilisateurs sont prédéfinies et non ambiguës. Cependant, de tels dialogues peuvent également sembler non naturels et les utilisateurs peuvent se sentir limités dans la manière dont ils peuvent exprimer leurs préférences. Les approches basées sur NLP, en revanche, ont longtemps souffert des limitations existantes, par exemple dans le contexte du traitement des commandes vocales. Cependant, ces dernières années, des avancées majeures ont été réalisées dans le domaine de la technologie linguistique. En conséquence, nous sommes aujourd'hui habitués à émettre des commandes vocales vers nos smartphones et assistants numériques à domicile, et ces appareils ont atteint un niveau de précision de reconnaissance impressionnant.

Parallèlement à ces évolutions dans le domaine des assistants vocaux, une adoption rapide de la technologie des chatbot a été observée ces dernières années [5]. Les chatbots, à la fois assez simples et plus sophistiqués, sont généralement capables de traiter le langage naturel et sont aujourd'hui largement utilisés dans divers domaines d'application, par exemple pour traiter les demandes de service client.

1.2.1 Définitions

- **Définition 1 :** Un Système de Recommandation Conversationnel est un système logiciel qui aide ses utilisateurs à atteindre les objectifs liés aux recommandations grâce à un dialogue multi-tours.
- **Définition 2 :** Un système de recommandation qui peut susciter les préférences dynamiques des utilisateurs et prendre des mesures en fonction de leurs besoins actuels grâce à des interactions multi-tours en temps réel. [6]

Une Interactions multi-tours, par une définition étroite, la *conversation* qui signifie des dialogues à plusieurs tours sous la forme d'un langage naturel écrit ou parlé ; dans une perspective plus

large, la conversation désigne toute forme d'interactions entre les utilisateurs et les systèmes, y compris le langage naturel écrit ou parlé, les champs de formulaire, les boutons et même les gestes [7].

Les systèmes de recommandation traditionnels fonctionnent de manière statique, estimant les préférences des utilisateurs sur les items en se basant sur l'historique des interactions passées. Cela a une limitation intrinsèque fondamentale: le système n'est pas capable de capturer les préférences actuelles d'un utilisateur, qui peuvent s'être éloignées de leurs préférences historiques.

Dans la recommandation conversationnelle, le processus devient interactif, à travers lequel les utilisateurs conversent avec le recommandeur en utilisant un langage naturel. De cette manière, un SRC devrait être capable d'obtenir ou de déduire les préférences dynamiques des utilisateurs à partir de leurs énoncés, qui peuvent consister en leurs descriptions directes ou indirectes de leurs besoins ou des réponses aux questions posées par le système de recommandation dans un contexte bien précis. Ces informations peuvent aider le système à ajuster efficacement sa stratégie de recommandation et à améliorer ses performances.

Le développement rapide des systèmes de recommandation et des agents de dialogue, ces dernières années, est l'occasion d'aider à définir comment ces deux technologies peuvent être combinées. La recommandation conversationnelle intègre de nombreuses concepts et technologies, toutes traitant de difficultés techniques, de sorte que le domaine a différentes approches pour les résoudre. Certains d'entre eux tirent parti des approches de bandit, pour équilibrer les compromis d'exploration et d'exploitation pour les utilisateurs de démarrage à froid dans des scénarios de recommandation conversationnelle [8].

1.2.2 Caractéristiques des Systèmes de Recommandation Conversationnels

- Une caractéristique fondamentale des SRC est leur orientation vers les tâches, c'est-à-dire qu'ils prennent en charge des tâches et des objectifs spécifiques aux recommandations. La tâche principale du système est de fournir des recommandations aux utilisateurs, dans le but de soutenir le processus de prise de décision de leurs utilisateurs ou de les aider à trouver des informations pertinentes.

  Les tâches supplémentaires des SRC comprennent l'acquisition des préférences de l'utilisateur ou la fourniture d'explications. Cette orientation de tâche spécifique distingue le SRC des

autres systèmes basés sur le dialogue, tels que le premier système ELIZA[2][9] ou similaire des systèmes de chatbot [10].

- La deuxième caractéristique principale d'un SRC selon la définition d'un SRC, est qu'il y a une interaction conversationnelle multi-tours. Cela contraste avec les systèmes qui prennent simplement en charge la réponse aux questions (outils Q&A).

  Fournir des recommandations de type Q&R en une seule fois est une caractéristique courante des assistants numériques personnels tels que *Siri* d'Apple et des produits similaires. Alors que ces systèmes peuvent déjà aujourd'hui répondre de manière fiable aux demandes de recommandation, par exemple pour un restaurant, ils rencontrent souvent des difficultés à maintenir une conversation à plusieurs tours. Un SRC met donc en œuvre explicitement ou implicitement une certaine forme de gestion de l'état du dialogue pour garder une trace de l'historique de la conversation et de l'état actuel.
- Les SRC peuvent être contrôlés par la voix, accepter du texte tapé ou obtenir leurs entrées via des champs de formulaire, des boutons ou même des gestes. De même, la sortie n'est pas contrainte et peut être de la voix, de la parole, du texte ou du contenu multimédia. Aucune hypothèse n'est également formulée quant à l'identité de la personne qui anime le dialogue.
- Généralement, la recommandation conversationnelle partage un certain nombre de similitudes avec la recherche conversationnelle [11]. En termes de tâches sous-jacentes, la recherche et la recommandation ont en commun qu'une tâche principale, est de classer les objets en fonction de leur pertinence supposée, soit pour une requête donnée (recherche) soit pour les préférences de l'utilisateur (recommandation).

  De plus, en termes de partie conversationnelle, les deux types de systèmes doivent interpréter les énoncés de l'utilisateur et lever l'ambiguïté des intentions de l'utilisateur au cas où les interactions en langage naturel seraient prises en charge. Dans les systèmes de recherche conversationnelle, l'hypothèse est souvent que l'interaction est basée sur « une forme écrite ou parlée » [11], alors que dans les SRC, divers types de modalités d'entrée sont possibles.

1.2.3 Architecture conceptuelle d'un SRC

Diverses approches techniques pour la construction des SRC ont été proposées au cours des deux dernières décennies. La spécificité de l'architecture technique de ces solutions dépend de la fonctionnalité du système, c'est-à-dire si la saisie vocale est prise en charge ou non., les composants de ces architectures peuvent être identifiés, comme le montre la Figure 1.1.

[2]**ELIZA** est un premier programme informatique de traitement du langage naturel créé de 1964 à 1966 au laboratoire d'intelligence artificielle du MIT par Joseph Weizenbaum .

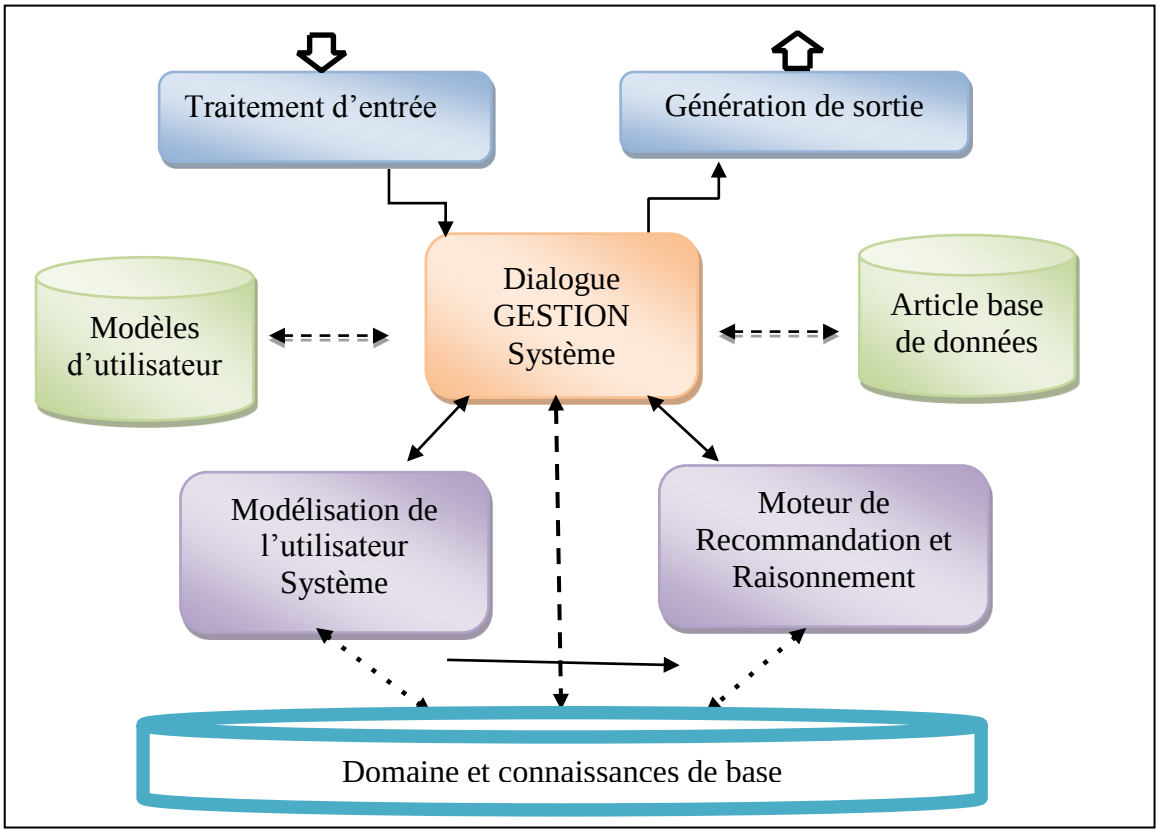


Figure 1.1 : Architecture typique d'un Système de Recommandation Conversationnel [5]

❖ ***Éléments de calcul :*** Une partie centrale d'une telle architecture est généralement un *Système de gestion de dialogue*. Ce composant pilote le flux de processus. Il reçoit les entrées traitées, par exemple les intentions, entités et préférences reconnues, et met à jour en conséquence l'état du dialogue et le modèle d'utilisateur. Après cela, à l'aide d'un moteur de recommandation et de raisonnement et de connaissances de base, il détermine l'action suivante et renvoie le contenu approprié comme une liste de recommandation.

❖ ***Le Système de modélisation utilisateur* :** peut-être une composante à part entière, en particulier lorsqu'il y a ou non des préférences d'utilisateur à long terme à prendre en compte. Dans certains cas, le profil de préférence actuel fait implicitement partie du système de dialogue. Le Moteur de recommandation et de raisonnement est chargé de récupérer un ensemble de recommandations, compte tenu de l'état actuel du dialogue et du modèle de préférence. Ce composant peut également implémenter d'autres fonctionnalités de raisonnement complexes. Outre, ces composants centraux, les architectures des SRC typiques comprennent des modules de traitement d'entrée et de sortie. Ceux-ci peuvent, par exemple, inclure la conversion parole-texte et la génération de parole. Du côté de la saisie, en particulier dans le cas de la saisie en langage naturel, des tâches supplémentaires sont généralement prises en charge, notamment détection

d'intention et reconnaissance d'entité [12], pour identifier les intentions et les entités des utilisateurs (par exemple, les attributs de l'item) dans leurs énoncés.

❖ ***Éléments de connaissance :*** Divers types de connaissances sont utilisés dans le SRC :

- La base de données d'articles est quelque chose qui est présente dans presque toutes les solutions, représentant l'ensemble des éléments recommandables, incluant parfois des détails sur leurs attributs.

- En plus de cela, différents types de domaine et connaissances de base sont souvent exploités par les SRC. De nombreuses approches codent explicitement savoir dialoguer de différentes manières, par exemple sous la forme d'états de dialogue prédéfinis, prise en charge d'intentions d'utilisateur, et des transitions possibles entre les états [35].

Ces connaissances peuvent être générales ou spécifiques à un domaine particulier. Les connaissances peuvent, en outre, être soit encodées par les concepteurs du système, soit automatiquement apprises à partir d'autres sources ou d'interactions précédentes. Un exemple typique d'approches d'apprentissage est celui qui utilise l'apprentissage automatique pour construire des modèles statistiques à partir de corpus de dialogues enregistrés.

En règle générale, les connaissances du domaine et de base peuvent être utilisées par tous les éléments de calcul. Le traitement des entrées peut nécessiter des informations sur les entités à reconnaître ou des connaissances sur les intentions prédéfinies. Le composant de modélisation utilisateur peut être construit sur des poids d'intérêt estimés concernant certaines caractéristiques des articles.

1.3 Les Principaux Défis des SRCs

Il existe de nombreux défis avec lesquels souffrent les SRCs, nous résumons dans ce manuscrit les cinq principaux défis comme suit :

1.3.1 Élicitation des préférences des utilisateurs basée sur des questions :

Les SRC offrent la possibilité d'obtenir explicitement la préférence des utilisateurs en posant des questions. Il faut répondre à deux questions importantes :

(1) Que demander ?

(2) Comment ajuster les recommandations en fonction de la réponse des utilisateurs?

Le premier se concentre sur la construction de questions pour obtenir autant d'informations que possible ; alors que le deuxième exploite les informations dans la réponse des utilisateurs pour faire des recommandations plus appropriées.

1.3.2 Stratégies de recommandation conversationnelles multi-tours :

Le système doit interagir de manière répétée avec un utilisateur et s'adapte dynamiquement à la réponse de l'utilisateur en plusieurs tours. Une stratégie efficace concerne quand poser des questions et quand faire des recommandations, i. e., laissez le modèle choisir entre :

(1) Continuer à poser des questions afin de réduire d'avantage l'incertitude des préférences, et

(2) Générer une recommandation basée sur l'estimation de la préférence actuelle de l'utilisateur.

En règle générale, le système doit viser une recommandation réussie en utilisant le moins de tours, car les utilisateurs perdront patience après trop de tours. De plus, certaines stratégies conversationnelles sophistiquées tentent de mener des dialogues de manière proactive qui peut introduire divers sujets et tâches dans les SRC

1.3.3 Compréhension et Génération du Langage Naturel :

Communiquer comme un être humain continue d'être l'un des défis les plus difficiles dans les SRC. Pour comprendre les intérêts et les intentions des utilisateurs, certaines méthodes des SRC définissent l'entrée du modèle sous forme de balises prédéfinies qui capturent les informations sémantiques et les préférences de l'utilisateur [14].

Certaines méthodes extraient les informations sémantiques des énoncés bruts des utilisateurs via des techniques de remplissage de slots et représentent les intentions des utilisateurs dans des paires slot-valeur [15].

Pour générer des réponses compréhensibles par l'homme, les SRC utilisent de nombreuses stratégies telles que la fourniture directe d'une liste de recommandations [16], incorporant des éléments recommandés dans un modèle de langage naturel basé sur des règles [17]. De plus, certains chercheurs proposent le cadre de bout en bout pour permettre aux SRC de comprendre

avec précision les sentiments et les intentions des utilisateurs à partir du langage naturel brut et de générer des réponses en langage naturel lisibles, fluides, cohérentes et significatives [18].

1.3.4 Compromis entre l'exploration et l'exploitation (E&E) :

L'un des problèmes des systèmes de recommandation est que chaque utilisateur ne peut interagir qu'avec quelques éléments de l'ensemble de données. Un grand nombre d'éléments susceptibles d'intéresser un utilisateur resteront invisibles pour l'utilisateur. Pour les utilisateurs de démarrage à froid (qui viennent de rejoindre le système et n'ont aucune ou très peu d'interactions), le problème est particulièrement grave.

Grâce à la nature interactive, les SRC peuvent explorer activement les éléments invisibles pour mieux capturer les préférences de l'utilisateur. De cette façon, les utilisateurs peuvent bénéficier d'avoir des chances d'exprimer leurs intentions et d'obtenir des recommandations mieux personnalisées. Cependant, le processus d'exploration a un prix. Comme les utilisateurs n'ont que peu de temps et d'énergie pour interagir avec le système, une exploration ratée fera perdre du temps et l'opportunité de faire des recommandations précises. De plus, l'exposition des éléments sans rapport nuit aux préférences des utilisateurs, [19]. Par conséquent, la recherche de compromis E&E est un problème critique dans les SRC.

1.3.5 Évaluation et Simulation utilisateur :

L'évaluation est un sujet important. Contrairement aux modèles de recommandation statiques optimisés sur les données hors ligne, les SRC mettent l'accent sur l'expérience utilisateur lors des interactions dynamiques. Par conséquent, nous ne devons pas seulement considérer l'évaluation au niveau du tour pour la génération de recommandations et de réponses, mais également prêter attention à l'évaluation au niveau de la conversation. En outre, l'évaluation des SRC nécessite un grand nombre d'interactions utilisateur en ligne, qui sont coûteuses à obtenir [20] Les solutions pratiques incluent :

(1) tirer parti de l'évaluation hors stratégie qui évalue la stratégie cible à l'aide des données enregistrées dans le cadre de la stratégie de comportement [21] et

(2) Introduire directement des simulateurs des utilisateurs pour remplacer les vrais utilisateurs dans l'évaluation [22].

A partir de ces défis, on va se baser, dans ce travail, sur l'**Élicitation des préférences des utilisateurs** qui constitue le défi le plus important et le plus étudié à travers la littérature, et qui caractérise exceptionnellement ce genre de recommandeurs. Dans ce qui suit, nous présentons en détails la notion d'élicitation des préférences dans les SRC avec un état de l'art détaillé.

1.4 L'élicitation des Préférences

L'élicitation des préférences est une procédure d'interaction entre le système d'aide à la décision et le décideur. Elle vise à récolter les informations préférentielles les plus utiles possibles afin de guider la recherche vers une solution pouvant correspondre au mieux aux préférences du décideur [64].

La qualité de la solution recommandée par le système dépend fortement de l'efficacité de la procédure d'élicitation. Cette efficacité est souvent mesurée par le nombre de questions qu'elle requiert pour la détermination d'une bonne recommandation, mais également par la pertinence des questions posées (d'un point de vue du caractère informatif), par l'effort cognitif qu'elle impose au décideur, et par le temps et la complexité de traitement des informations acquises. La difficulté de la conception d'une méthode efficace réside dans le fait que la simplicité cognitive ne va pas toujours avec l'efficacité informative, notamment lorsque les préférences du décideur sont complexes.

1.4.1 Les méthodes d'élicitation de préférence dans les SRCs :

Traditionnellement, les recommandations étaient basées sur un modèle unique basé sur l'évaluation des utilisateurs ou l'historique des achats. Cependant, les préférences de l'utilisateur peuvent changer de temps à autre en fonction de facteurs internes et externes. Par conséquent, il peut être utile de demander directement aux utilisateurs plus d'informations. On sait que les utilisateurs pourraient être disposés à donner plus de précisions sur leur préférence. Les recommandations conversationnelles permettent une interaction itérative entre l'utilisateur et le système pour clarifier les préférences. Deux approches majeures émergent [23] : *la question basée sur les items et la question basée sur les caractéristiques*.

- ***L'approche basée sur les items :*** demande aux utilisateurs d'exprimer leur préférence par rapport aux items. D'un autre côté,
- ***L'approche basée sur les caractéristiques :*** basé sur les préférences de l'utilisateur sur et les caractéristiques (ou facettes, attributs).

Les deux approches présentent des problèmes clés similaires :

- *Quelle question doit être posée ?* et
- *Comment intégrer les commentaires des utilisateurs pour améliorer la précision des recommandations* ?

Il est essentiel que les recommandataires conversationnels obtiennent efficacement les commentaires des utilisateurs, ce qui signifie éviter de poser trop de questions et s'assurer que les questions sont posées dans un ordre raisonnable.

L'emplacement est rempli d'une caractéristique d'un ensemble de caractéristiques associées aux items, sous la forme d'une question. Par example:

• *Utilisateur : Je veux aller au restaurant.*
• *Système : Quelle <feature = "cuisine"> aimez-vous ?*
• *Utilisateur : J'aime <value = "Asian">.*
• *Système : Voici quelques recommandations pour vous*.

Dans un tel contexte, le système doit hiérarchiser une caractéristique à demander (parmi toutes les caractéristiques disponibles), ce qui lui permettra de restreindre efficacement les candidats à la recommandation.

De plus, étant donné une question à un moment donné, les utilisateurs peuvent ne pas toujours avoir une préférence spécifique. Ainsi, il peut être utile de faire des suggestions sur les valeurs de caractéristiques susceptibles d'intéresser les utilisateurs. L'hypothèse est que de telles suggestions peuvent réduire l'effort cognitif des utilisateurs pour donner leur avis.

Nous définissons dans ce qui suit les deux approches, en citant quelques travaux de l'état de l'art.

1.4.1.1 Approches basées sur les items

Les premières études demandent directement aux utilisateurs des opinions sur un élément lui-même [24]. Contrairement aux SRs traditionnels qui doivent estimer les préférences de l'utilisateur à l'avance, les SRC peuvent construire et modifier le profil de l'utilisateur pendant le processus d'interaction. Dans les modèles de SRs traditionnels, les items recommandés sont produits de manière relativement stable à partir de tous les candidats.

Dans le scénario SRC, les éléments recommandés doivent être mis à jour après que le système a reçu les commentaires d'un utilisateur et il pourrait s'agir d'un changement complet afin de s'adapter aux préférences en temps réel de l'utilisateur. Par conséquent, au lieu de mettre à jour simplement les paramètres des modèles en ligne, certaines règles ou mécanismes explicites sont nécessaires.

Nous introduisons trois méthodes qui peuvent susciter l'attitude des utilisateurs envers les items et peuvent rapidement ajuster les recommandations.

a) **Méthodes basées sur le choix :** L'idée principale de l'élicitation des préférences basée sur le choix est de laisser les utilisateurs choisir de manière récurrente leurs éléments ou ensembles d'éléments préférés parmi les options actuellement proposées. Les stratégies courantes incluent :

(1) Le choix d'un élément parmi deux options données [25]

(2) Sélection d'un élément dans une liste d'éléments donnés [26] et

(3) Le choix d'un ensemble d'éléments dans deux listes données [27]

Une fois que l'utilisateur a choisi les éléments préférés, les méthodes modifient les recommandations en fonction du choix de l'utilisateur. Par exemple, Loepp et al. ont utilisé le modèle de Factorisation Matricielle (FM) [13] pour initialiser les vecteurs d'intégration des utilisateurs et des éléments, puis sélectionnez deux ensembles d'éléments dans l'espace d'intégration d'éléments en tant qu'ensembles candidats et laissez un utilisateur choisir l'un des deux ensembles. Il est important de s'assurer que les deux ensembles candidats sont aussi différents ou distinguables que possible. Pour y parvenir, les auteurs adoptent un algorithme de FM, qui factorise la matrice d'interaction utilisateur-item et obtient les vecteurs de un à un par ordre décroissant. Par conséquent, les facteurs, c'est-à-dire les différentes dimensions des vecteurs d'intégration, sont classés par caractère distinctif. Ensuite, les auteurs sélectionnent de manière itérative deux ensembles d'éléments avec une seule valeur de facteur variant.

Par exemple, si deux facteurs représentent le degré des films de 'Humour' et 'Action', respectivement, alors les deux ensembles candidats sont un ensemble de films avec un degré élevé de 'Humour' et un autre avec un faible degré de 'Humour', tandis que le degré d''Action' des deux ensembles est fixé au niveau moyen. Lorsqu'un utilisateur choisit un ensemble d'éléments, le vecteur d'intégration de préférences de l'utilisateur est défini sur la moyenne des vecteurs d'intégration des éléments choisis. Le choix devient plus difficile à mesurer que le processus d'interaction se poursuit. Les utilisateurs peuvent choisir d'ignorer la question, ce qui signifie qu'ils ne peuvent pas faire la différence entre les deux ensembles d'éléments ou qu'ils ne s'en soucient pas.

b) Élicitation de préférence bayésienne :

De plus, il existe des études basées sur une vision probabiliste de l'élicitation des préférences, qui a fait l'objet de recherches de longue date [28]. Fondamentalement, il existe une fonction

utilitaire ou une fonction score $u(x_j,u_i)$ représentant la préférence de l'utilisateur i pour l'élément j. Habituellement, il peut être écrit comme une fonction linéaire comme

$$u(x_j, u_i) = x_j^T u_i \quad (1)$$

Dans un cadre bayésien, pour un utilisateur *i*, la préférence est modélisée par une distribution probabiliste au lieu d'un vecteur déterministe, ce qui signifie que le vecteur u_i est échantillonné à partir d'une croyance antérieure d'un utilisateur$P(U^{(i)})$. Par conséquent, l'utilitaire d'un article *j* pour un utilisateur 'i' est calculée comme l'espérance :

$$\mathbb{E}[u(x_j, u_i)] = \int_{ui\sim u(i)} P(u_i)u(x_j, u_i)du_i \quad (2)$$

L'élément avec l'utilité maximale attendue pour l'utilisateur *i* est considéré comme les éléments de recommandation :

$$argmax_j E[u(x_{j,}u_i)] \quad (3)$$

Sur la base de la fonction utilitaire, le système peut sélectionner certains éléments à interroger et la distribution des croyances des utilisateurs peut être mise à jour en fonction des commentaires des utilisateurs. Plus précisément, étant donné une réponse de l'utilisateur r_j À la question q, la croyance postérieure de l'utilisateur $p(q, r_i)$ peut s'écrire :

$$p(u_i|q, r_i) = \frac{P(u_i|q,r_i)p(u_i)}{\int_{u_i} P(r_i|q,u_i)P(u_i)du_i} \quad (4)$$

En ce qui concerne la stratégie de requête, c'est-à-dire la sélection des éléments à demander, il existe différents critères. Par example, [29] proposent un cadre de processus décisionnel de Markov partiellement observé (POMDP) comme stratégie de requête séquentielle. [30] utilisent le paradigme de la valeur attendue de l'information comme stratégie relativement (**EVOI : expected value of information**) pour sélectionner les éléments à interroger. De plus, le type de requête peut être classé en deux types différents :

(1) une requête de comparaison par paire : dans laquelle les utilisateurs sont tenus de choisissez ce qu'ils préfèrent entre deux articles ou deux ensembles d'articles [31] ou

(2) une requête aussi tard, où les utilisateurs doivent choisir parmi plusieurs options données [32].

c) Recommandation interactive :

Les modèles sont principalement basés sur l'apprentissage par renforcement. Certains chercheurs adoptent un algorithme de bandit multi-bras (MAB)[3][33] .L'avantage est double. Premièrement, les algorithmes MAB sont efficaces et prennent naturellement en charge les scénarios conversationnels. Deuxièmement, les algorithmes MAB peuvent exploiter les éléments que les utilisateurs ont aimés auparavant et explorer les éléments que les utilisateurs peuvent aimer mais n'ont jamais essayés auparavant. Il existe également des chercheurs qui formulent la recommandation interactive comme un problème de méta-apprentissage qui peut s'adapter rapidement à de nouvelles tâches [34]**.**Une tâche ici consiste à faire des recommandations basées sur plusieurs historiques de conversation. Les méthodes de méta-apprentissage et les méthodes basées sur le MAB ont la capacité d'équilibrer l'exploration et l'exploitation. Récemment, les chercheurs ont incorporé des modèles d'apprentissage par renforcement profond (DRL)[4] dans des systèmes de recommandation conversationnel [35]**.** Contrairement aux méthodes basées sur le MAB qui supposent généralement que la préférence de l'utilisateur est inchangée pendant l'interaction, les méthodes basées sur le DRL peuvent modéliser une préférence dynamique et une utilité à long terme [36]**.**

Les Frameworks sans modèle tels que Deep Q-network (DQN) [37] et le gradient de politique déterministe profond (DDPG) [38] sont utilisés dans les scénarios de recommandation interactifs. La plupart des méthodes basées sur l'apprentissage par renforcement (Reinforcement Learning RL) souffrent souvent de problèmes d'efficacité et ne peuvent pas gérer les utilisateurs à démarrage à froid. [39] proposent d'intégrer un graphe de connaissances dans la recommandation interactive pour résoudre ces problèmes. Pour plus de travaux qui tirent parti du RL dans les SRC, vous pouvez consulter [40].

Cependant, exiger directement des éléments est inefficace pour créer le profil utilisateur car l'ensemble d'éléments candidats est volumineux. Dans les applications SRC du monde réel, les utilisateurs s'ennuieront à mesure que le nombre de conversations qui tourne augmente. Il est plus pratique de poser des questions centrées sur les attributs, c'est-à-dire de demander aux

[3] **MAB** : est un problème classique qui illustre bien le dilemme E&E (Katehakis et Veinott, 1987; Auer et al., 2002). Le nom vient de l'histoire où un joueur à une rangée de machines à sous (dont chacune est connue sous le nom de « bandit à un bras ») veut maximiser son gain attendu et doit décider quelles machines jouer, combien de fois jouer chaque machine, dans quel ordre les jouer, et s'il faut continuer avec le machine actuelle ou essayez une autre machine. Le problème est difficile car toutes les machines à sous sont des boîtes noires, dont les propriétés, c'est-à-dire la probabilité de gagner, ne peuvent être estimées que par les récompenses observées dans les expériences précédentes. T .

[4] **DRL** :(L'apprentissage par renforcement profond est la combinaison de l'apprentissage par renforcement (RL) et de l'apprentissage en profondeur. Ce domaine de recherche a été en mesure de résoudre un large éventail de tâches décisionnelles complexes qui étaient auparavant hors de portée d'une machine. Ainsi, Deep RL ouvre de nombreuses nouvelles applications dans des domaines tels que la santé, la robotique, les réseaux intelligents, la finance et bien d'autres)

utilisateurs s'ils aiment un attribut (ou un sujet/une catégorie dans certains ouvrages), puis de faire des recommandations basées sur ces attributs [46]. Par conséquent, l'estimation et l'utilisation des préférences d'un utilisateur envers les attributs deviennent un problème de recherche clé.

1.4.1.2 Approches basées sur les attributs :

Il est plus efficace de poser des questions sur les attributs, car le fait que les utilisateurs aiment ou n'aiment pas un attribut peut réduire considérablement le nombre de candidats à la recommandation. Le défi est de déterminer une séquence d'attributs à demander afin de minimiser l'incertitude des besoins actuels des utilisateurs [41]**.** La base de critiques susmentionnée les méthodes entrent dans cette catégorie. En outre, il existe d'autres types de méthodes, nous introduisons certaines branches traditionnelles comme ci-dessous.

c) Modèles d'ajustement à partir des interactions historiques

Une conversation peut être considérée comme une séquence d'entités comprenant des éléments consommés et des attributs mentionnés, et l'objectif est d'apprendre à prédire le prochain attribut à demander ou le prochain élément à recommander. Par conséquent, le réseau de neurones séquentiel tel que le modèle d'unité récurrente fermée (GRU) [42] et le modèle de mémoire à long court terme (LSTM) [43] peut être naturellement adopté dans ce contexte, en raison de sa capacité à capturer la dépendance à long et à court terme dans les modèles de comportement des utilisateurs.

Un travail exemplaire est le modèle de questions et recommandations (Q&R) proposé par [44]**,** où l'interaction entre le système et un utilisateur est mise en œuvre comme un système de sélection. À chaque tour, le système demande à l'utilisateur de choisir un ou plusieurs sujets distincts (par exemple, NBA, bandes dessinées ou cuisine) dans la liste donnée, puis recommande des éléments de ces sujets à l'utilisateur. Il contient un module de déclenchement pour décider s'il faut poser une question sur les attributs ou faire une recommandation.

Le mécanisme de déclenchement peut être aussi simple qu'un mécanisme aléatoire ou peut être plus sophistiqué, c'est-à-dire utilisant des critères capturant l'état de l'utilisateur, voire être initié par l'utilisateur. Au t-ième pas de temps, le sujet suivant q sur lequel l'utilisateur a cliqué peut être prédit en fonction du historique de visionnage de l'utilisateur e1, …, eT comme : P(q|e1 , …, eT). Après que l'utilisateur ait cliqué sur un sujet *q*, le modèle peut recommander un article *r* sur la base de la probabilité conditionnelle qui s'écrit : $P(r|e_1, \ldots, e_T, q)$. Les deux

probabilités conditionnelles sont implémentées en tant qu'architecture GRU [45] **.**Cet algorithme est déployé sur YouTube, pour obtenir les préférences des utilisateurs en démarrage à froid.

[46] ont proposé un paradigme « System Ask User Response » (SAUR). Pour chaque élément, ils utilisent les informations riches de révision et convertissent une phrase contenant une paire aspect-valeur en un vecteur latent via le modèle GRU. Ensuite, ils adoptent un module de mémoire avec mécanisme d'attention [47] pour effectuer à la fois la prochaine tâche de génération de question (détermination de l'attribut à demander) et la prochaine tâche de recommandation d'élément. Encore une fois, ils développent également un déclencheur heuristique pour décider s'il est temps d'afficher les Top-N items recommandés aux utilisateurs ou pour continuer à poser des questions sur les attributs.

Une limitation du travail est que les auteurs supposent que toutes les informations contenues dans les critiques peuvent soutenir le comportement d'achat, mais ce n'est pas vrai car les utilisateurs peuvent se plaindre de certains aspects des articles achetés, par exemple, un utilisateur peut écrire "*64 giga octets ne suffisent pas*". L'utilisation d'informations sans discrimination va induire le modèle en erreur et détériorer les performances.

Les énoncés produits par le système, c'est-à-dire les questions, sont construits avec des modèles de langage prédéfinis, ce qui signifie que le système doit prêter attention uniquement à l'aspect et à la valeur. Il s'agit d'un cadre courant dans les études de pointe sur les SRC, car la tâche principale ici est la recommandation au lieu de la génération de langage [48]

Notez que ces types de méthodes ont un inconvénient commun : apprendre à partir des comportements historiques des utilisateurs ne peut pas aider à comprendre la logique derrière l'interaction. En tant que systèmes interactifs, ces modèles ne considèrent pas comment réagir aux commentaires lorsque les utilisateurs rejettent la recommandation, c'est-à dire qu'ils essaient simplement d'adapter les préférences à l'interaction historique et n'envisagent pas de stratégie explicite pour gérer les différents commentaires.

a) Réduction de l'incertitude :

Contrairement aux méthodes séquentielles basées sur un réseau de neurones qui n'ont pas de stratégie explicite pour gérer toutes sortes de commentaires des utilisateurs, certaines études tentent de construire une logique simple pour affiner les éléments candidats.

- *Méthodes basées sur la critique :*

Le modèle de critique surmentionné est généralement équipé d'une tactique heuristique pour susciter la préférence de l'utilisateur sur les attributs [49]. Dans les modèles de critique

traditionnels, où la critique sur une valeur d'attribut (par exemple, « pas rouge » pour la couleur ou « moins cher » pour le prix) est utilisé pour reconstruire l'ensemble candidat en supprimant les éléments avec des attributs insatisfaits [50]**.** Les méthodes basées sur les vecteurs neuronaux prennent la critique dans le vecteur latent, qui est chargé de générer à la fois les éléments recommandés et les attributs expliqués.

Par exemple, [51] proposaient un modèle de Filtrage Collaboratif Neuronal Explicable (CE-NCF) pour la critique. Ils utilisaient le modèle NCF [52] pour encoder la préférence d'un utilisateur *i* pour un article *j* comme vecteur latent $\hat{z}_{i,j}$, et $\hat{z}_{i,j}$est utilisé pour produire le score de notation $\hat{r}_{i,j}$ ainsi que le vecteur d'attribut expliqué $\hat{s}_{i,j}$ Les attributs sont composés d'un ensemble de phrases clés telles que « *doré, cuivre, orange, noir, jaune* » et chaque dimension de $\hat{s}_{i,j}$ correspond à un certain attribut. Lorsqu'un utilisateur n'aime pas un attribut et le critique en temps réel, le système met à jour le vecteur d'attribut expliqué $\hat{s}_{i,j}$ en mettant la cote correspondante à zéro. Ensuite, le vecteur mis à jour $\tilde{s}_{i,j}$ est utilisé pour mettre à jour le vecteur latent $\hat{z}_{i,j}$ être $\tilde{z}_{i,j}$. Par conséquent, le score de recommandation $\check{r}_{i,j}$ est mis à jour. Suite à ce réglage, [53] ont changé le modèle NCF de base [60] pour être un modèle d'auto encodeur variationnel (VAE), et ce modèle génératif peut aider le système de critique à avoir une meilleure efficacité de calcul, une stabilité améliorée et une convergence plus rapide.

- *Méthodes axées sur l'apprentissage par renforcement :*

L'apprentissage par renforcement est également utilisé dans les SRC pour sélectionner les attributs appropriés à demander [17] **.**Fort d'un réseau de politiques approfondies, le système sélectionne non seulement les attributs, mais détermine également une stratégie de contrôle sur le moment où changer le sujet de la conversation en cours.

- *Candidats contraints par un graphique :*

Le graphe est une structure répandue pour représenter la relation entre différentes entités. Il est naturel d'utiliser des graphiques pour filtrer les éléments en fonction d'un ensemble d'attributs. Par exemple, [54] ont proposé un algorithme de raisonnement de chemin interactif sur un graphe hétérogène sur lequel les utilisateurs, les articles et les attributs sont représentés sous forme de nœuds et un arc qui connecte deux nœuds représente une relation entre deux nœuds, par exemple, un utilisateur a acheté un article, ou un article a une certaine valeur pour un attribut.

À l'aide du graphique, une conversation peut être convertie en un chemin sur le graphique. Les auteurs comparent l'incertitude de préférence pour les attributs et choisissent les attributs à

poser avec le maximum d'incertitude. Ici, la préférence pour un certain attribut est modélisée par la préférence moyenne pour les éléments qui ont cet attribut. Par conséquent, l'espace de recherche et la surcharge de l'algorithme peuvent être considérablement réduits en utilisant les informations du graphique.

Il existe d'autres études qui appliquent les réseaux de neurones graphiques (GNN) pour apprendre une représentation puissante des éléments et des attributs, de sorte que les informations sémantiques dans les vecteurs d'intégration appris peuvent aider les modèles SRC de bout en bout à générer des recommandations appropriées. Par exemple, le modèle GCN et ses variantes [55] sont adoptés sur le graphe de connaissances dans les modèles SRC récents [56].

1.4.2 Autres méthodes :

Il y a d'autres tentatives pour faire des recommandations basées sur les commentaires des utilisateurs sur les attributs. Par exemple, [57] ont proposé un système de recommandation basé sur des questions et sur un modèle de factorisation matricielle étendu, qui considère simplement les données d'évaluation des utilisateurs, pour combiner les commentaires en temps réel des utilisateurs.

L'hypothèse de base est que si un utilisateur aime un article, alors il aimera les attributs de cet article. Ainsi, à chaque tour, le système sélectionnera l'attribut qui comporte le maximum d'incertitude à demander. En d'autres termes, si un attribut est connu pour être partagé par la plupart des éléments qu'un utilisateur aime, il n'a pas besoin de poser de questions sur cet attribut. De même, il n'est pas nécessaire de poser des questions sur les attributs que les utilisateurs n'aiment pas. Seulement s'il n'est pas sûr qu'un utilisateur aime un attribut, alors poser des questions sur cet attribut peut fournir le plus d'informations. Les paramètres des matrices peuvent être mis à jour après que les utilisateurs ont fourni des commentaires.

En outre, en utilisant des idées similaires aux modèles sur-mentionnés basés sur des items de demande, des modèles basés sur le MAB [58] et les approches bayésiennes [59] sont également développés dans les SRC demandant des d'attributs.

Les recommandations conversationnelles encapsulent plusieurs sous-tâches d'apprentissage automatique, notamment l'élicitation des préférences, la recommandation d'articles et la génération de langage. Dans ce travail, nous nous concentrons sur l'élicitation des préférences en tant que tâche principale d'un système de recommandation conversationnel et nous étudions ces approches à partir de l'axe d'analyse de sentiment.

1.5 Intentions et apprentissage des sentiments :

Avec la croissance des médias sociaux, des blogs, des forums de discussion, des sites d'avis en ligne, etc., les grandes entreprises se sont rendu compte qu'être sensible aux sentiments peut les aider à mieux comprendre le comportement des utilisateurs, à suivre et à gérer leur présence et leur image en ligne et à utiliser ces informations.

L'analyse des sentiments aide les ordinateurs à comprendre, interpréter et utiliser les langages humains. Elle transforme le texte non structuré en données structurées à l'aide de deux types de représentation du langage :

1.5.1 Représentation linguistique sans contexte :

Chaque mot de la phrase est représenté individuellement sans égard aux autres mots de la phrase. Exemple : méthode word2vec et Gant.

1.5.2 Représentation linguistique contextuelle :

Chaque mot de la phrase est représenté par rapport aux autres mots de la phrase. Puisque la représentation contextuelle du langage prend en compte la relation de chaque mot avec d'autres mots, la directionnalité entre en jeu:

- **Unidirectionnalité** : prédit le mot suivant en fonction des mots passés. Exemple : méthode Skip-gram, sac de mots.
- **Bidirectionnalité** : prédit le mot en fonction des contextes avant et après des mots qui l'entourent.

Les réseaux de neurones sont connus pour extraire automatiquement des caractéristiques, donc ils peuvent être utilisés pour extraire les intentions et les sentiments des utilisateurs dans les CRS. Un exemple classique dans les CRS est le cadre de bout en bout proposé par [18], qui prend en entrée les énoncés bruts de l'utilisateur et produit directement les réponses dans l'interaction. Ils collectent l'ensemble de données via la plateforme de crowdsourcing Amazon Mechanical Turk (AMT) . Ils associent les travailleurs AMT et attribuent à chacun un rôle.

En plus, d'utiliser les réseaux de neurones basés sur RNN, certains SRC adoptent le modèle de réseaux de neurones convolutifs (CNN) [W8]**,** qui s'est avérée très efficace pour modéliser la sémantique à partir du langage naturel brut [W9]. Cependant, les réseaux de neurones profonds sont souvent critiqués pour être non transparents et difficiles à interpréter.

[60] ont étudié les représentations des encodeurs bidirectionnels des transformateurs (BERT) [61], une technologie puissante pour la pré-formation en NLP développée par Google, pour analyser ces paramètres peuvent capturer et stocker des informations sémantiques sur des éléments tels que des livres, des films et de la musique pour les SRC. Les informations sémantiques comprennent deux types de connaissances nécessaires pour effectuer une recherche et une recommandation conversationnelles, à savoir les connaissances basées sur le contenu et les connaissances basées sur la collaboration.

Après avoir présenté toute une panoplie de travaux de l'état de l'art sur les SRC et plus précisément sur l'élicitation de préférences, nous présentons dans la section suivante un tableau récapitulatif qui sert comme moyen de comparaison entre les différentes méthodes citées ci-dessus.

1.6 Travaux réalisés sur l'élicitation des préférences dans les SRC

Type de système	Approches	Méthode	Algorithme	Avantages	Inconvénients
Basé sur les Items (éléments) (Bell et al,2007)	Collaborative	Méthode basé sur le choix (Sepliarskaia et al, 2018),	Algorithme MF factorielle (Bell et al.,2007)	- Ajustement des recommandations de l'utilisateur d'une manière rapide	- N'utilise pas le langage naturel dans leur interaction avec l'utilisateur
	Basée sur le Contenu	Elicitation de préférence bayésienne (Vendrov et al.,2020)	Gradient Chen, Haokun, Dai, Xinyi, Han, Cai, Zhang, Weinan, Wang, Xuejian, Tang, Ruiming, Zhang, Yuzhou, Yu, Yong, 2019ba.	- Elle exploite des logiciels et du matériel ML hautement optimisés pour faciliter la mise en œuvre, les performances et la parallélisassions - Il se généralise à une variété de types de requêtes, y compris celles impliquant des éléments non présents dans l'ensemble de données, ou des éléments partiellement spécifiés - Il accepte les attributs d'élément continus - Il peut utiliser des métriques différentiables arbitraires et des utilitaires non linéaires, permettant une optimisation de bout en bout à l'aide d'une ascension de gradient. y compris ceux impliquant des éléments non présents dans l'ensemble de données, ou des éléments partiellement spécifiés	- Il est prohibitif en termes de calcul de construire des requêtes avec un EVOI maximal dans les RS avec de grands espaces d'éléments -Incapable de s'adapter à des paramètres réalistes avec des millions d'éléments.
	Apprentissage par	Recommandation	Bandis Multi-bras	- Utilise des algorithmes MAB	- Souffre souvent de problème

	renforcement profond (RL)	interactive		MAB (Zhao et al.,2013 ; Christakopoulou et al., 2016 Wang et al.,2018)	efficace - Peuvent exploités l'historique des éléments aimés par l'utilisateur et non exploités	de démarrage à froid - S'ennuie à mesuré le nombre de conversation augmente - Manque de confiance au début (score trop bas) - Poser trop de question peut nuire l'utilisateur
Basé sur les attributs	Réseaux de neurones sequentiel	Modèle d'ajustement à partir de l'interaction historique		-Architecture GRU Modèle d'unité récurrent fermée (ChoBart van Merrienboer et al., 2014)	- La capacité à capturer la dépendance a long et à court terme dans les modèles de comportement des utilisateurs	-Apprendre à partir des comportements historiques des utilisateurs ne peut pas aider à comprendre la logique derrière l'interaction.
	Modèle filtrage collaboratif neuronal	**Réduire l'incertitude**	Méthode basé sur la critique	-FCN (Il et al ., 2017) -Modèle d'Autoencodeur Variationnel (VAE) (Luo et al.,2020b)	- Avoir une meilleure efficacité de calcul - Une stabilité améliorée et une convergence plus rapide	/
			Méthode axées sur l'apprentissage par renforcement (Soleil et zhang., 2018 ; Lei et al., 2020a,2020b)	Avancées et défis dans les systèmes de recommandation conversationnels : une enquête	/	-Ils n'ont pas une stratégie explicite pour gérer toute sorte de commentaire
			Candidat contrainte par un graphique	Algorithme de raisonnement de chemin interactif	- Chemin connue et Claire - Basé sur CPR (resonnement passé conversation)	

			(Lei et al., 2020b)	(Lei et al., 2020b) GCN (Kipf et Welling 2017 ; Schlichtkrull et al., 2018)		
Analyse des sentiments	Collaboratif et Contenue	Analyse des sentiments		-Réseaux de neurones profonds (RNN) - Réseaux de neurone convolutionnel (CNN) (Ren et al., ; Liu et al., 2020ab) BERT (Devlin et al.,2019)	- Très efficace pour modéliser la sémantique à partir du langage naturel brut - Bidirectionnel - Il bien utilisé application talen - Les résultats peuvent être apprise et mémoriser - Longue séquence temporel (grande mémoire) Bien utilisé dans les applications des traitements d'image - Bidirectionnel - Système Multi couches -Des modèles préformés ou personnalisés	- Les réseaux de neurones profonds sont souvent critiqués pour leur manque de transparence et leur interprétation difficile -Mémoire petite (garde l'info un court terme) - Non explicatif -Utilise plusieurs ressources (mémoires, processeur) et utilise beaucoup de temps

Tableau 1.1 : Travaux réalisées dans l'élicitation des préférences

1.7 Le Modèle BERT

BERT (**B**idirectional **E**ncoder **R**epresentation from **T**ransformer.), c'est un modèle d'intégration de pointe publié par Google en fin 2018. Il a créé une percée majeure dans le domaine de la NLP en fournissant de meilleurs résultats dans de nombreuses tâches de la NLP, telles que la réponse aux questions, génération de texte, classification de phrases, et bien d'autres encore. L'une des principales raisons car le succès de BERT est qu'il s'agit d'un modèle d'intégration basé sur le contexte, contrairement à d'autres modèles populaires incorporer des modèles, tels que word2vec, qui sont sans contexte.

Le transformateur est l'une des architectures d'apprentissage en profondeur les plus populaires qui est principalement utilisé pour les tâches de traitement du langage naturel. Depuis l'avènement du transformateur, il a remplacé RNN et LSTM pour diverses tâches [4].

Le modèle du transformateur est entièrement basé sur le mécanisme de d'attention, Le transformateur utilise un type spécial de mécanisme d'attention appelé auto- attention.

1.7.1 Les Entrées du modèle

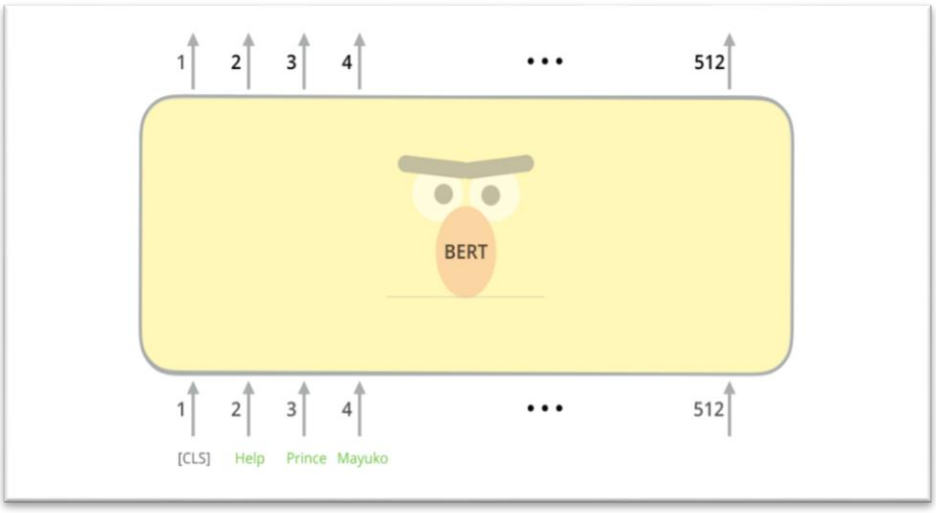


Figure 1.2 entrée du Modèle BERT [W11]

Le transformateur est constitué d'une architecture codeur-décodeur. Nous alimentons la phrase d'entrée (source phrase) à l'encodeur. L'encodeur apprend la représentation de la phrase d'entrée et envoie la représentation au décodeur. Le décodeur reçoit la représentation apprise par l'encodeur en entrée et génère la phrase de sortie (phrase cible).

Le premier jeton d'entrée est fourni avec un jeton spécial [CLS] qui signifie ici Classification.

L'encodeur, BERT [4] prend une séquence de mots en entrée qui continue de remonter la pile. Chaque couche applique l'auto-attention et transmet ses résultats à travers un réseau de rétroaction, puis les transmet à l'encodeur suivant.

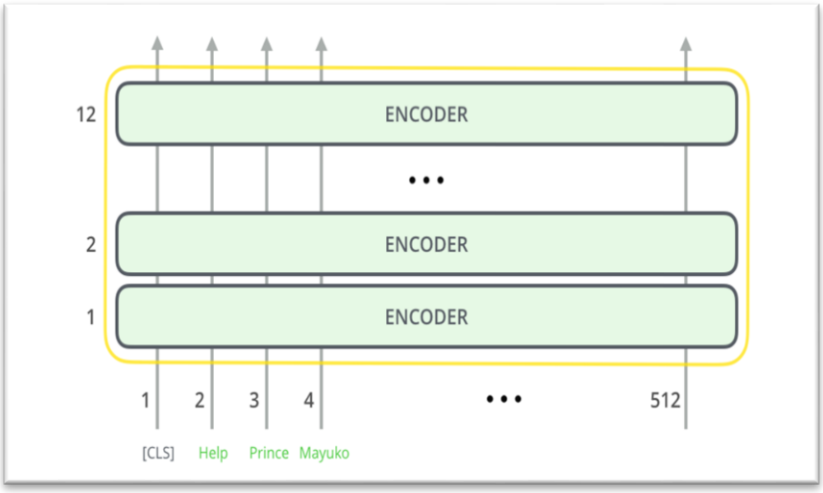


Figure 1.3 Encodeur Bert, [W11]

En termes d'architecture, cela a été identique au transformateur jusqu'à présent (à part la taille, qui n'est qu'une configuration que nous pouvons définir). C'est à la sortie que l'on commence à voir comment les choses divergent.

1.7.2 La Sortie du Modèle

Le transformateur se compose d'une pile d'un certain nombre d'encodeurs. La sortie d'un codeur est envoyé comme entrée à l'encodeur au-dessus. Le final l'encodeur renvoie la représentation de la phrase source donnée en sortie. Nous alimentons le phrase source comme entrée de l'encodeur et obtenir la représentation de la phrase source comme production. Chaque position génère un vecteur de taille hidden_size (768 dans BERT Base).

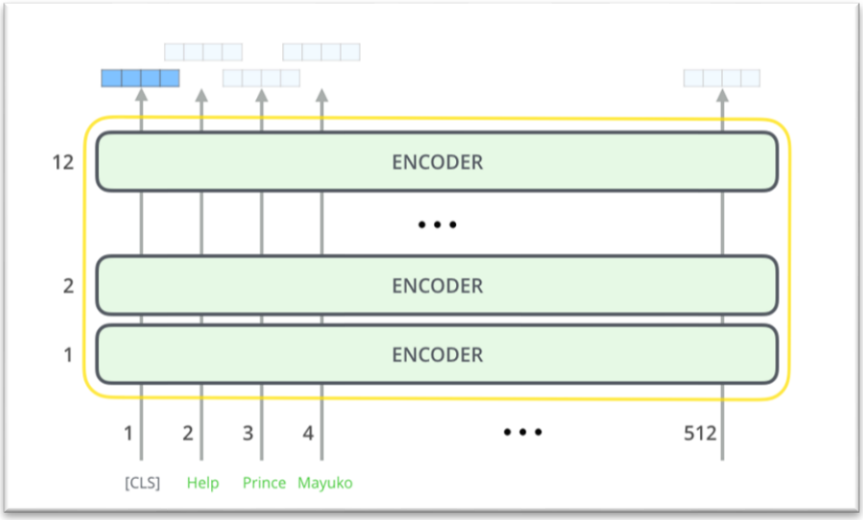


Figure 1.4 Sortie du Modèle [W11]

Ce vecteur peut maintenant être utilisé comme entrée pour un classificateur de notre choix. On obtient d'excellents résultats en utilisant simplement un réseau de neurones à une seule couche comme classificateur comme le montre la Figure 1.5

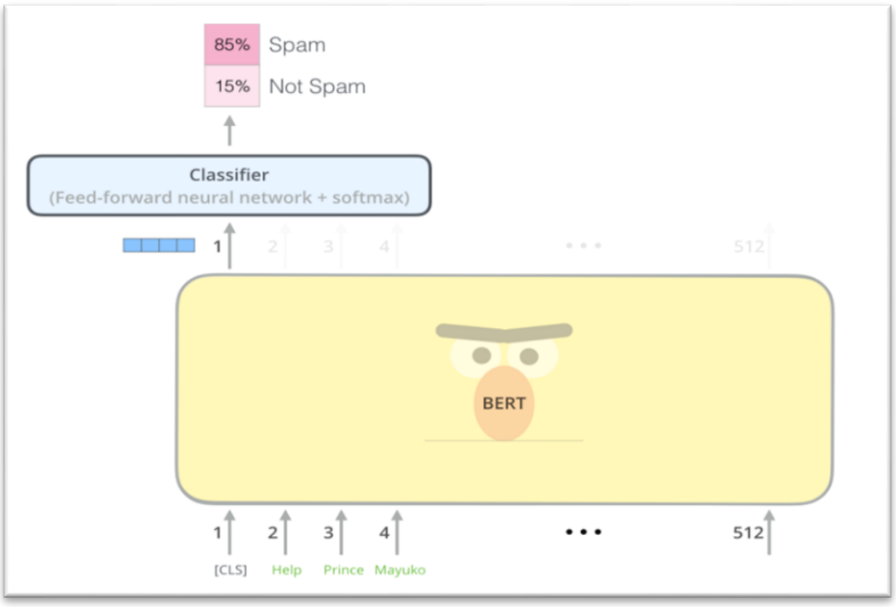


Figure 1.5 Utilisation d'un réseau de neurones à une seule couche comme classificateur [W11]

Il existe deux versions de BERT :

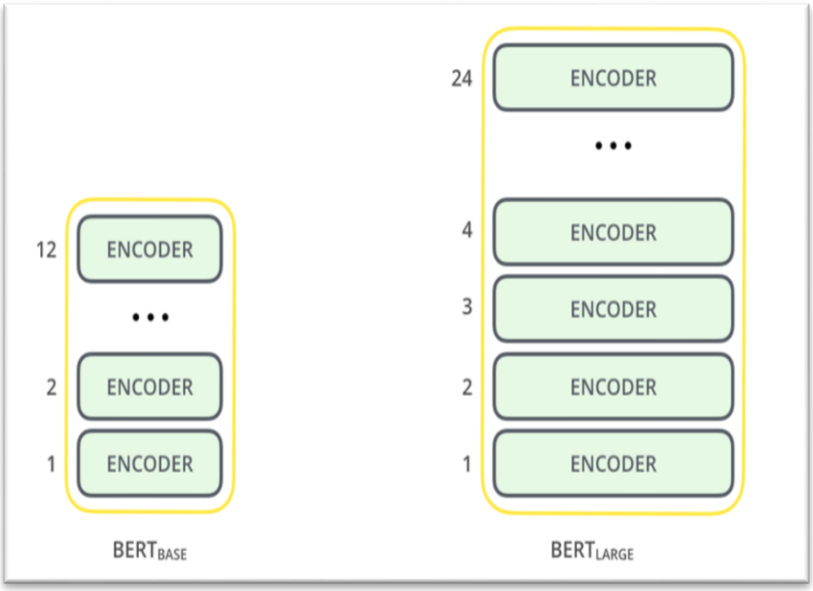


Figure 1.6 Les versions de Bert (Bert Large et Bert base) [W11]

- ***BERT Base*** contient 12 encodeurs de transformateur, 12 têtes d'attention et 110M de paramètres. Nous allons utiliser BERT Base pour des raisons de matériels et de performance.
- ***BERT Large*** contient 24 encodeurs de transformateur, 16 têtes d'attention et 340 millions de paramètres.

1.7.3 Procédures de pré-formation Vs procédures de réglage fin pour le BERT

Le modèle BERT pré-formé peut être affiné avec une seule couche de sortie supplémentaire pour apprendre un large éventail de tâches telles que la traduction automatique neuronale, la réponse aux questions, l'analyse des sentiments et la synthèse de texte sans modifications substantielles de l'architecture spécifique à la tâche [61].

Détaillions les composants d'un modèle BERT : Bidirectionnel, Représentation Encodeur , Transformer, incorporation de mot pré-formés

- **Bidirectionnel :** Contrairement à d'autres modèles de représentation linguistique, BERT, étant un modèle contextuel, est conçu pour pré-entraîner des représentations bidirectionnelles profondes à partir de texte non étiqueté en conditionnant conjointement le contexte gauche et droit dans toutes les couches [61].

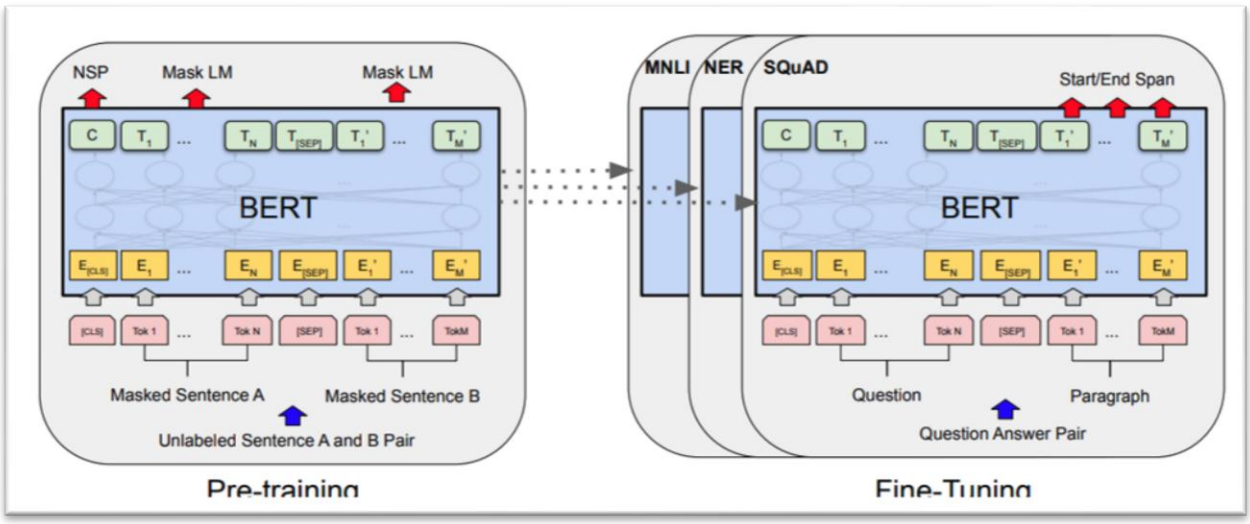


Figure1.7 Préformation et Réglage fin [W11]

Les modèles de langage standard (LM) ne peuvent être entraînés que de gauche à droite ou de droite à gauche. C'est ce qui fait la particularité du BERT : il allège la contrainte d'unidirectionnalité en s'appuyant sur un Mask Language Model (MLM), inspiré de la tâche de Cloze (Taylor, 1953). Le masquage signifie que nous devons masquer certains jetons d'entrée au hasard, puis prédire ces jetons masqués. BERT a été formé en masquant 15% des jetons en pourcentage

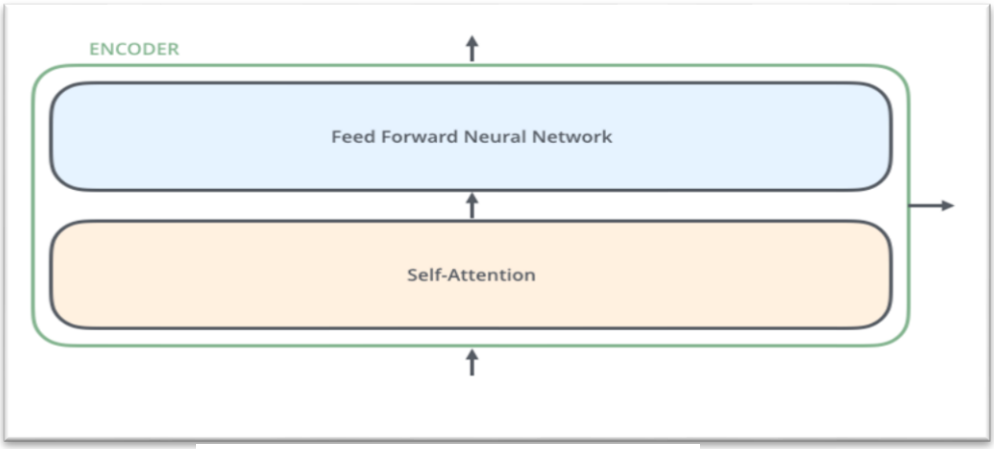


Figure 1.8 Le Codeur ER [W11] .

- **Représentation du codeur (ER)**

L'Encoder a deux couches : Auto-attention et Feed Forward Neural Network

▪ **Auto-attention :** regardez chaque mot et les positions de tous les autres mots qui l'entourent. Par exemple, lorsque le modèle traite le mot it, il examine également tous les autres mots de la phrase en même temps tout en accordant une attention particulière aux mots **The** et **animal**. Cela signifie que les mots d'une longue séquence ont des chances égales.

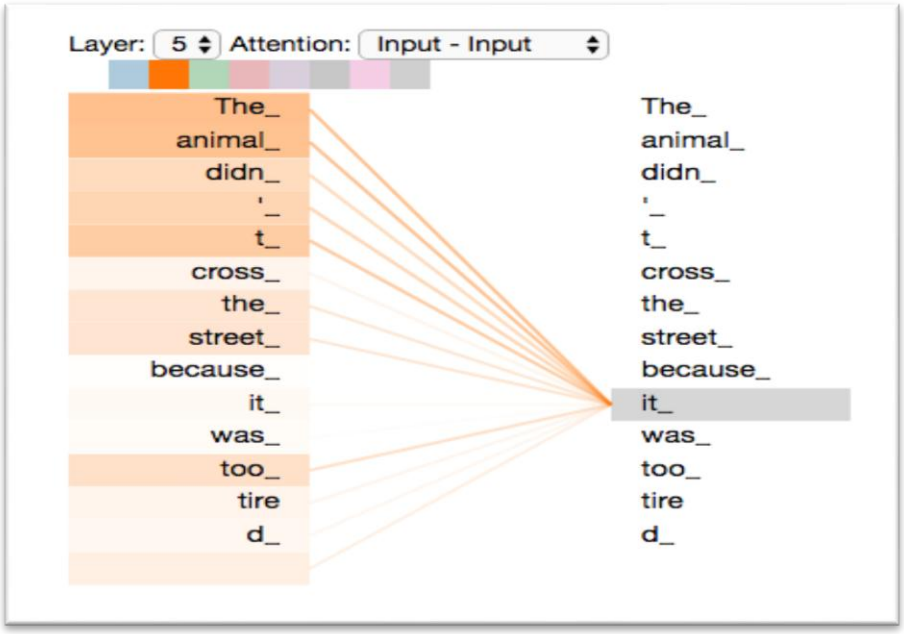


Figure 1.9 Auto attention [W11]

▪ **Forward Neural Network** : calcule les caractéristiques hiérarchiques non linéaires.

L'architecture BERT se compose d'une pile de fichiers Encodeur. Il prend une séquence de mots en entrée et la traite dans l'Encoder pile. Chaque couche applique l'auto-attention (nous en discuterons ci-dessous) et transmet ses résultats à travers un réseau de rétroaction, puis les transmet au codeur suivant [W11]

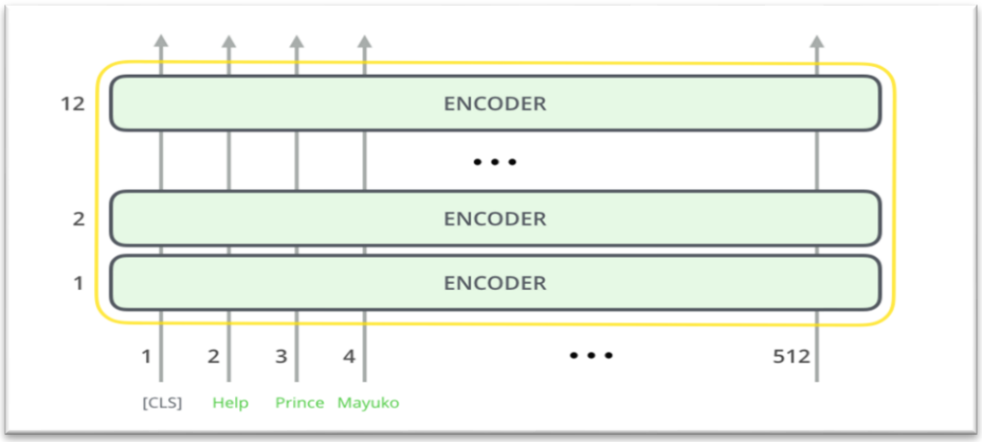


Figure 1.10 L'encodeur de base [W11]

- **Transformer :**

BERT utilise Transformer, un mécanisme d'attention qui apprend les relations contextuelles entre les mots ou les sous-mots d'un texte, c'est-à-dire apprend l'importance de tous les mots de la phrase en examinant leurs positions spécifiques dans la phrase. Le transformateur lit des séquences entières de jetons à la fois et ne prend pas en compte la directionnalité tandis que le mécanisme d'attention prête attention à un mot spécifique dans la phrase simultanément.

Alors que la plupart des modèles de NLP sont basés sur des réseaux de neurones récurrents et à convolution, l' article Attention Is All You Need a présenté le modèle Transformer qui est basé uniquement sur les mécanismes d'attention, c'est-à-dire en se passant entièrement de la récurrence et des convolutions [54].Certains des modèles NLP les plus avancés incluent aujourd'hui des transformateurs dans leur architecture.

L'architecture du modèle de transformateur.

1.7.4 L'architecture du modèle Bert:

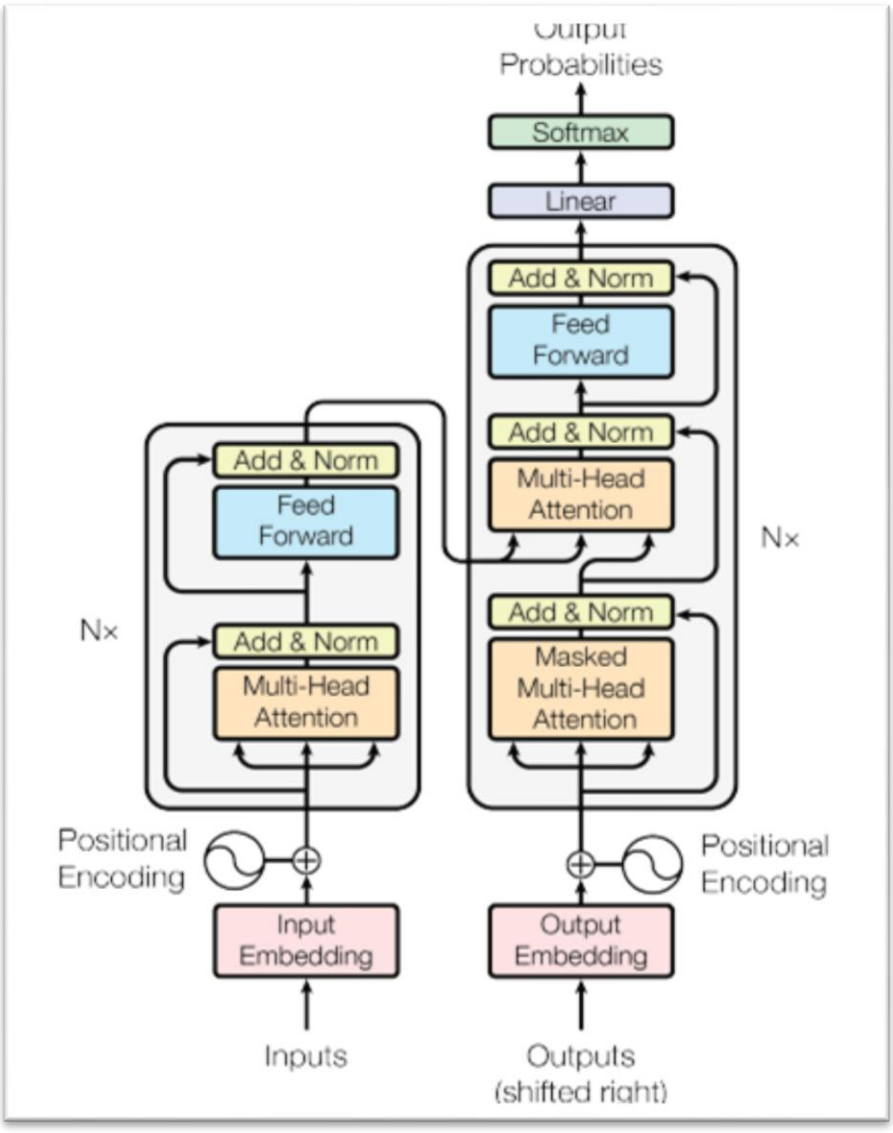


Figure 1.11 L'architecture du modèle de transformateur [62]

modèle Transformer ci-dessus comporte 2 parties :

L'encodeur et le décodeur, chacun se compose de deux sous-couches :Auto-Attention Network (SAN) modélise le contexte et Feed-Forward Network (FFN) calcule les caractéristiques hiérarchiques non linéaires plus de détail dans [4].

- **Encodeur (gauche) :** lit les entrées de texte.
- **Décodeur (à droite)** : effectue des prédictions en fonction des entrées de texte.

La séquence d'entrée est introduite dans le Encoder et progresse vers le haut. Sa sortie est ensuite introduite dans le Décodeur.

Intégration Positionnel permet au modèle d'apprendre la position relative des mots dans la phrase.

1.7.5 Incorporation de mots pré-formés

Le BERT a également adapté l'approche de formation basée sur les fonctionnalités d'incorporation de mots pré-formés ELMo (Embedding from Language Model) où un réseau neuronal pré-formé produit des incorporations de mots qui sont ensuite utilisées comme fonctionnalités dans les modèles de langage bidirectionnels profonds (biLM) ultérieurs. Le modèle BERT est formé sur 2500 millions de mots sur Wikipedia anglais et 800 millions de mots sur BooksCorpus . Pour plus de détail sur le modèle BERT, veuillez se référencer à [4].

1.8 Conclusion

Dans ce chapitre, nous avons présenté les différentes méthodes et techniques dans le domaine de la recommandation conversationnelle, et plus précisément dans l'élicitation de préférence par les techniques du traitement du langage naturel. Nous avons résumé les travaux de l'état de l'art dans le domaine de l'élicitation des préférences dans un tableau comparatif avec leurs avantages et inconvénients.

Ensuite, nous avons présenté d'une manière détaillée le modèle BERT qui est le modèle le plus récent du deep learning qui concerne le traitement du langage naturel, et grâce à son efficacité éprouvée il est appliqué récemment dans quelques travaux sur la recommandation [1], ce qui nous a influencé de l'utiliser dans les SRC afin d'en analyser les sentiments des utilisateurs [2][3] à partir de leur conversations avec le système.

Dans le chapitre suivant, nous allons présenter notre système et conception avec le modèle BERT.

Chapitre 2 : Etude de Projet

2.1 Introduction

Dans ce chapitre, nous présentons notre solution proposée pour l'élicitation de préférence, plus précisément l'analyse des sentiments des utilisateurs à partir de leurs commentaires dans le contexte des systèmes de recommandation conversationnels. Afin d'effectuer une analyse assez performante, nous visons l'utilisation du modèle BERT, Le modèle du deep learning le plus récent et le plus performant pour le traitement automatique du langage naturel.

Nous commençons par une architecture du système, puis nous présentons un diagramme du processus général de notre système avec explication de chaque partie.

Ensuite, nous présentons toutes les étapes suivies par notre système afin d'effectuer une analyse du sentiment, du prétraitement des commentaires à la classification des sentiments en utilisant deux algorithmes parmi ceux du modèle BERT, à travers un exemple explicatif pour mieux comprendre toutes les notions et étapes.

2.2 Contexte du projet et problématique

Traditionnellement, les recommandations étaient basées sur un modèle unique basé sur les actions passées des utilisateurs. Cependant, les recommandations conversationnelles permettent d'obtenir progressivement les préférences de l'utilisateur en effectuant un dialogue utilisateur-système. Par exemple, les systèmes peuvent demander quelle est la préférence de l'utilisateur pour une caractéristique associée aux items. Dans de tels systèmes, il est important de concevoir une conversation efficace, qui minimise le nombre de questions posées tout en maximisant les informations de préférence obtenues.

Le dialogue entre l'utilisateur et le système nécessite un traitement du langage naturel (Natural Language Processing NLP) pour analyser et traiter les commentaires fournis par l'utilisateur, dont l'élicitation des préférences est l'un des principaux objectifs de cette tâche.

D'où la question qui se pose est comme suit : « *Comment peut-on extraire efficacement les préférences des utilisateurs à partir de leur réponses dans les conversations avec le système, afin de fournir des recommandations pertinentes dans les SRC*? »

2.3 Approche Proposée

Afin de répondre à cette problématique, on propose dans ce manuscrit de concevoir un nouveau modèle pour l'élicitation des préférences des utilisateurs dans les SRC. Le modèle est la fois contextuel et **basé sur un récent** modèle de deep learning pré-entraîné, appelé BERT (Bidirectional Encoder Représentation from Transformers)**, qui permet le traitement** automatique du langage naturel pour traiter les réponses (commentaires) fournies par l'utilisateur afin d'analyser ses sentiments et déduire ses intentions et préférences qui seront utilisées comme entrée pour le SRC.

2.4 Motivations

Le Traitement Automatique du Langage Naturel (Natural Language processing NLP) est un sous domaine de l'apprentissage automatique (Machine learning) qui a pour objectif de donner à des programmes informatiques la capacité de comprendre et traiter le langage humain.

En octobre 2018, la filière d'intelligence artificielle de Google (Google AI) a publié un modèle de deep learning pré-entraîné, appelé BERT, capable de résoudre plusieurs problématiques de NLP. Bert offre les meilleurs résultats actuellement. Google a même avancé

que ce modèle dépassait les performances humaines sur certaines tâches, ce qui est une grande étape pour tout algorithme de machine learning.

Dans ce travail, on a visé l'utilisation du modèle de langage BERT pour pouvoir effectuer un traitement efficace des réponses (commentaires) de l'utilisateur, afin d'en attirer ses sentiments et déduire ses préférences.

2.5 Objectifs

Nous avons fixé un ensemble d'objectifs à atteindre afin de répondre à notre problématique.

- Construire un modèle d'analyse de sentiments à l'aide de BERT
- Effectuer un traitement automatique avec TALN et en utilisant le modèle BERT dans le domaine de la recommandation, afin d'aider l'utilisateur a surmonté le problème de choix et d'aide à la décision pour des recommandations automatiques dans les SRC.
- Utilisation et application des NLP dans le domaine de la recommandation conversationnelle pour l'analyse et l'extraction des préférences utilisateur
- Application de plusieurs Modèles BERT dans les SRC comparaison des résultats avec différents jeux de données.

2.6 Conception et Architecture

Dans cette section, nous allons présenter en détails notre modèle d'analyse de sentiments à partir de commentaires fournis par les utilisateurs à l'aide du modèle BERT. Nous commençons par une architecture générale du système. Ensuite, nous présentons les étapes détaillées de notre modèle proposé.

2.6.1. Architecture Générale du Système (*Preference Elicitation In Convertiationnel Recommander System PE-CRS*)

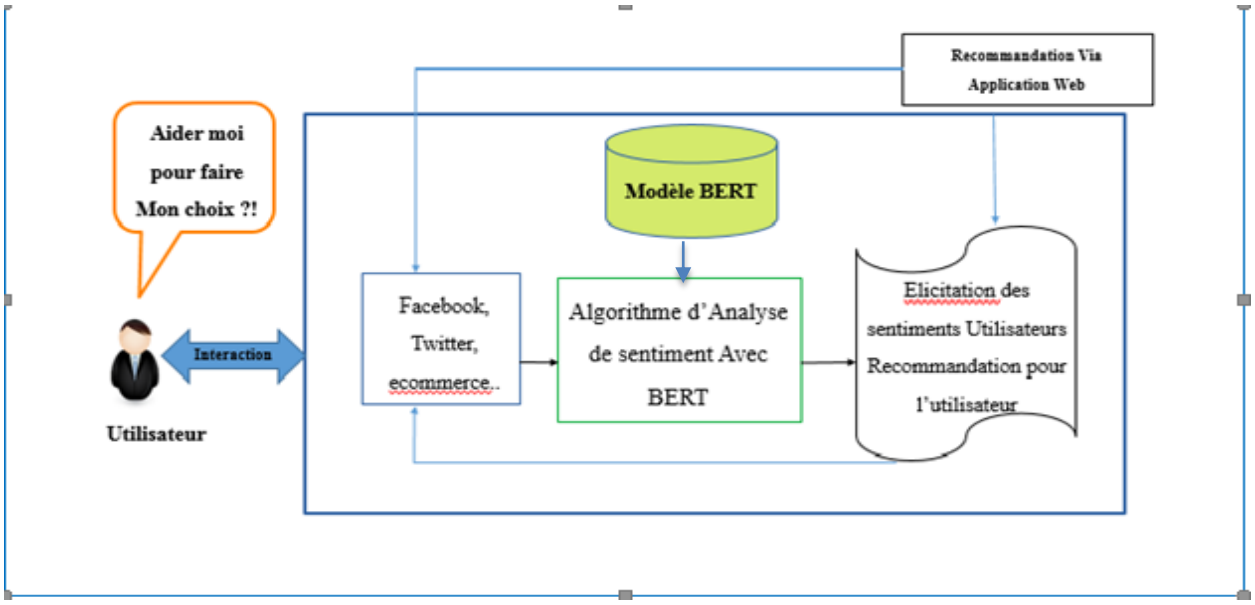
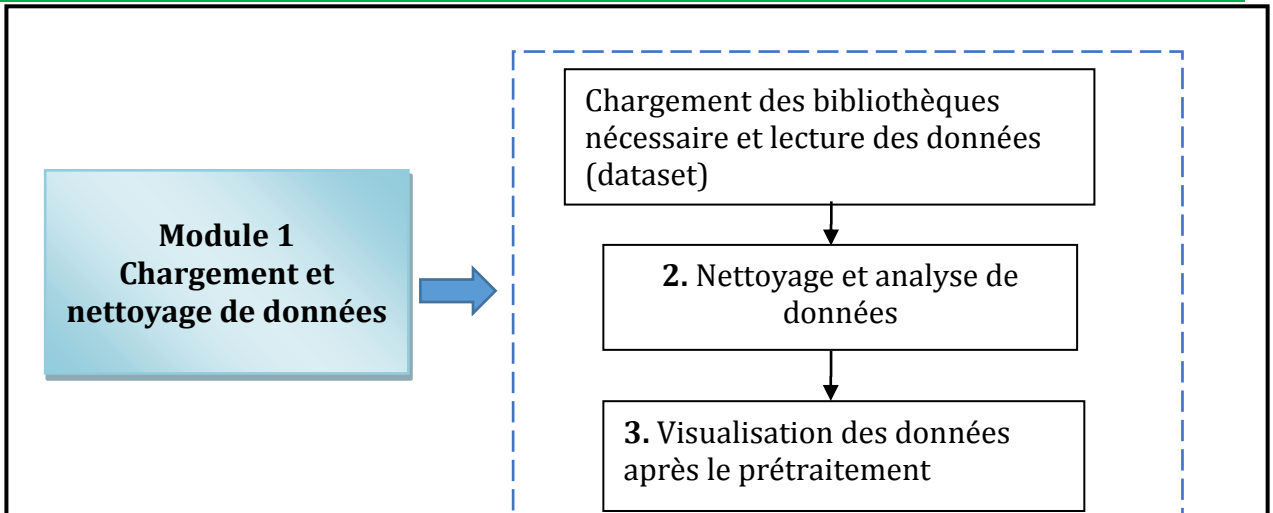


Figure 2.1 Architecture générale de système d' PE-CRS

Cette architecture décrit le système **PE-CRS** du proposé, en expliquant comment les différentes composantes du système interagir et communiquent entre elles et répond à la question **Aider moi pour faire mon choix ?**

Afin de réponde à cette question un utilisateur émet une requête ou consulte un site web de ventre en ligne ou émet un commentaire via un site de communication social via le navigateur, l'algorithme d'analyse de sentiment calcule la polarité (sentiment 0 négatif, 1 positif) et essaye de prédire le sentiment via le modèle proposé, et à l'aide de l'apprentissage profond de comprendre le sentiment et essaye de comprendre aux mieux les choix adapter pour cet utilisateur, pour l'aidé à mieux choisir .Cette architecture de recommandation utilise le Modèle pré-formé BERT et la puissance de calcul et la formation des réseaux de neurones profonds (Deep Learning).Et dans les sections suivantes, nous présentons un organigramme explicatif de notre modèle proposé avec étapes détaillées.

2.6.2. Organigramme Explicatif du fonctionnement du système PE-CRS



Notre algorithme comporte quatre modules, où chaque module englobe un ensemble d'étapes,

Nous allons expliquez ces étapes par module comme suit :

2.6.2.1 Module 1 : Chargement et nettoyage de données

- **Chargement des bibliothèques nécessaires et Lecture des données :**

Dans cette étape, on charge les Bibliothèques et on télécharge les jeux de données nécessaires à l'application pour leurs traitements et analyses.

Exemple du code source :

```
import transformers
from transformers import BertModel, BertTokenizer, AdamW, get_linear_schedule_with_warmup
import torch
import gc
gc.collect()
torch.cuda.empty_cache()
import numpy as np
import pandas as pd
import seaborn as sns
df = pd.read_csv("reviews.csv")
```

- **Nettoyage et analyse de données :**

Nous allons Supprimer les mots vides et faire le nettoyage des données et création d'un dictionnaire pour la suite.

Exemple code :

le traitement des langage suppressoion des mot vide ect ..

Les mots vides sont les mots anglais qui n'ajoutent pas beaucoup de sens à une phrase. Ils peuvent être ignorés en toute sécurité sans sacrifier le sens de la phrase. **

In [12]:

```
stop = set(stopwords.words('english'))
punctuation = list(string.punctuation)
stop.update(punctuation)
print('phase terminer')
```

- **Visualisation des données après le prétraitement :**

Visualisation et vérification de données après prétraitement

exemple code :

```
df=pd.read_csv('/kaggle/input/imdb-dataset-of-50k-movie-
reviews/IMDB Dataset.csv')
df.head()
```

review	sentiment	
0	one reviewers mentioned watching oz episode ex...	1
1	wonderful little filming technique fashion giv...	1
2	thought wonderful way spend time hot summer si...	1
3	basically family little boy thinks zombie clos...	0
4	petter time visually stunning film mattei offe...	1

2.6.2.2 Module 2 : Découpage et vectorisation de données

- **Découpage des données en deux sous data : apprentissage et test**

Notre jeu de données sera divisé en deux partie l'une pour l'apprentissage et la formation et l'autre pour le test. Nous utiliserons 80 % de l'ensemble de données comme données d'apprentissage et évaluerons les performances sur les 20 % restants (ensemble restant).

Exemple code : la fonction train_test_split découpe les données en données apprentissage et test.

```
df_train, df_test = train_test_split(df, test_size=0.2, random_state=RANDOM_SEED)
df_val, df_test = train_test_split(df_test, test_size=0.5, random_state=RANDOM_SEED)
```

- **Tokénisation et codage des données d'entrées :**

Le modèle BERT ne prendra pas de texte brut en entrée, donc la première étape que nous devons faire est de transformer le texte en tenseurs numériques (ou jetons), en effectuant les étapes suivantes :

- ✔ Tokéniser les textes en jetons
- ✔ Ajoutez des jetons spéciaux au début et à la fin de chaque phrase
- ✔ Créer une bibliothèque de vocabulaire tokénizer

- ✔ Remplir les séquences à une longueur de séquence fixe (chercher le mot le plus long et fixer la taille)
- ✔ Spécifiez explicitement ce que sont les jetons de remplissage avec le "masque d'attention". Ce dernier est un masque qui remplace chaque séquence par 1 s'il y'a un mot et par 0 s'il y'a les jeton [PAD].(padding jeton de remplissage de vide)

Avant de transmettre les données d'entrée à BERT, nous convertissons l'entrée en intégrations à l'aide de trois couches d'intégration que contient BERT qui sont :

- ✔ **Couche 1 :** Intégration de jetons.
- ✔ **Couche 2 :** Intégration de segments.
- ✔ **Couche 3 :** Position du segment.

Nous expliquons le principe de fonctionnement de chacune de ces couches d'intégration dans ce qui suit :

- ❖ **Incorporation de jetons :**

Tout d'abord, nous avons une couche d'intégration de jetons, qui a pour rôle de diviser la phrase en mots (chaque mot représente un jeton) et l'ensemble des mots de tous les commentaires d'un utilisateur constituent ses jetons.

Afin de comprendre le principe de cette couche nous présentons un exemple explicatif :

***Exemple* :** Considérons les deux phrases suivantes :

- ***Phrase A* :** Algiers is a beautiful city.
- ***Phrase B* :** Visit Algiers.

Tout d'abord, nous symbolisons les deux phrases et obtenons les jetons, comme indiqué ici dans notre exemple :

jetons = [Algiers, is, a, beautiful , city, visit, Algiers]

Ensuite, nous ajoutons un nouveau jeton, appelé jeton **[CLS]**(CLaSsification), uniquement au début de la première phrase:

jetons = [[CLS], Algiers, is, a, beautiful , city, visit, Algiers]

Puis, nous ajoutons un nouveau jeton appelé **[SEP]** (SEParation) à la fin de chaque phrase :

jetons = [**[CLS],** Algiers, is, a, beautiful, city, **[SEP],** visit, Algiers**,[SEP]]**

Le jeton **[CLS]** n'est ajouté qu'au début de la première phrase, tandis que le jeton **[SEP]** est ajouté à la fin de chaque phrase.

Le jeton **[CLS]** est utilisé pour faire des tâches de classification alors que le jeton **[SEP]** est utilisé pour indiquer la fin de chaque phrase.

Maintenant, avant de transmettre tous les jetons à l'encoder, nous convertissons les jetons en *intégrations* à l'aide de la **couche d'intégration** (Embedding) de jetons (deuxième couche).

Comme le montre le tableau suivant 2.1, les jetons **E[CLS]** indiquent l'intégration du jeton **[CLS],** et **EAlgiers** indique l'intégration du jeton Algiers.

Input	[CLS]	Algiers	is	a	beautiful	city	[SEP]	visit	Algiers	[SEP]
Intégration Jetons	E[CLS]	E Algiers	Eis	Ea	Ebeautiful	Ecity	E[SEP]	Evisit	EAlgiers	E[SEP]

Tableau 2.1 Intégration jetons

- **Intégration de segments :**

La couche d'intégration de segments est utilisée pour distinguer entre les deux phrases données **A** et **B**. Considérons les deux mêmes phrases que nous avons vues dans l'exemple précédent ;

Nous devons donner une sorte d'indicateur à notre modèle pour distinguer les deux phrases. Pour ce faire, nous envoyons les jetons embedding (de la couche précédente) comme entrée pour la couche d'intégration de segment.

La couche d'Intégrations de segments renvoie uniquement l'une des deux incorporations **EA** ou **EB** en tant que production. Autrement dit, si le jeton d'entrée appartient à la phrase **A**, alors le jeton sera mappé à l'intégration **EA**, et si le jeton appartient à la phrase **B**, alors il sera mappé à l'intégration **EB** comme le montre le tableau suivant 2.2:

Input	[CLS]	Algiers	is	a	beautiful	city	[SEP]	Visit	Algiers	[SEP]
Intégration de segment	EA	EA	EA	EA	EA	EA	EA	EB	EB	EB

Tableau 2.2 Intégration de segment

- **Position de l'intégration**

Comme nous avons une couche d'intégration de position, donc il faut donner des informations sur la *position des mots* (tokens, jetons) dans notre phrase avant de les envoyer directement a la couche suivante. Donc, on va utiliser **la couche d'intégration de position** pour obtenir l'intégration de position pour chaque jeton dans la phrase.

Le tableau 2.3 présente les différentes intégrations de position de l'exemple précédent, où E0 indique la position d'intégration du jeton **[CLS]**, E1 indique la position du jeton Algeiers, et ainsi de suite :

Input	[CLS]	Algiers	is	a	beautiful	city	[SEP]	visit	Algiers	[SEP]
Intégration de Position	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9

Tableau 2.3 intégration de position

A la fin, on obtient la représentation finale des différentes étapes du codage des phrases A et B, qui sera utiliser comme entrée pour le l'encoder BERT.

Examinons maintenant la représentation finale des données d'entrée. Comme indiqué dans le tableau 2.4 nous convertissons d'abord les phrases d'entrée en jetons d'incorporation et alimentons les jetons au couche, d'intégration de segment et d'intégration de position et les additionner et obtenir le tableau 2.4.

Ensuite, nous résumons tous les incorporations ensemble et alimentons en entrée l'encoder BERT

Input	[CLS]	Algiers	is	a	beautiful	city	[SEP]	visiter	Alger	[SEP]
intégration de Jeton	E[CLS]	E Algiers	Eis	Ea	Ebeautiful	Ecity	E[SEP]	Evisit	Ealgiers	E[SEP]
Intégration de segment	+ EA	+ EA	+ EA	+ EA	+ EA	+ EA	+ EA	+ EB	+ EB	+ EB
Intégration de Position	+ E0	+ E1	+ E2	+ E3	+ E4	+ E5	+ E6	+ E7	+ E8	+ E9

Tableau 2.4 Représentation finale

Maintenant, à cette étape, nous avons appris à convertir l'entrée en représentations incorporées à l'aide de trois couches d'intégrations, dans la partie suivante, nous allons utilisez cette entrée en apprendrons d'avantage sur le tokenizer utilisé par BERT est WordPiece[5] .

Lorsque nous tokenisons à l'aide du tokenizer WordPiece, nous vérifions d'abord si le mot est présent dans notre vocabulaire.

Si le mot est présent dans le vocabulaire, alors nous l'utilisons comme jeton. Sinon nous divisons le mot en sous-mots et nous vérifions Si le sous-mot est présent dans le vocabulaire, alors nous l'utilisons comme un jeton.

De cette façon, nous continuons à diviser et à vérifier le sous-mot avec le vocabulaire jusqu'à ce que nous atteindre des personnages individuels.

[5] **Wordpiece** : est un algorithme de segmentation de sous-mots utilisé dans le traitement du langage naturel. Le vocabulaire est initialisé avec des caractères individuels dans la langue, puis les combinaisons de symboles les plus fréquentes dans le vocabulaire sont ajoutées de manière itérative au vocabulaire.

Maintenant nous allons construire notre analyseur de sentiment à l'aide des deux modèles BERT qui seront présenté.

2.6.3 Modèles d'Analyseur de sentiment basé sur BERT (Modèle bert-base-cased et distilbert-base-uncased):

Nous allons expliquer notre modèle d'analyse de sentiment avec deux variantes différentes de BERT qui sont ((bert-base-cased et distilbert-base-uncased) :

2.6.3.1 Le modèle pré-formé bert-base-cased :

BERT est un modèle de transformateurs pré-entraîné sur un large corpus de données anglaises de manière auto-supervisée. Ce modèle est sensible à la casse. Il est principalement destiné à être affiné sur les tâches en aval qui utilisent la phrase entière (potentiellement masquée) pour prendre des décisions, telles que la classification des séquences, la classification des jetons ou la réponse aux questions. L'équipe qui a publié BERT n'a pas écrit de carte modèle pour ce modèle, donc cette carte modèle a été écrite par l'équipe Hugging Face[6]

Il a été introduit dans un article et publié pour la première fois dans le référentiel PePean

Le modèle a été formé sur 4 TPU cloud en configuration Pod (16 puces TPU au total) pour un million d'étapes avec une taille de lot de 256. La longueur de la séquence était limitée à **128** jetons pour 90 % des étapes et 512 pour les 10 % restants. L'optimiseur utilisé est Adam.

2.6.3.2 Le modèle pré-formé distilbert-base-uncased:

DistilBERT est un modèle de transformateurs, plus petit et plus rapide que BERT base, qui a été pré-entraîné sur le même corpus de manière auto-supervisée, en utilisant le modèle de base BERT comme enseignant. Ce modèle est sans boîtier. Le modèle vise principalement à être affiné sur les tâches qui utilisent la phrase entière (potentiellement masquée) pour prendre des décisions, telles que la classification des séquences, la classification des jetons ou la réponse aux questions. Pour des tâches telles que la génération de texte.

Nous allons maintenant expliquer notre travail avec deux jeux de données différents pour pouvoir comparer les résultats des deux modèles utilisés (BERT-base, distilbert) . Le premier est un Dataset sur les avis des utilisateurs vis à vis les applications Play Store, et le second est un Dataset sur les critiques des films de la base de données IMDB.

[6] ***Hugging Face*** *est une startup française qui s'est fait connaître grâce à un Chatbot créé pour les adolescents elle est très prometteuse qui entend apporter une révolution dans le domaine du ML en fournissant des technologies de traitement du langage naturel open source. Elle a été fondée en 2016 par Clément Delangue et Julien Chaumond, deux ingénieurs français).*

Nous allons expliquer notre modèle via un exemple illustratif d'analyse de sentiment simple : Supposons que nous voulions effectuer une tâche d'analyse des sentiments binaire (0 négative, 1 positive) et disons que nous avons l'ensemble de données montré dans le tableau 2.5 suivant :

Phrase	Label
I love Alegria	**1**
Algiers is a beautiful city	**1**
Visit Algiers	**1**
Khalil hated the movie	**0**
It was good day	**1**
We not love the game	**0**

Tableau 2.5 Ensemble de données

Comme nous pouvons le constater dans le tableau précédent, nous avons des phrases et leur étiquette correspondante, où 1 indique un sentiment positif et 0 indique un sentiment négatif. Nous pouvons ainsi former un classificateur pour classer le sentiment d'une phrase en utilisant l'ensemble de données.

Tout d'abord, nous avons besoin de vectorisés le texte, nous savons d'avance que BERT apprend l'intégration contextuelle (c-à-d **prédit le mot en fonction des contextes *avant* et *après des mots qui l'entourent*).**

- Prenons la première phrase de notre ensemble de données (I love Algéria).
- Tout d'abord, nous tekenisons la phrase en utilisant le générateur de jetons WordPiece .

Considérez la phrase suivante : I love Alegria. Après tokenisation de la phrase, nous obtiendrons le jetons:

jetons = [I, love, Algéria]

Maintenant, nous ajoutons le jeton [CLS] au début et le jeton [SEP] à la fin. Ainsi, notre liste des jetons devient ceci :

jetons = [[CLS], I, love, Algéria, [SEP]]

De même, nous pouvons tokeniser toutes les phrases de notre ensemble d'entraînement, on calculons la phrase la plus longue. Ensuite la choisir comme taille fixe pour toutes les phrases.

Disons que nous fixons la longueur des jetons à 7 (la phrase la plus longue dans notre jeu de donnée est 5 mot plus les jetons CLS et SEP vaut 7) pour toutes les phrases de notre ensemble de données, si nous regardons notre liste de jetons précédente, la longueur des jetons est de 5. Pour

faire les jetons de longueur 7, nous ajoutons un nouveau jeton appelé [PAD]. Ainsi, maintenant notre jetons sont les suivants :

jetons = [[CLS], I, love, Algéria, [SEP], [PAD], [PAD]]

La prochaine étape consiste à faire comprendre à notre modèle que le jeton [PAD] est ajouté uniquement pour correspondre aux longueurs des jetons et ne fait pas partie des jetons réels. Pour ce faire, nous introduisons un masque d'attention.

Nous définissons la valeur du masque d'attention sur 1 dans toutes les positions et sur 0 à la position où nous avons un jeton [PAD], comme illustré ici :

masque_attention = [1,1,1,1,1,0,0]

Ensuite, nous mappons tous les jetons à un identificateur prédéfini par **BERT (ID)** pour chaque jeton unique. Supposons que ce qui suit est l'ID mappé du jeton obtenu :

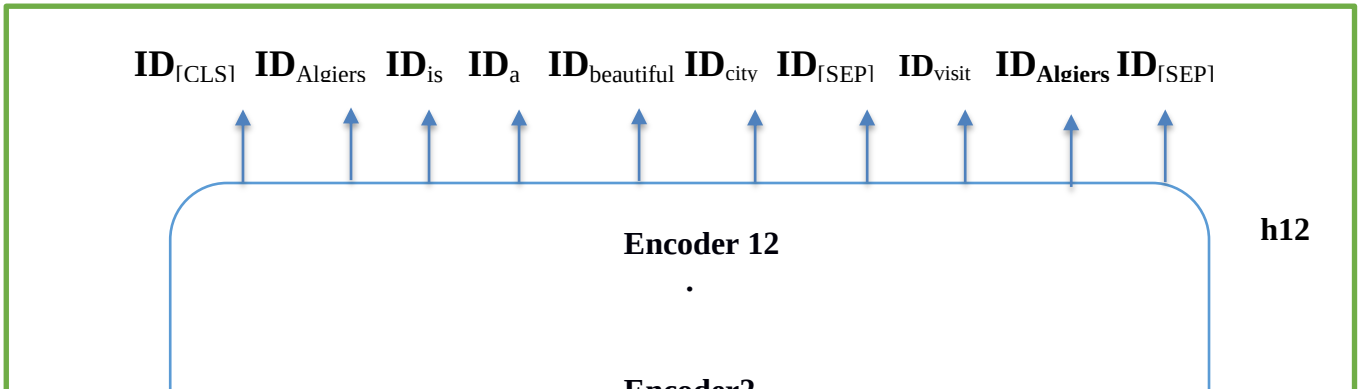
token_ids = [101, 1045, 2293, 3000, 102, 0, 0]

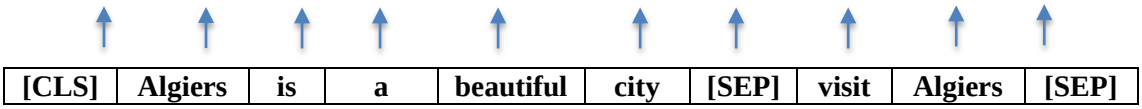
Cela implique que l'ID 101 indique le jeton **[CLS],** 1045 indique le jeton I, 2293 indique Love et 3000 indique le jeton Algéria, et ainsi de suite.

Maintenant, nous alimentons les token_ids et les masque_attention comme entrée du BERT modéliser et obtenir la représentation vectorielle (incorporation) de chacun des jetons.

La Figure 2.3 montre comment nous utilisons le modèle BERT pré-formé pour obtenir les incorporations. Comme nous pouvons le voir, une fois que nous alimentons les jetons en entrée, l'encodeur 1(couche caché h1) calcule la représentation de tous les jetons et l'envoie **à** l'encodeur suivant, qui est l'encodeur 2 (couche caché h2). L'encodeur 2 prend la représentation calculé par l'encodeur 1 en entrée, calcule sa représentation et l'envoie au suivant encodeur, qui est l'encodeur 3 (couche caché h3).

De cette façon, chaque encodeur envoie sa représentation au suivant encodeur au-dessus. L'encodeur final, qui est l'encodeur 12 (couche caché h12), renvoie la représentation finale (incorporation) de tous les jetons de notre phrase.





Input	[CLS]	Algiers	is	a	beautiful	city	[SEP]	visit	Algiers	[SEP]
Jeton intégrations	E[CLS]	E Algiers	Eis	Ea	Ebeautiful	Ecity	E[SEP]	Evisit	Ealgiers	E[SEP]
Intégrations de segment	+ EA	+ EA	+ EA	+ EA	+ EA	+ EA	+ EA	+ EB	+ EB	+ EB
Intégrations de Position	+ E0	+ E1	+ E2	+ E3	+ E4	+ E5	+ E6	+ E7	+ E8	+ E9

Figure 2.3 Représentation BERT pré-formé

Nous avons ajouté le jeton **[CLS]** au début de notre phrase, la représentation du jeton **[CLS]** contiendra la représentation agrégée de l'ensemble de la phrase. Ainsi, nous pouvons ignorer les intégrations de tous les autres jetons et prendre l'intégration de jeton **[CLS]** et affectez-le comme représentation de notre phrase. Ainsi, la représentation de notre phrase « I love Algeria » n'est que la représentation du jeton **[CLS].**

De manière très similaire, nous pouvons calculer la représentation vectorielle de toutes les phrases dans notre ensemble de donnée.

Nous présentons dans ce qui suit un extrait du code source de l'exemple précédent que nous avons programmé avec l'éditeur de code jupyter et l'environnement kaggle pour exécution (environnement complet de deep learning et python avec tensorflow).

❖ **Codage de la phrase :**

```
[40] #sample_txt = ' I love Algerie'
     sample_txt = ' I love Algeria '
     #sample_txt = 'When was I last outside? I am stuck a
```

```
[41] tokens = tokenizer.tokenize(sample_txt)
     token_ids = tokenizer.convert_tokens_to_ids(tokens)

     print(f'  Sentence: {sample_txt}')
     print(f'\n  Tokens: {tokens}')
     print(f'\n  Token IDs: {token_ids}')   # Each token
```

```
  Sentence:  I love Algeria

  Tokens: ['I', 'love', 'Algeria']

  Token IDs: [146, 1567, 11347]
```

```
[42] len(tokens)

     3
```

❖ **Ajout des tokens spéciaux CLS et SEP**

```
[32] tokenizer.sep_token, tokenizer.sep_token_id

     ('[SEP]', 102)
```

[CLS] - we must add this token to the start of each sentence,

```
[33] tokenizer.cls_token, tokenizer.cls_token_id

     ('[CLS]', 101)
```

There is also a special token for padding:

```
[34] tokenizer.pad_token, tokenizer.pad_token_id

     ('[PAD]', 0)
```

```
[46] encoding_test.keys()

     dict_keys(['input_ids', 'attention_mask'])
```

```
[48] print(' length of the first sequence is :  ', len(encoding_test['input_ids'][0]))
     print('\n The input id\'s are : \n', encoding_test['input_ids'][0])
     print('\n The attention mask generated is : ', encoding_test['attention_mask'][0])

      length of the first sequence is :   32

      The input id's are :
      tensor([  101,   146,  1567, 11347,   102,     0,     0,     0,     0,     0,
                  0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
                  0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
                  0,     0])

      The attention mask generated is :  tensor([1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0])
```

We can inverse the tokenization to have a look at the special tokens:

```
tokenizer.convert_ids_to_tokens(encoding_test['input_ids'].flatten())

['[CLS]',
 'I',
 'love',
 'Algeria',
 '[SEP]',
 '[PAD]',
 '[PAD]',
 '[PAD]',
 '[PAD]',
```

❖ **Encodage final de la phrase :**

2.6.3.3 Module 3 : Configuration et création de l'analyseur de sentiment

➢ **Configuration du Modèle BERT (Analyseur de sentiment) :**

Nous utiliserons BERT-base , bert-base-cased avec quelques caractéristiques de ce modèle : 15% des jetons sont masqués.

- Dans 80% des cas, les jetons masqués sont remplacés par **[MASK].**
- Dans 10% des cas, les jetons masqués sont remplacés par un jeton aléatoire (différent) de celui qu'ils remplacent appelé **[UNK].**
- Dans les 10 % de cas restants, les jetons masqués sont laissés tels quels sont.

La figure suivante présente notre modèle d'analyse de sentiment basé sur BERT bert-base-cased :

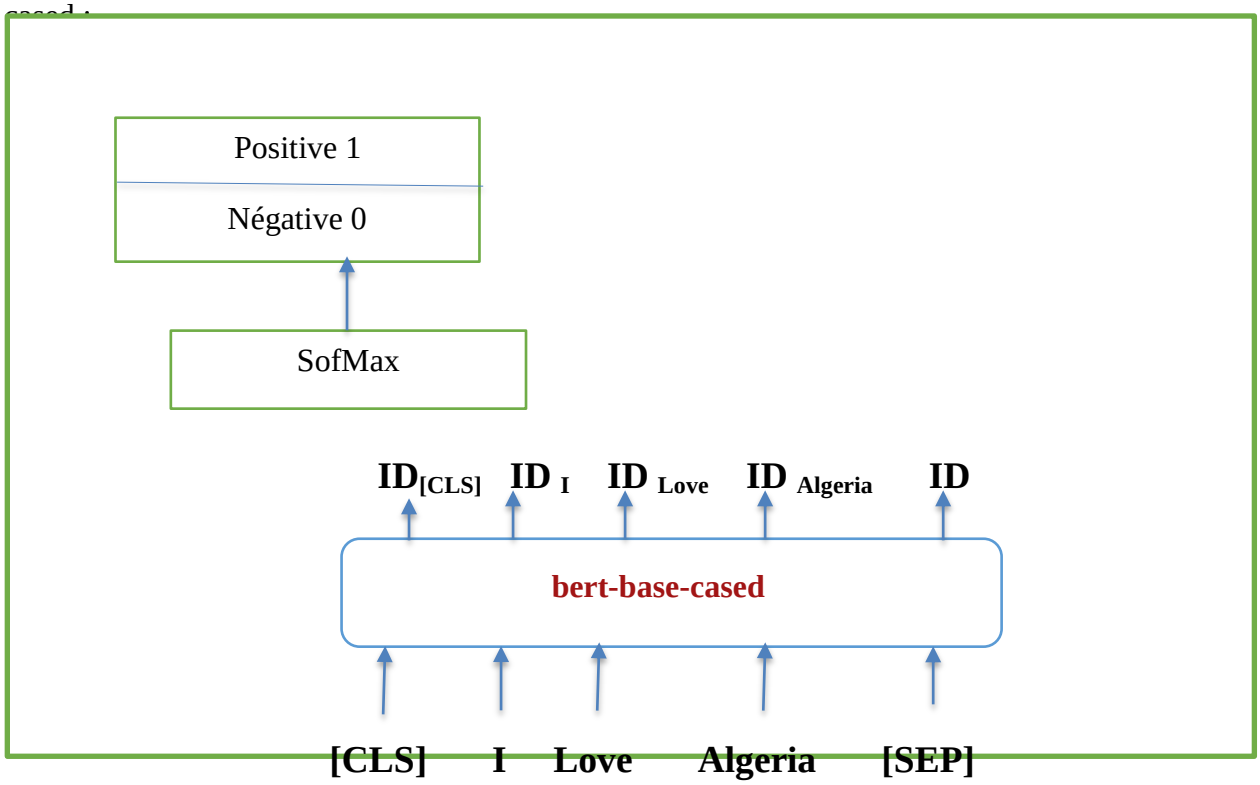


Figure 2.4 Analyse de sentiment base sur BERT bert-base-uncased

Extrait du code qui permet la configuration du modèle :

```
bert_model.config
```

```
BertConfig {
  "_name_or_path": "bert-base-cased",
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "position_embedding_type": "absolute",
  "transformers_version": "4.18.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 28996
}
```

- **Création d'un chargeur et optimiseur de données :**

Dans cette étape, nous présentons comment crée un chargeur de donnée et l'utilisation d'une fonction d'optimisation.

Nous définissons une fonction **DataLoader** qui fournit un moyen de créer un ensemble de données à partir des données déjà chargées en mémoire. Elle accepte les données sous forme de tableau numpy et de tenseur.

Exemple code source :

```
def create_data_loader(df, tokenizer, max_len, batch_size):
  ds = GPReviewDataset(
    reviews=df.content.values,
    targets=df.sentiment.values,
    tokenizer=tokenizer,
    max_len=max_len
  )

  return DataLoader(
    ds,
    batch_size=batch_size,
    #num_workers=4
  )
```

3.6.3.4 Module 4 : Evaluation de l'analyseur

- **Apprentissage du Modèle et affichage des résultats**

Dans cette étape nous procédons à l'apprentissage de notre modèle et pour cela on utilise la fonction Train().

Pour la procédure d'apprentissage de BERT, nous utiliserons l'optimiseur AdamW, fourni par *Hugging Face (*qui permet la correction de la dégradation du poids, et pour la fonction de perte on utilise la fonction CrossEntrepieLoss().

Notre boucle d'apprentissage commence jusqu'à la fin du processus d'apprentissage du modèle.

- *Voici un extrait du code avec 10 époques d'apprentissage :*

```
[ ]  EPOCHS = 10

     optimizer = AdamW(model.parameters(), lr=2e-5, correct_bias=False)
     total_steps = len(train_data_loader) * EPOCHS    # Number of batches

     scheduler = get_linear_schedule_with_warmup(
       optimizer,
       num_warmup_steps=0,     # Recommended in the BERT paper.
       num_training_steps=total_steps
     )

     loss_fn = nn.CrossEntropyLoss().to(device)
```

- Extrait de la boucle d'apprentissage :

```
Epoch 1/10
----------
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_
  FutureWarning,
Train loss 0.7985532670896104 accuracy 0.6345671505662385
Val   loss 0.6866903288778171 accuracy 0.7112462006079028

Epoch 2/10
----------
Train loss 0.545893948571615 accuracy 0.7893896784981379
Val   loss 0.6753270231115008 accuracy 0.7586626139817629

Epoch 3/10
----------
Train loss 0.3912518960221368 accuracy 0.8805198753515239
Val   loss 0.888957380857409 accuracy 0.786018237082067

Epoch 4/10
----------
Train loss 0.28670660891664984 accuracy 0.9287831572546933
Val   loss 1.0068502023666916 accuracy 0.8060790273556232

Epoch 5/10
----------
Train loss 0.20158464814292829 accuracy 0.9530288059588051
Val   loss 1.0650720894396037 accuracy 0.8200607902735563

Epoch 6/10
----------
Train loss 0.1569837243123883 accuracy 0.964201565706468
Val   loss 1.066982521163243 accuracy 0.8243161094224924
```

➢ **Evaluation du modèle :**

Pour déterminer l'efficacité de notre système en termes de prédiction et de classification de sentiment, des indicateurs comme la précision et la perte sont utilisés. Voici un exemple du code source :

Evaluation

```
[ ]  test_acc, _ = eval_model(
       model,
       test_data_loader,
       loss_fn,
       device,
       len(df_test)
     )

     test_acc.item()
```

```
/usr/local/lib/python3.7/
  FutureWarning,
0.8212765957446809
```

- **Les résultats de l'évaluation :**

```
plt.plot(history['train_acc'], label='train accuracy')
plt.plot(history['val_acc'], label='validation accuracy')

plt.title('Training history')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()
plt.ylim([0, 1]);
```

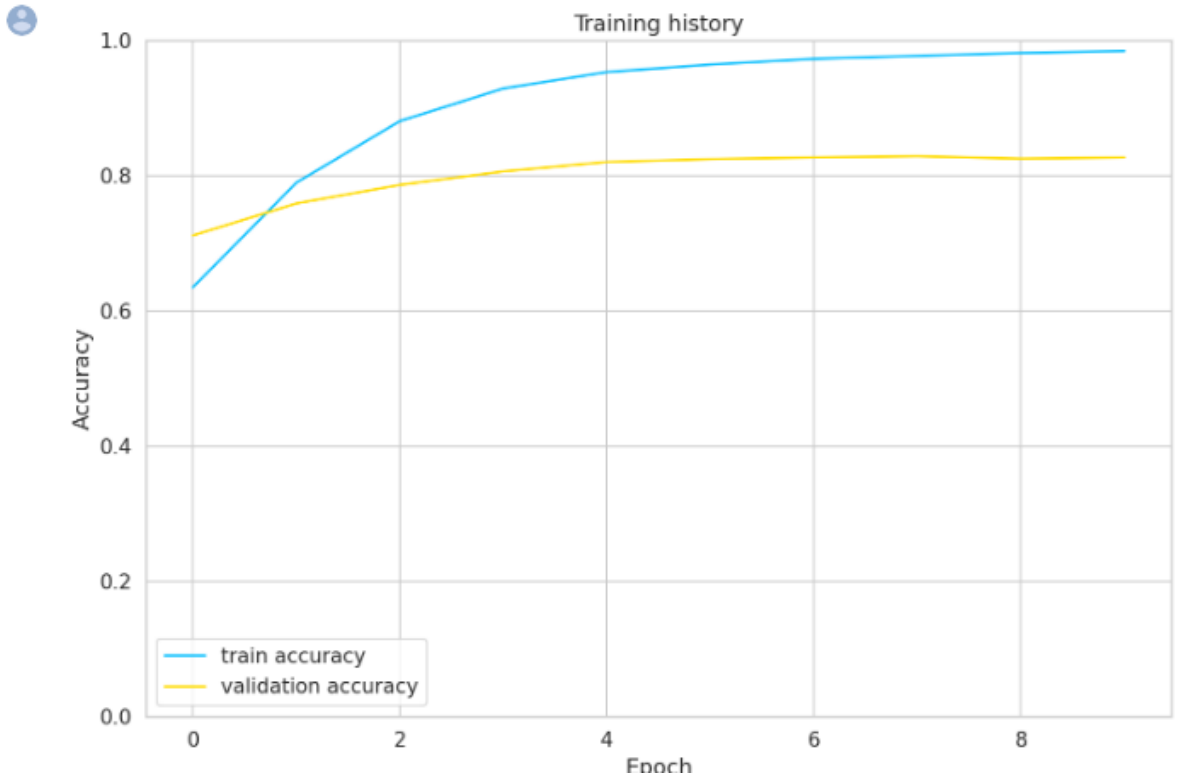


Figure 2.5 Résultat de précision du modèle

2.7. Conclusion

Dans ce chapitre, nous avons proposé un modèle d'élicitation de préférences (analyse de sentiments) à partir des commentaires des utilisateurs. Tout d'abord, nous avons présenté l'architecture du notre système. Ensuite, un diagramme explicatif du processus de fonctionnement de notre système est présenté suivie d'une explication de toutes les étapes en détails et illustrée avec des extraits de codes.

Dans notre système, deux modèles de BERT (bert-base-uncased et distilbert-base-uncased) ont été utilisés pour créer notre analyseur de sentiments.

Afin d'évaluer la performance de ces deux modèles, une série de tests est effectuée sur différents Datasets et les résultats sont présentés dans le chapitre suivant.

Chapitre 3 : Expérimentation et Test

3.1 Implémentation

Dans ce chapitre, nous allons présenter les différents outils et la plateforme utilisée pour la mise en œuvre de notre système, et les métriques utilisées pour nos expériences menées visant à répondre aux questions suivantes :

- **Question de recherche 1** – L'utilisation des nouveaux modèles BERT proposés sont-ils performants avec l'utilisation de l'analyse de sentiment pour l'enrichissement de profil utilisateur ?
- **Question de recherche 2** – Est-ce-que l'utilisation de l'architecture basée sur les modèles BERT est-elle bénéfique pour les SRs modernes et complexes ?

Réponse aux questions : oui on a trouvé que l'utilisation des modèles BERTs est plus bénéfique et elle peut apporter un plus à la recherche surtout dans le domaine PNL et la recherche d'information et elle peut remplacer d'autres modèles comme RNN et CNN mais reste à découvrir plus de modèle de BERT avec d'autre jeux de données (en arabe par exemple).

3.1.1 Outils de développements

3.1.1.1 Langage de programmation et plateforme

Python est un langage portable, dynamique, extensible, gratuit, qui permet une approche modulaire et orientée objet de la programmation. Python est développé depuis 1989 par Guido van Rossum et de nombreux contributeurs bénévoles [65].

- **Caractéristiques du langage**

Voici les principales caractéristiques de Python :

- Python est portable,
- Python gère ses ressources (mémoire, descripteurs de fichiers...) sans intervention du programmeur, par un mécanisme de comptage de références (proche, mais différent, d'un garbage collector).
- Il n'y a pas de pointeurs explicites en Python.
- Python est (optionnellement) multi-thread.

- Python est orienté-objet. Il supporte l'héritage multiple et la surcharge des opérateurs. Dansson modèle objets, et en reprenant la terminologie de C++, toutes les méthodes sont virtuelles.
- Python intègre, comme Java ou les versions récentes de C++, un système d'exceptions, qui permettent de simplifier considérablement la gestion des erreurs.[63]

Pourquoi Python?

- Python fonctionne sur différentes plates-formes (Windows, Mac, Linux, Raspberry Pi, etc.).
- Python a une syntaxe simple semblable à la langue anglaise.
- Python a une syntaxe qui permet aux développeurs d'écrire des programmes avec moins de lignes que certains autres langages de programmation.
- Python fonctionne sur un système interprète, ce qui signifie que le code peut être exécuté dès qu'il est écrit. Cela signifie que le prototypage peut être très rapide.
- Python peut être traité de manière procédurale, orientée objet ou de manière fonctionnelle [63].

Pour Plus De Détail Sur Le Langage Python Consulter Le Livre Open Source « Apprendre à programmer avec Python » [63]

- **Plate Forme Utiliser :** La plate-forme utiliser pour nos expériences est le système d'exploitation Fedora **V35**.
- **Fedora(GNU/Linux)** : est un système d'exploitation libre (distribution GNU/Linux communautaire) développée par le projet Fedora et sponsorisée par l'entreprise Red Hat, qui lui fournit des développeurs ainsi que des moyens financiers et logistiques. Fedora est prompte à inclure des nouveautés et peut être considérée comme une vitrine technologique pour le monde du logiciel libre, auquel elle contribue largement via les projets amont tels que le noyau Linux, GNOME, Network Manager, PackageKit, PulseAudio, Wayland, la célèbre suite de compilateurs GCC et bien d'autres. Tous les six mois une nouvelle version sort ; elle est maintenue treize mois.

➢ Technologies et Bibliothèques utilisées

Nous avons utilisez pour le développement et le test de notre modèle les bibliothèques et le modèle **BERT, KERAS et TensoFlow de Google.**

- **BERT (Bidirectional Encoders Representations from Transformers)**
- **KERAS La bibliothèque Python Deep Learning.**

- **TensoFlow plateforme Google pour Machine Learning.**

➢ **Pourquoi BERT?**

L'acronyme BERT signifie « Bidirectional Encoder Representations from Transformers ». Lancé en octobre 2019, son objectif est, comme souvent chez Google, de fournir des résultats toujours plus pertinents aux recherches des internautes. Comment s'y prend-il ? En essayant de comprendre les intentions formulées derrière les expressions naturelles.

Avant BERT, les mots-clés étaient pris individuellement sans forcément créer des liens entre eux. Désormais, **les phrases sont prises dans leur ensemble tout en tenant compte du contexte**. Ce qui inclut les prépositions, les expressions longues et complexes, les mots de liaison, les nuances polysémiques, les pronoms… tout ce qui peut faire partie du langage naturel et qu'un internaute peut inclure dans sa requête.

Google BERT constitue donc **une solution pour la recherche vocale** toujours croissante, puisque cette dernière comprend, en majorité, des questions ou des expressions naturelles et conversationnelles.

Cependant, l'algorithme permet également au moteur de recherche de mieux comprendre les contenus qu'il indexe. [W6]

- **KERAS :** est une API de réseaux de neurones de haut niveau, écrit en Python et capable de fonctionner au-dessus de tensorflow , CNTK ou Théano . Il a été développé pour permettre une expérimentation rapide. *Pouvoir faire de la recherche de qualité est essentiel pour pouvoir passer de l'idée au résultat le plus rapidement possible.*

Keras est une bibliothèque d'apprentissage en profondeur qui :

- Permet un prototypage simple et rapide (convivialité, modularité et extensibilité).
- Prend en charge les réseaux convolutionnels et les réseaux récurrents, ainsi que les combinaisons des deux.
- Fonctionne de manière transparente sur le processeur et le processeur graphique.

- **TensorFlow :** est un outil open source d'apprentissage automatique développé par Google. Le code source a été ouvert le 9 novembre 2015 par Google et publié sous licence Apache.

Il est fondé sur l'infrastructure DistBelief, initiée par Google en 2011, et est doté d'une interface pour Python et Julia.

TensorFlow est l'un des outils les plus utilisés en IA dans le domaine de l'apprentissage machine.

3.1.2 Matériels utilisés

La machine utilisée pour nos expériences est un pc portable qui a les caractéristiques suivantes :

- processeur : Intel CORE Q72 I7 vitesse 1.6Ghz x8
- type de processeur : 64 bit
- mémoire RAM : 4 GO
- Disques dur : 1 To

carte graphique : Gforce 5 Go de mémoire

3.1.3 Les métriques d'évaluations et les fonctions de perte

Dans cette phase le modèle doit être testé pour s'assurer de sa capacité de généralisation sur les données non utilisées dans la phase d'entrainement. Il couvre 20% des données originales.

Les performances des classificateurs sont évaluées en fonction des résultats de la comparaison des étiquettes de class réelles en utilisant les mesures suivante, précision, perte.

Pour le teste de notre modèle, on calcule la précision sur un corpus de test jamais vue par le modèle, et si la précision est élevée (plus de 70%) on occupe le modèle sinon, on refait le traitement avec d'autres hyperparamètres.

Une métrique est une fonction utilisée pour juger de la performance de notre modèle. Les fonctions de métrique doivent être fournies dans le paramètre metrics lorsqu'un modèle est compilé.

Pour notre travail nous avons utilisés les métriques suivantes :

Fonction d'erreur	*Loss : binary cros entropy*
Fonction de précision	*accuracy*
Optimiseur	*Adam from keras import metrics* *Optimizer=adamW(modelparameters(),lr=2e-5,correct, bias=False*

Tableau 3.1 les Métriques d'évaluation

Vous pouvez soit transmettre le nom d'une métrique existante, soit transmettre une fonction symbolique Theano / TensorFlow.

Le Résultats est une Valeur de tenseur unique représentant la moyenne du tableau de sortie pour tous les points de données.

3.1.4 Utilisation du modèle BERT

On prend un texte (commentaire, tweet, critique...etc), on le prétraite et le donne à notre modèle de catégorisation des sentiments sauvegardés comme une entrée, et notre modèle va catégoriser notre texte soit en sentiment négatif ou positif.

3.1.5 Les jeux de données utilisées pour les tests (DataSet) :

Nous Avons utilisez plusieurs jeux de données différent détailler comme suit :

3.1.5.1 Le Dataset Avis Google Play Store (reviews.csv):

Google Play, anciennement Android Market, est un service de distribution numérique exploité et développé par Google. Il sert de magasin d'applications officiel pour les appareils certifiés fonctionnant sur le système d'exploitation Android, permettant aux utilisateurs de parcourir et de télécharger des applications développées avec le kit de développement logiciel (SDK) Android et publiées via Google. Il a franchi plus de *82 milliards de téléchargements d'applications* avec plus de *3,5 millions d'applications publiées*, ce qui en fait le plus grand magasin d'applications au monde.

Les données contiennent plus de 12 000 formats (reviews.csv) avis de différentes applications de magasin d'applications par de vrais utilisateurs. Les données contiennent également la note qui leur a été attribuée afin qu'elles puissent être classées en avis positifs ou négatifs.

Les Données sont divisées on 80 % données d'entraînement, 10 % de données de test et 10 % de données de validation.

3.1.5.2 Le Dataset IMDB de 50 000 critiques de films

Ensemble de données IMDB (Internet Movie Database, abrégé en IMDb[7]), est une base de données en ligne sur le cinéma mondial, sur la télévision, et plus secondairement les jeux vidéo) contenant 50 000 critiques de films pour le traitement du langage naturel ou l'analyse de texte. Il s'agit d'un ensemble de données pour la classification des sentiments binaires contenant beaucoup plus de données que les ensembles de données de référence précédents. Il se compose d'un ensemble de 25 000 critiques de films très polaires pour la formation et de 25 000 pour les tests. Nous devons donc prédire le nombre d'avis positifs et négatifs à l'aide d'algorithmes de classification ou d'apprentissage en profondeur. Ici, nous allons donc utiliser BERT et l'entraîner à classer correctement les avis comme positifs/négatifs.

7 https://datasets.imdbws.com/

3.2 Résultats obtenus et discussion

3.2.1 Performance du modèle Basé sur l'analyse des sentiments et BERT (Distilber) avec le dataset IMDB (3 époques d'apprentissage)

Dans la première expérimentation nous avons entrainé notre modèle de classification binaire de sentiment sur l'ensemble de jeu de données IMDB contient 50 000 critiques de films pour l'analyse de texte Il s'agit d'un ensemble de données pour la classification des sentiments binaires contenant beaucoup de données. Il se compose d'un ensemble de 25 000 critiques de films très polaires pour la formation et de 25 000 pour les tests.

L'apprentissage se fait en appelons la fonction fit(), qui entraînera le modèle en découpant les données en "lots" de taille batch size de 32 , et en itérant à plusieurs reprises sur l'ensemble de données pour un nombre donné de époque = 3.

L'instruction concernée est :

```
history = model.fit(x_train,y_train,batch_size = 32
                    ,validation_data=(x_test,y_test),epochs = 3)
```

L'objet renvoyé **history** contient un enregistrement des valeurs de perte et des valeurs métriques pendant l'entraînement.

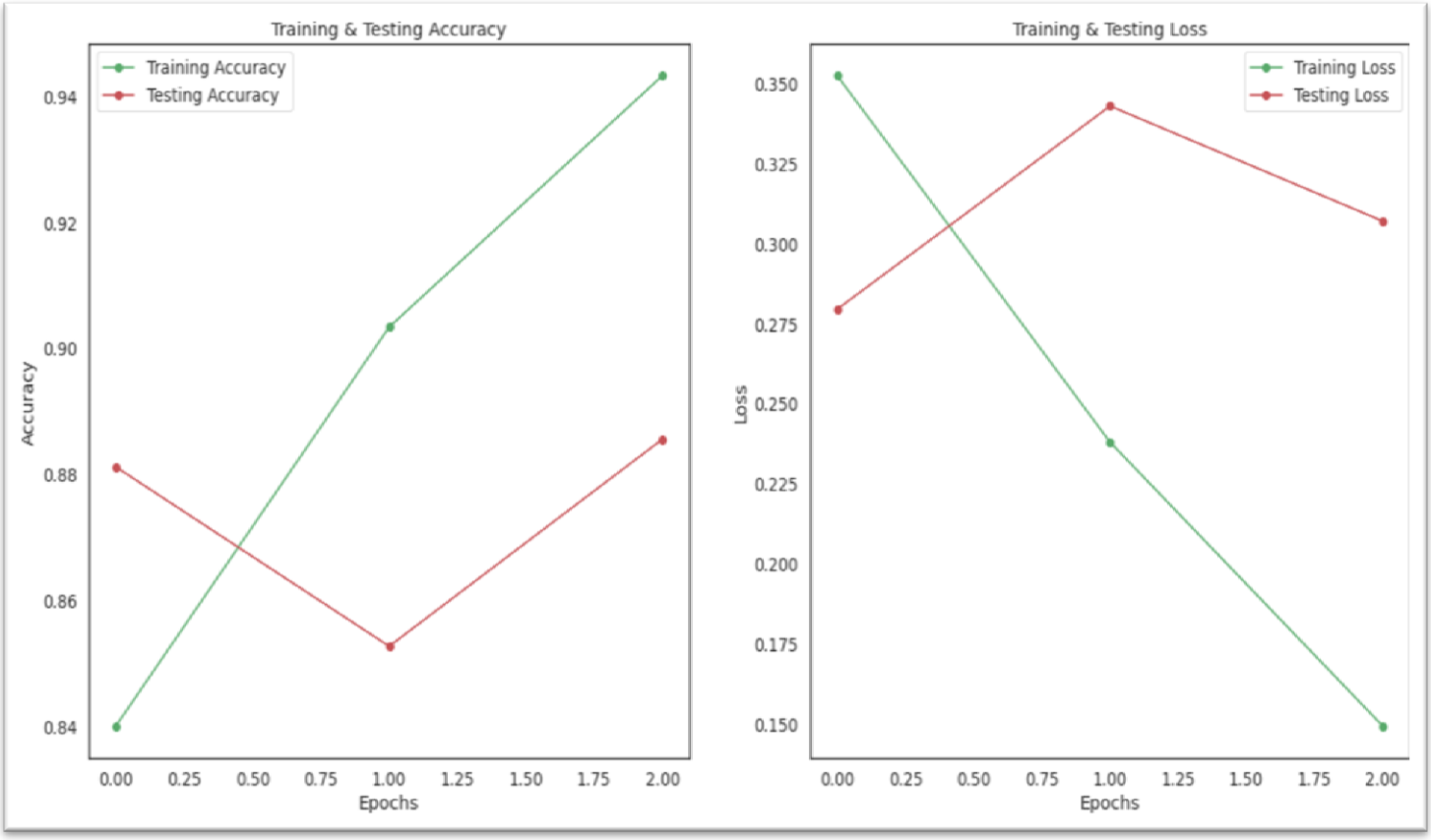
Nous devons donc prédire le nombre d'avis positifs et négatifs à l'aide d'algorithmes de classification ou d'apprentissage en profondeur. Ici, nous allons donc utiliser BERT et l'entraîner à classer correctement les avis comme positifs/négatifs, avec 3 nombres d'époques d'apprentissages et les résultats comme suit :

```
Epoch 1/3
1172/1172 [==============================] - 1018s 869ms/step - loss: 0.3522 -
accuracy: 0.8400 - val_loss: 0.2794 - val_accuracy: 0.8811
Epoch 2/3
1172/1172 [==============================] - 1016s 867ms/step - loss: 0.2379 -
accuracy: 0.9033 - val_loss: 0.3428 - val_accuracy: 0.8526
Epoch 3/3
1172/1172
[==============================] - 1015s 866ms/step - loss: 0.1494 - accuracy: 0.9433
- val_loss: 0.3069 - val_accuracy: 0.8855
```

➢ Analyse Après L'entraînement Du Modèle

Nous devons diviser notre jeu de données en un données pour formation (train) et données pour validation (val) en utilisant la fonction train_test_split de package **Tensorflow**.
Après la phase d'entrainement la phase de test nous avons utilisé 20 % des données de notre dataset et nous évaluons le modèle sur les données de test via evaluate() et nous
Obtiendrons les résultats de précision et la perte de notre modèle suivant :

391/391 [==============================] - 99s 253ms/step - loss: 0.3069
- Accuracy: 0.8855



La précision du modèle sur les données de test est : 88.551 %

Figure 3.1 Performance du modèle basé sur BERT (Distilber-bert-uncase) apprentissage et Test avec 3 époques

➢ Analyse & Discussion des Résultats

Nous remarquons d'après les graphes obtenues que les fonctions de perte commence à 0.3522 pendant l'apprentissage et 0.2794 pendant le test à l'époque 1 dans le graphe de la fonction de perte ensuite la fonction de perte pour la formation (en couleur verte dans la figure 3.1) descend

pendant les deux autres époques jusqu'à 0.237 et 0.149 et l'autre fonction de perte de test (en rouge dans la figure 3.1) augmente a 0.342 à l'époque 2 ensuite descend à 0.306 à la fin de l'époque 3. Pour la fonction de précision commence à 0.8400 pour celle de d'apprentissage (en couleur verte figure 3.1) et augmente à l'époque 2 et 3 à 0.903 et 0.943 consécutivement. La fonction de précision de test commence à 0.8811 (en couleur rouge en figure 3.1) et descend jusqu'à 0.852 après 2 époques d'apprentissage et remonte au 0.885 à l'époque 3.

les résultats sont comme suit dans le tableau 3.2 :

Nbre époques = 3	**Fonction Précision**	**Fonction Perte**
Apprentissage	0.840 à 0.9433	0.352 à 0.1494
Test	0.881 à 0.8855	0.279 à 0.3069

Tableau 3.2 Performance du modèle Distilber-bert-uncase pour 3 époques

Dans la phase d'apprentissage en remarque que la précision de formation du modèle augment progressivement et atteint un score très satisfaisant (0.943) mais la fonction précision pour le test diminue après deux époque ensuite remonte jusqu'à 0.885 parce que le système commence à apprendre après certain époque et la quantité de données utilisé pour la phase d'apprentissage et beaucoup plus volumineuses (80% des données) et celle pour le test et a 20%.

Pour la fonction de perte de formation en remarque qu'elle diminue progressivement pendant toute la phase d'apprentissage jusqu'à 0.149 et celle pour le test augmente jusqu'à 2 époque ou le système commence à apprendre et a diminué la perte d'information et redescend jusqu'à 0.306.

Nous pouvons pré conclus que le modèle (avec trois époques de formation) très performant pendant la phase d'apprentissage (94%) et moins performant pendant la phase de test (88%).

3.2.2 Performance du modèle Basé sur l'analyse des sentiments et BERT(Distilber) et le dataset IMDB (10 époques d'apprentissage)

Dans la deuxième expérimentation nous avons entrainé notre modèle de classification binaire de sentiment sur l'ensemble de jeu de données IMDB contient 50 000 critiques de films pour l'analyse de texte et avec les mêmes paramètres que la première expérimentation. Il s'agit d'un ensemble de données pour la classification des sentiments binaires contenant beaucoup de données. Il se compose d'un ensemble de 25 000 critiques de films très polaires pour la formation et de 25 000 pour les tests.

L'apprentissage se fait en appelons la fonction **fit(),** qui entraînera le modèle en découpant les données en "lots" de taille **batch size de 32** , et en itérant à plusieurs reprises sur l'ensemble de données pour un nombre donné de **époque = 10.** Linstruction concernée est :

```
history = model.fit(x_train,y_train,batch_size = 32
                    ,validation_data=(x_test,y_test),epochs = 10)
```

L'objet renvoyé **history** contient un enregistrement des valeurs de perte et des valeurs métriques pendant l'entraînement.

Nous devons donc prédire le nombre d'avis positifs et négatifs à l'aide d'algorithmes de classification et d'apprentissage en profondeur. Ici, nous allons donc utiliser BERT et l'entraîner à classer correctement les avis comme positifs/négatifs, avec 10 nombres d'époques d'apprentissages et les résultats comme suit :

Epoch 1/10
1172/1172 [==========] - 1015s 866ms/step - loss: 0.3479 - accuracy: 0.8450 - val_loss: 0.2788 - val_accuracy: 0.8817
Epoch 2/10
1172/1172 [==========] - 1014s 865ms/step - loss: 0.2327 - accuracy: 0.9046 - val_loss: 0.2732 - val_accuracy: 0.8896
Epoch 3/10
1172/1172 [==========] - 1014s 865ms/step - loss: 0.1465 - accuracy: 0.9443 - val_loss: 0.3241 - val_accuracy: 0.8802
Epoch 4/10
1172/1172 [==========] - 1014s 865ms/step - loss: 0.0800 - accuracy: 0.9709 - val_loss: 0.4310 - val_accuracy: 0.8838
Epoch 5/10
1172/1172 [==========] - 1014s 865ms/step - loss: 0.0432 - accuracy: 0.9846 - val_loss: 0.4658 - val_accuracy: 0.8854
Epoch 6/10
1172/1172 [==========] - 1014s 865ms/step - loss: 0.0321 - accuracy: 0.9889 - val_loss: 0.5150 - val_accuracy: 0.8839
Epoch 7/10
1172/1172 [==========] - 1014s 865ms/step - loss: 0.0235 - accuracy: 0.9915 - val_loss: 0.5274 - val_accuracy: 0.8837
Epoch 8/10
1172/1172 [==========] - 1013s 864ms/step - loss: 0.0212 - accuracy: 0.9926 - val_loss: 0.5569 - val_accuracy: 0.8885
Epoch 9/10
1172/1172 [==========] - 1013s 864ms/step - loss: 0.0180 - accuracy: 0.9939 - val_loss: 0.6014 - val_accuracy: 0.8851
Epoch 10/10

1172/1172 [==========] - 1013s 864ms/step - loss: 0.0149 - accuracy: 0.9941 - val_loss: 0.5850 - val_accuracy: 0.8860

391/391 [==========] - 98s 251ms/step - loss: 0.5850 - accuracy: 0.8860La précision du modèle après Test des données est : 88.59999775886536

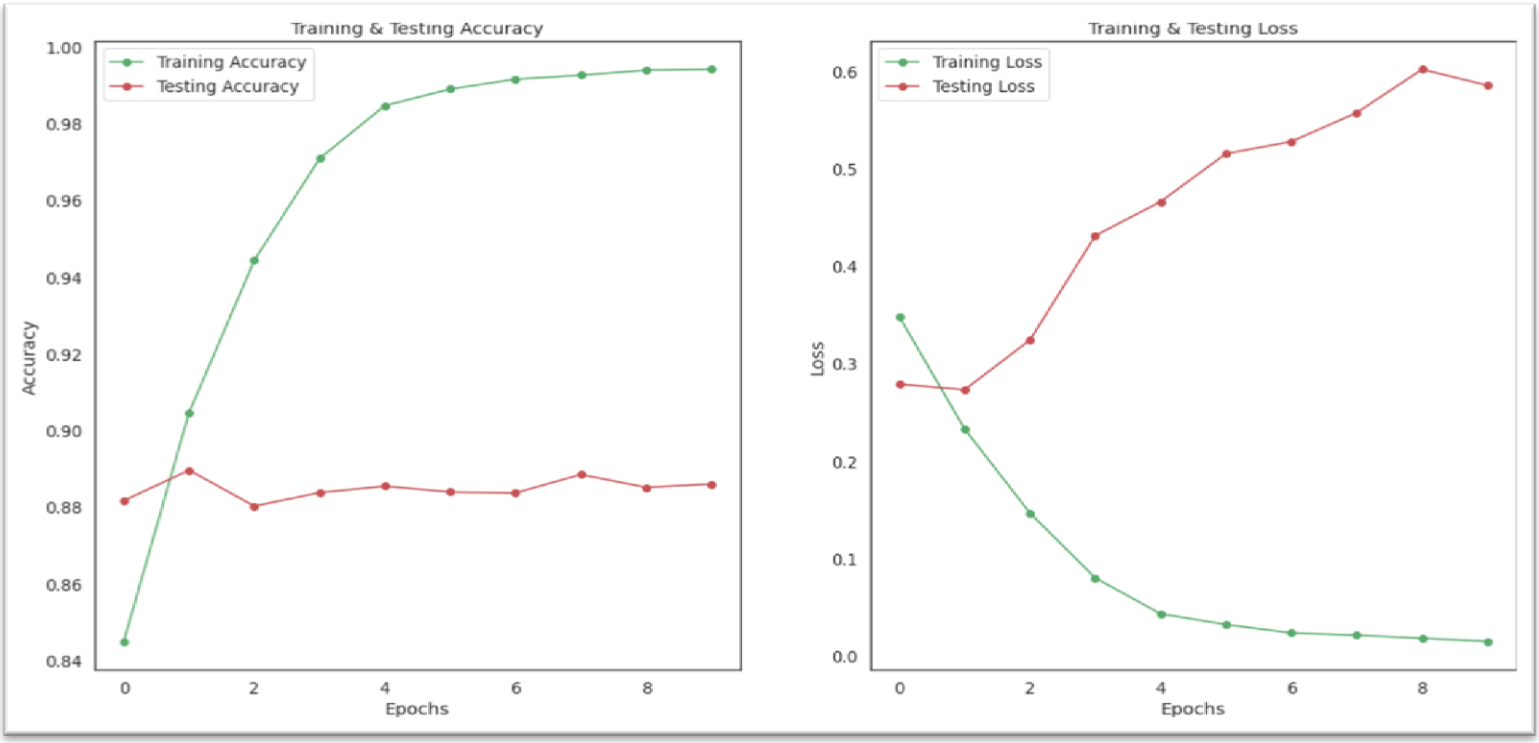


Figure 3.2 Performance du modèle basé sur BERT (Distilber-bert-uncase) apprentissage et test avec 10 époques

- **Analyse & Discussion des Résultats**

Nous remarquons d'après les graphes obtenues que les fonctions de perte commence à 0.3479 pendant l'apprentissage et 0.2788 pendant le test à l'époque 1 dans le graphe de la fonction de perte, ensuite la fonction de perte pour la formation (en couleur verte dans la figure 3.2) diminue progressivement pendant toutes les phases d'époques jusqu'à 0.0149 et l'autre fonction de perte de test (en rouge dans la figure 3.2) diminue pendant deux époques à 0.2732 à l'époque 2 ensuite commence a remonté à l'époque 9 à 0.601 et redescend à l'époque 10 à 0.0.5850 à la fin de l'époque 10. Pour la fonction de précision commence à 0.8450 pour celle de d'apprentissage (en couleur verte figure 3.2) et augmente progressivement jusqu'à l'époque 7 ou elle devient presque stable à 0.991 et à l'époque 10 à 0.994. La fonction de précision de test commence à 0.8811 (en couleur rouge en figure 3.2) et monte à 0.889 à l'époque 2 et redescend à 0.880 à l'époque 3 et remonte et redescend depuis l'époque 4 jusqu'à la fin de l'époque 10 à 0.886.

Les résultats sont comme suit dans le tableau 3.3 :

Nbre époques= 10	Fonction Précision	Fonction Perte
Apprentissage	**0.845 à 0.994**	**0.347 à 0.149**
Test	**0.881 à 0.886**	**0.278 à 0.585**

Tableau 3.3 Performance du modèle Distilber-bert-uncase avec 10 époques

Nous pouvons conclure que pendant cette phase d'expérience le modèle et très performant et répond bien à la phase de formation avec une performance de 99 % et une perte minimum de 0.149.

Mais pour la phase test nous pouvons dire que le modèle a progressé dans la précision de 0.01 et aussi la perte à augmenter de presque le double que la première expérience (de 0.278 à 0.585) a cause de la répartition des données et quelque réglage des hyper paramètres.

3.2.3 Performance du modèle Basé sur l'analyse des sentiments et BERT (basic model uncase) et le Dataset IMDB (3 et 10 époques d'apprentissage)

Nous continuant avec le même dataset mais avec le modèle Bert basic uncase ,3 et 10 époques et les résultats sont comme suit :

- **Avec 3 époques**

Epoch 1/2
2814/2814 [==========] - 1399s 497ms/step - loss: 0.2408 - accuracy: 0.8989 - val_loss: 0.3109 - val_accuracy: 0.8910

Epoch 2/2
2814/2814 [==========] - 1392s 495ms/step - loss: 0.0682 - accuracy: 0.9759 - val_loss: 0.5317 - val_accuracy: 0.8792

Epoch 3/3
2814/2814 [==========] - 1397s 496ms/step - loss: 0.0322 - accuracy: 0.9890 - val_loss: 0.4951 - val_accuracy: 0.8942

- **Avec 10 époques :**

```
Epoch 1/10
2814/2814 [==========] - 1400s 497ms/step - loss: 0.2456
- accuracy: 0.8960 - val_loss: 0.3409 - val_accuracy: 0.8752
Epoch 2/10
2814/2814 [==========] - 1397s 496ms/step - loss: 0.0706
- accuracy: 0.9755 - val_loss: 0.4346 - val_accuracy: 0.8892
Epoch 3/10
2814/2814 [==========] - 1397s 496ms/step - loss: 0.0322
- accuracy: 0.9890 - val_loss: 0.4951 - val_accuracy: 0.8942
Epoch 4/10
2814/2814 [==========] - 1397s 497ms/step - loss: 0.0208
- accuracy: 0.9933 - val_loss: 0.5692 - val_accuracy: 0.8924
Epoch 5/10
2814/2814 [==========] - 1396s 496ms/step - loss: 0.0177
- accuracy: 0.9941 - val_loss: 0.5296 - val_accuracy: 0.8922
Epoch 6/10
2814/2814 [==========] - 1396s 496ms/step - loss: 0.0140
- accuracy: 0.9952 - val_loss: 0.5393 - val_accuracy: 0.8928
Epoch 7/10
```

```
2814/2814 [==============================] - 1396s 496ms/step - loss: 0.0117
- accuracy: 0.9960 - val_loss: 0.6013 - val_accuracy: 0.8870
Epoch 8/10
2814/2814 [==============================] - 1397s 496ms/step - loss: 0.0110
- accuracy: 0.9966 - val_loss: 0.6312 - val_accuracy: 0.8942
Epoch 9/10
2814/2814 [==============================] - 1396s 496ms/step - loss: 0.0094
- accuracy: 0.9969 - val_loss: 0.5745 - val_accuracy: 0.8910
Epoch 10/10
 2814/2814 [==============================] - 1395s 496ms/step - loss: 0.0086
    - accuracy: 0.9970 - val_loss: 0.6650 - val_accuracy: 0.8902
```

3.2.4 Performance du modèle Basé sur l'analyse des sentiments et BERT (basic model uncase) et le Dataset Avis Google Play Store (reviews.csv) (3 époques d'apprentissage)

Pour le deuxième modèle BERT (basic modèle) nous avons utilisez des données qui contiennent plus de 12 000 formats (reviews.csv) avis de différentes applications de magasin d'applications par de vrais utilisateurs.

Nous créons un *DataLoader* qui fournit un moyen de créer un ensemble de données à partir des données déjà chargées en mémoire. Il accepte les données sous forme de tableau et de tenseur. Ensuite, nous allons créer un itérateur pour notre ensemble de données en utilisant la classe torch *DataLoader*. Cela aidera à économiser de la mémoire pendant l'entraînement et à augmenter la vitesse d'entraînement.

Pour la procédure d'apprentissage nous utiliserons l'optimiseur **AdamW** fourni par Hugging Face. Les paramètres du modèle BERT sont :

- Batch size: 16
- Learning rate (Adam)=2e-5
- Number of epochs: 5,10

Nous présentons ici, les résultats de performance du modèle sur l'ensemble de données **reviews.csv** qui représente les avis sur Google store avec 3 nombres d'époques d'apprentissage

Epoch 1/3

Train loss 0.6911430652856827 accuracy 0.7239895958383353
Val loss 0.5896255040093313 accuracy 0.7718174539631705
Epoch 2/3

Train loss 0.48559568008184434 accuracy 0.8104241696678672
Val loss 0.6259272545576096 accuracy 0.7493995196156925
Epoch 3/3

Train loss 0.34605593435019255 accuracy 0.87374949979992
Val loss 0.7471799006379103 accuracy 0.7461969575660528

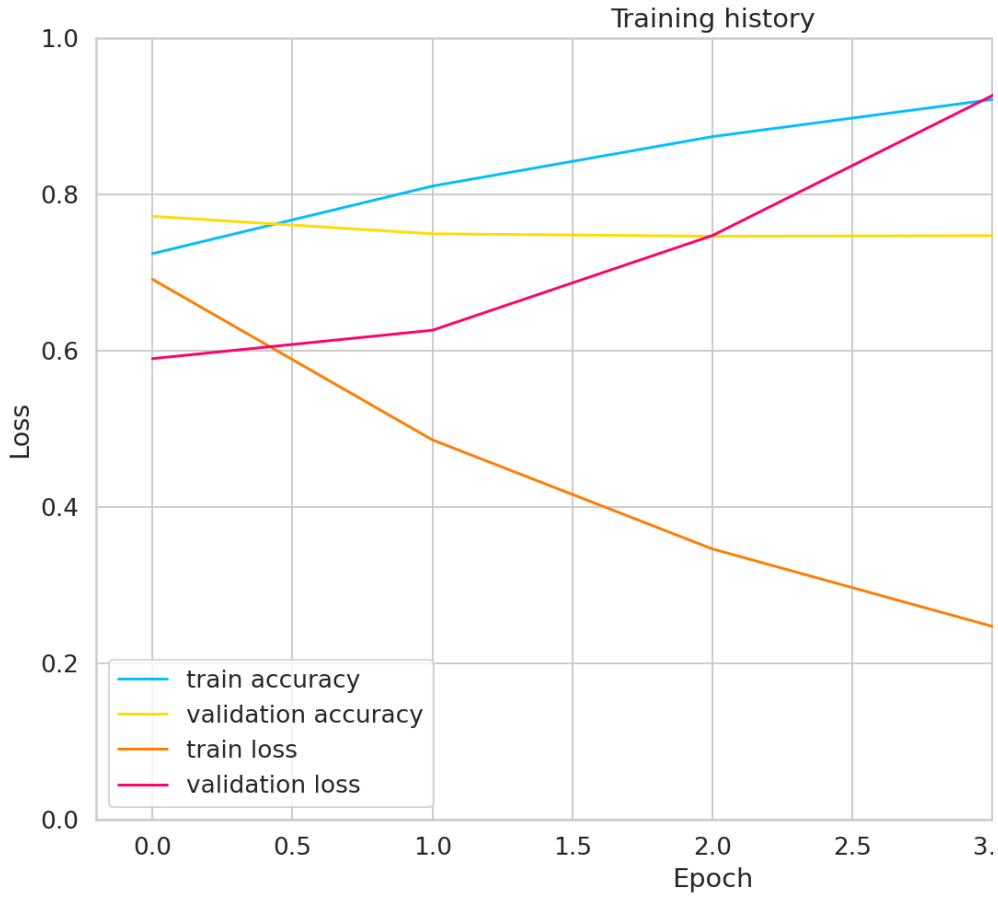


Figure 3.3 Performance du modèle basé sur BERT (basic model uncase) apprentissage et test avec 3 époques

- **Discussion des Résultats**

Nous remarquons d'après les graphes obtenues que nous avons quatre fonctions deux pour la précision (couleur Bleu pour l'apprentissage, jaune pour validation) et deux pour la perte (couleur orangé perte apprentissage et couleur rouge perte de validation) les fonctions de perte commence à 0.691 pendant l'apprentissage et 0.589 pendant le test à l'époque 1 ensuite la fonction de perte pour la formation diminue progressivement pendant toutes les phases de formation d'époques jusqu'à 0.184 et l'autre fonction de perte de validation augmente pendant tous les époques à 1.092.

Pour la fonction de précision commence à 0.723 pour celle de l'apprentissage et augmente progressivement jusqu'à l'époque 5 à 0.945. La fonction de précision de test commence à 0.771 et diminue à 0.749 à l'époque 2 et reste presque stable jusqu'à la fin de formation à 0.747.

Les résultats sont comme suit dans le tableau 3.4 :

Nbre époques= 3	**Fonction Précision**	**Fonction Perte**
Apprentissage	**0.723 à 0.875**	**0.691 à 0.346**
Test	**0.771 à 0.747**	**0.589 à 1.747**

Tableau 3.4 performance du modèle (basic-uncase) avec 3 époques

Nous pouvons conclure que pendant cette phase d'expérience le modèle et très performant et répond bien à la phase de formation avec une performance de 94.5 % et une perte minimum de 0.118.

Mais pour la phase test nous pouvons dire que le modèle est presque stable avec une précision de 75% et aussi une perte qui augmenter de presque le double que le premier démarrage de l'expérience (de 0.589 à 1.091) cela peut être du a la répartition des classe de sentiment ou il peut distinguer les sentiments positifs et les sentiments négatifs mais il a du mal avec les sentiments neutres

3.2.5 Performance du modèle Basé sur l'analyse des sentiments et BERT (basic model uncase) et le Dataset Avis Google Play Store (reviews.csv) (10 époques d'apprentissage)

Pour le même model BERT précédant nous gardons les mêmes paramètres mais nous augmentons le nombre d'époques d'apprentissage on espère obtenir de bonne résultat que celle de 5 époques. Nous présentons les résultats obtenues de performance du modèle sur l'ensemble de données **reviews.csv** qui représente les avis sur Google store avec 10 nombres d'époque d'apprentissage

```
Epoch 1/10
----------
Train loss 0.6917546150445938 accuracy 0.7264905962384954
Val   loss 0.5997012070839918 accuracy 0.7590072057646117
Epoch 2/10
----------
Train loss 0.49289548637866976 accuracy 0.8011204481792717
Val   loss 0.6435634686222559 accuracy 0.7493995196156925
Epoch 3/10
----------
Train loss 0.34903768949210645 accuracy 0.8738495398159264
Val   loss 0.7344782854182811 accuracy 0.7341873498799039
Epoch 4/10
----------
Train loss 0.2586550265178084 accuracy 0.9182673069227691
Val   loss 0.9526580310886419 accuracy 0.733386709367494
Epoch 5/10
----------
Train loss 0.19132201708387583 accuracy 0.9427771108443378
Val   loss 1.1719827731198902 accuracy 0.7237790232185748
Epoch 6/10
----------
Train loss 0.14482437502853573 accuracy 0.9610844337735095
Val   loss 1.3333565023126481 accuracy 0.732586068855084
```

Epoch 7/10

Train loss 0.1079322527319193 accuracy 0.9700880352140857
Val loss 1.6314030644259876 accuracy 0.7237790232185748
Epoch 8/10

Train loss 0.09417999772601761 accuracy 0.9752901160464187
Val loss 1.6668235789748687 accuracy 0.7253803042433946
Epoch 9/10

Train loss 0.07481908812741749 accuracy 0.9804921968787516
Val loss 1.7300943883159492 accuracy 0.732586068855084
Epoch 10/10

Train loss 0.06699888246003538 accuracy 0.9819927971188476
Val loss 1.804545645095125 accuracy 0.72457966373098470

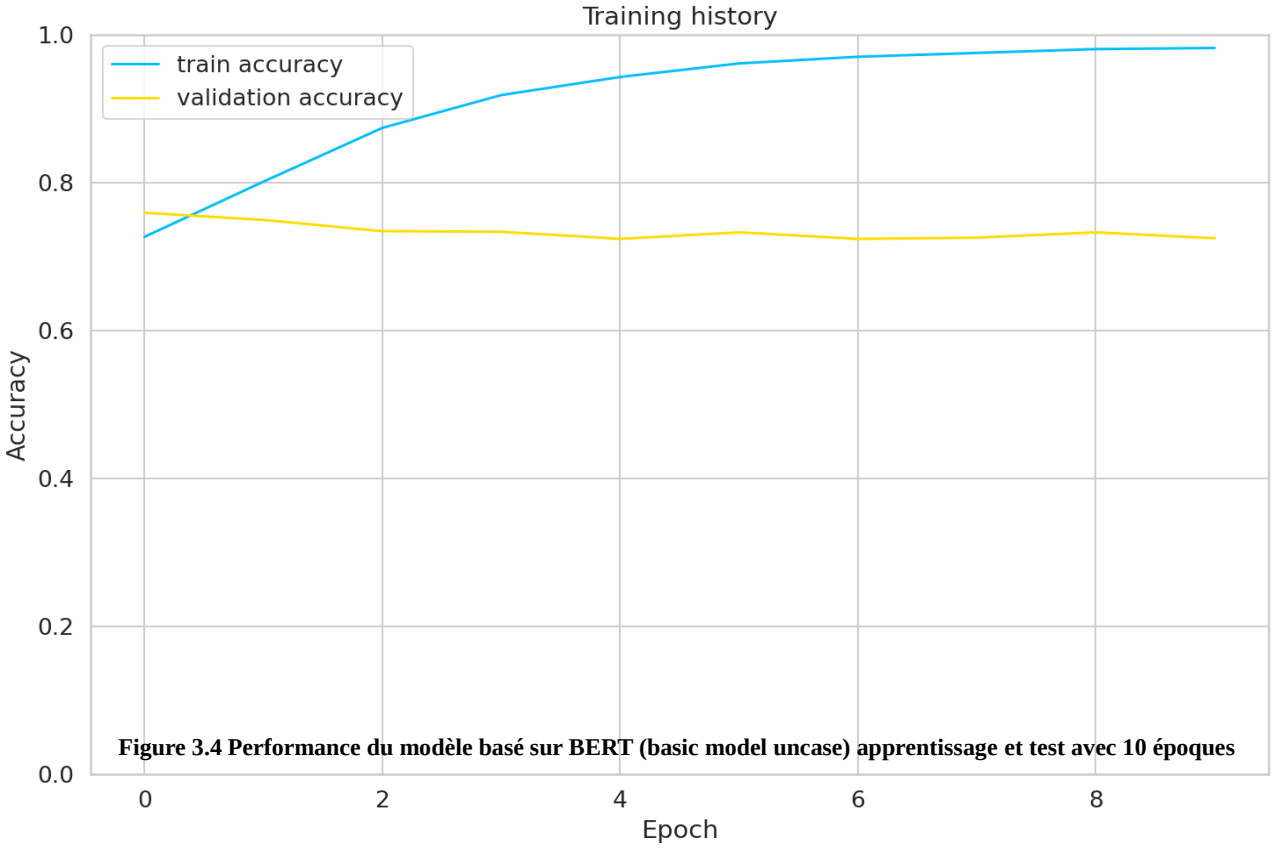


Figure 3.4 Performance du modèle basé sur BERT (basic model uncase) apprentissage et test avec 10 époques

- **Discussion des Résultats**

Nous remarquons d'après les graphes obtenues que nous avons deux fonctions pour la précision (couleur bleu pour l'apprentissage, jaune pour validation).

D'après les étapes d'apprentissages les fonctions de perte commence à 0.691(train_loss, val_loss) pendant l'apprentissage et 0.599 pendant le test à l'époque 1 ensuite la fonction de perte pour la formation diminue progressivement pendant toutes les phases de formation

d'époques jusqu'à 0.074 et l'autre fonction de perte de validation augmente pendant tous les époques à 1.804.

Pour la fonction de précision commence à 0.727 pour celle de l'apprentissage et augmente progressivement jusqu'à l'époque 10 à 0.980. La fonction de précision de test commence à 0.759 et diminue à 0.749 à l'époque 2 et reste presque stable jusqu'à la fin de formation à 0.724.

Nous remarquons d'après le graphe obtenu que la fonction de perte à commencer à 0.691

Pendant l'apprentissage et la fonction de précision à 0.726 et pendant la phase test la fonction perte à 0.599 et la fonction précision à 0.759, après 10 époques d'apprentissage et de validation les résultats sont comme suit dans le tableau 3.5 :

Nbre époques= 10	Fonction Précision	Fonction Perte
Apprentissage	**0.726 à 0.980**	**0.691 à 0.074**
Test	**0.759 à 0.732**	**0.599 à 1.804**

Tableau 3.5 performance des modèles (basic-uncase) avec 10 époques

Nous pouvons conclure que pendant cette phase d'expérience le modèle et très performant et répond bien à la phase de formation avec une performance de 98 % et une perte minimum de 0.074. Mais pour la phase test nous pouvons dire que le modèle est presque stable avec une précision de 73% et aussi une perte qui augmenter de presque le double que le premier démarrage de l'expérience (de 0.599 à 1.804) cela peut être du a la répartition des classe de sentiment ou il peut distinguer les sentiments positifs et les sentiments négatifs mais il a du mal avec les sentiments neutres.

3.2.6 Performance du model LSTM (Long Short Term Memory) (dataset IMDB movie avec 3 et 10 époques)

Dans cette expérience nous voulons tester et voir ce modèle de langage et comparer ces résultats avec celle obtenue aux précédant expériences (BERT modèle), nous avons utilisez L'incorporation de mots qui est l'une des représentations populaires du vocabulaire de document. Il est capable de capturer le contexte d'un mot dans un document, la similitude sémantique et syntaxique, la relation avec d'autres mots, ...etc

Fondamentalement, il s'agit d'une représentation vectorielle de mots utilisés pour d'autres applications de traitement du langage naturel.

Nous téléchargeons l'intégration pré-formée et l'utilisons dans notre modèle.

```
!wget http://nlp.stanford.edu/data/glove.6B.zip
!unzip glove.6B.zip
```

L'incorporation de mots pré-entraînés comme **GloVe(Global Vectors for Word Representation** est un algorithme d'apprentissage non supervisé pour obtenir des représentations

vectorielles pour les mots. La formation est effectuée sur des statistiques globales agrégées de cooccurrence mot-mot à partir d'un corpus, et les représentations résultantes présentent des sous-structures linéaires intéressantes de l'espace vectoriel des mots**)** [W5] donne plus d'informations sur un mot qui peut être utilisé pour la classification.

Les réseaux de neurones récurrents RNN peuvent gérer une séquence de données et apprendre un modèle de séquence d'entrée pour donner une séquence ou une valeur scalaire en sortie. Dans notre cas, le réseau de neurones génère une prédiction de valeur scalaire (1positifs ,0 negatif).

Pour l'architecture du modèle, nous utilisons

1**) Couche d'intégration :** Génère un vecteur d'intégration pour chaque séquence d'entrée.

2) Couche Conv1D : Son utilisation pour convoluer les données en vecteurs de caractéristiques plus petits.

3) LSTM (Long Short Term Memory) : c'est une variante de RNN (Recurrent Neural Network) qui a une cellule d'état de mémoire pour apprendre le contexte des mots qui sont plus loin dans le texte pour porter une signification contextuelle plutôt que juste des mots voisins comme dans le cas de RNN.

4) Dense - Couches entièrement connectées pour la classification

Voici les principaux paramètres du modèle et celle de données :

TRAIN_SIZE = 0.8
MAX_NB_WORDS = 100000
MAX_SEQUENCE_LENGTH = 30
EMBEDDING_DIM = 300
LR = 1e-3
BATCH_SIZE = 1024
EPOCHS = 15

le dataset utiliser et un ensemble de tweet telechargé depuis le site kaggle c'est des Tweets concernent de véritables catastrophe d'un ensemble de données de 10 000 tweets classés manuellement.

Les résultats après 10 époques d'apprentissage sont comme suit :

----Training the network----
Epoch 1/10
992/992 - 721s - loss: 0.4738 - accuracy: 0.7608 - val_loss: 0.3001 - val_accuracy: 0.8756
Epoch 2/10
992/992 - 708s - loss: 0.2682 - accuracy: 0.8929 - val_loss: 0.2469 - val_accuracy: 0.9007
Epoch 3/10
992/992 - 708s - loss: 0.1955 - accuracy: 0.9261 - val_loss: 0.2539 - val_accuracy: 0.9042
Epoch 4/10
992/992 - 707s - loss: 0.1551 - accuracy: 0.9413 - val_loss: 0.2655 - val_accuracy: 0.9018
Epoch 5/10
992/992 - 705s - loss: 0.1475 - accuracy: 0.9453 - val_loss: 0.3339 - val_accuracy: 0.8652
Epoch 6/10
992/992 - 705s - loss: 0.1164 - accuracy: 0.9575 - val_loss: 0.3317 - val_accuracy: 0.8958

Epoch 7/10
992/992 - 707s - loss: 0.0844 - accuracy: 0.9704 - val_loss: 0.3666 - val_accuracy: 0.8958
Epoch 8/10
992/992 - 706s - loss: 0.0645 - accuracy: 0.9777 - val_loss: 0.4036 - val_accuracy: 0.8964
Epoch 9/10
992/992 - 707s - loss: 0.0568 - accuracy: 0.9806 - val_loss: 0.3961 - val_accuracy: 0.8934
Epoch 10/10
992/992 - 706s - loss: 0.0470 - accuracy: 0.9837 - val_loss: 0.3941 - val_accuracy: 0.8925
992/992 [==============================] - 177s 179ms/step - loss: 0.0133 - accuracy: 0.9973
Train score: [0.013264034874737263, 0.9973213076591492]
248/248 [==============================] - 44s 17

Validation score : [0.39408978819847107, 0.8924744725227356]

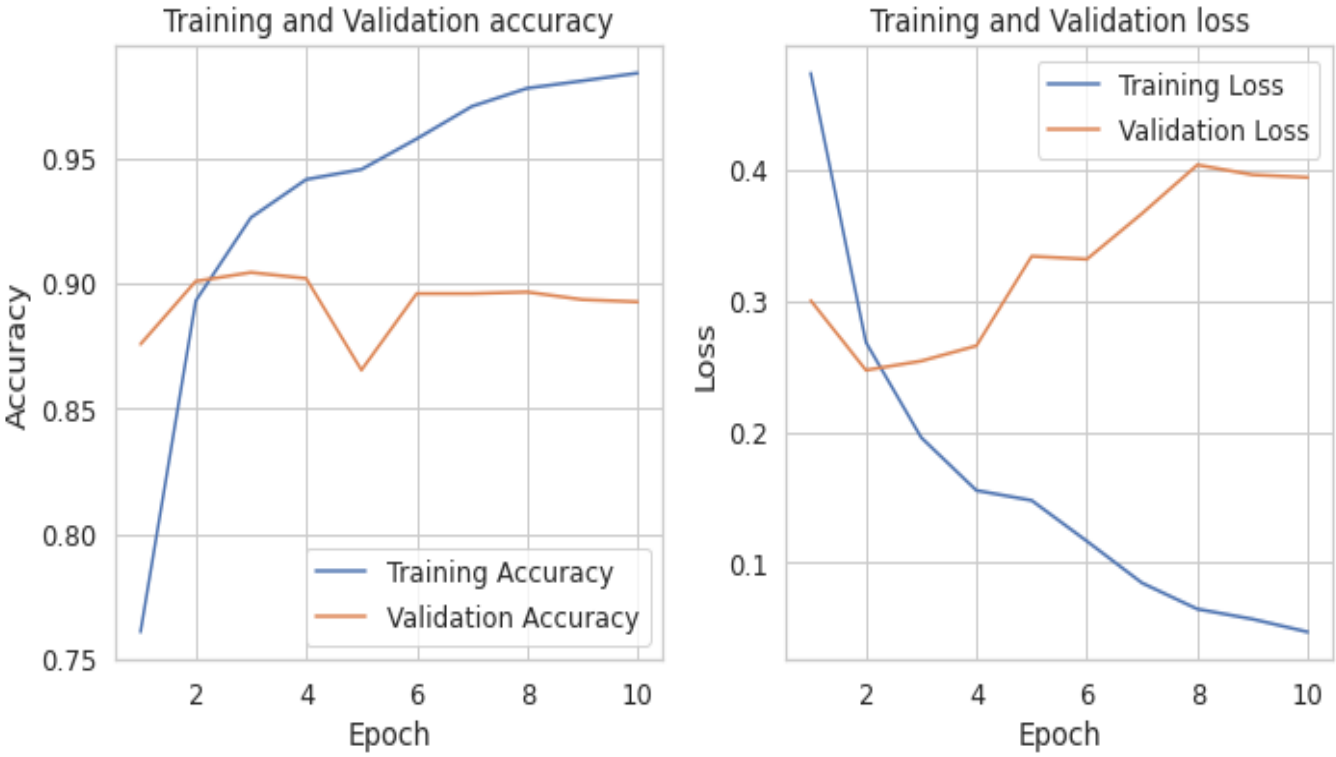


Figure 3.5 Performance du modèle LSTM après 10 époques d'apprentissage

- **Discussion Des Résultats**

Nous remarquons d'après les graphes obtenues que les fonctions de perte commence à 0.4738 pendant l'apprentissage et 0.3001 pendant le test à l'époque 1 dans le graphe ensuite la fonction de perte pour la formation (en couleur bleu dans la figure 3.5) diminue progressivement pendant toutes les phases d'époques jusqu'à 0.047 et l'autre fonction de perte de test (en rouge cl dans la figure 3.5) diminue pendant 4 époques à 0.265 ensuite commence a remonté à l'époque 10 à 0.394 jusqu'à la fin de l'époque 15 à 0.5302.

Pour la fonction de précision de test commence à 0.875 (en couleur bleu figure 3.5) et augmente progressivement jusqu'à l'époque 4 arrivée à 0.901 est reste presque stable pendant entre(0.865 et 0.892) jusqu'à la fin de formation . La fonction de précision de formation commence à 0.760 (en couleur rouge foncé en figure 3.5) et monte à progressivement jusqu'à la fin de l'époque 15 à 0.997.

Les résultats sont comme suit dans le tableau 3.6 :

Nbre époques = 15	**Fonction Précision**	**Fonction Perte**
Apprentissage	0.554 à 0.912	0.673 à 0.221
Test	0.757 à 0.802	0.539 à 0.530

Tableau 3.6 performance du modèle LSTM avec 10 époques

Nous pouvons conclure que pendant cette phase d'expérience le modèle LSTM et très performant et répond bien à la phase de formation avec une performance de plus 91 % et une perte qui diminue arrivée à 0.221.

Mais pour la phase test nous pouvons dire que le modèle est presque stable avec une précision de 80 % et mais une perte qui à diminue à l'époque 9 à 0.435 et depuis le 1 époque augmenter en arrivant à la première valeur du démarrage de l'expérience (de **0.539 à 0.530**).

3.2.7 Performance du model LSTM (Long Short Term Memory) (dataset pIMDB avec 3 époques)

Voici les résultats avec 3 époques pendant l'apprentissage **:**

Epoch 1/3

992/992 - 721s - loss: 0.4738 - accuracy: 0.7608 - val_loss: 0.3001 - val_accuracy: 0.8756

Epoch 2/3

992/992 - 708s - loss: 0.2682 - accuracy: 0.8929 - val_loss: 0.2469 - val_accuracy: 0.9007

Epoch 3/3

992/992 - 708s - loss: 0.1955 - accuracy: 0.9261 - val_loss: 0.2539 - val_accuracy: 0.9042

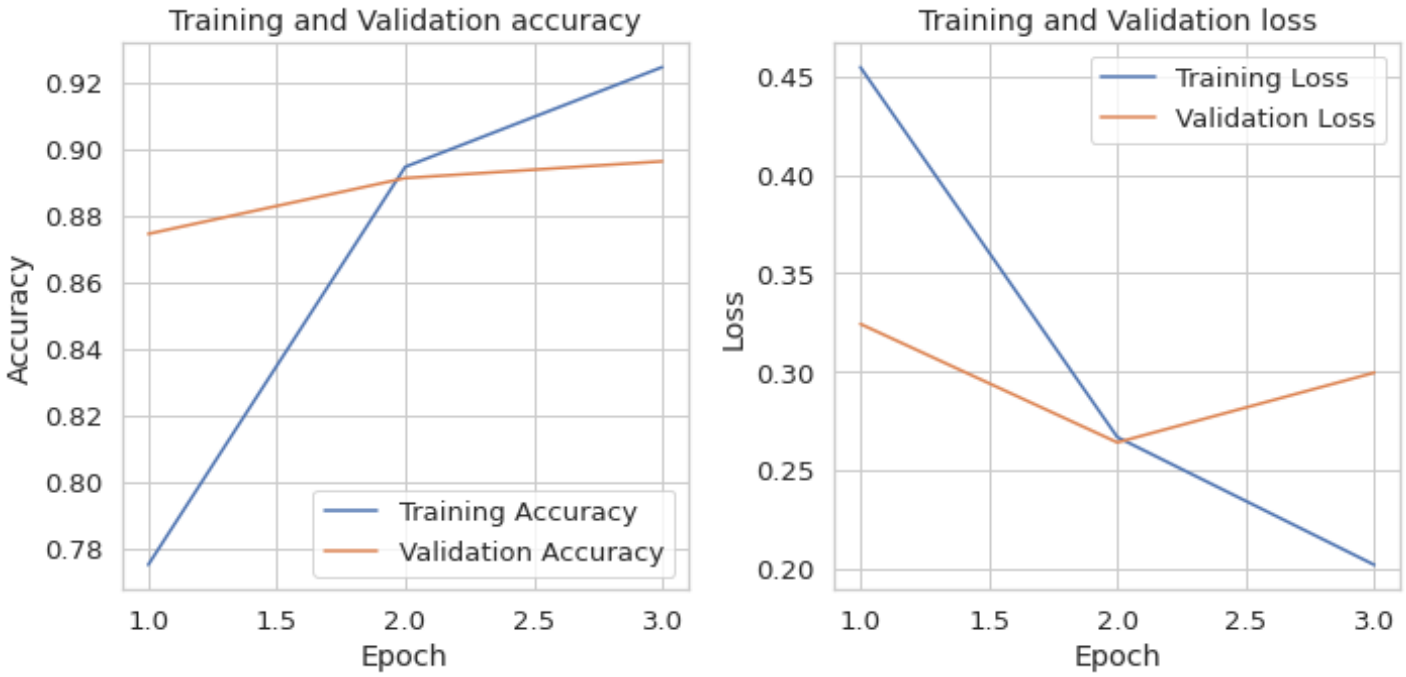


Figure 3.6 Performance du modèle LSTM après 3 époques d'apprentissage

3.2.8 Performance du model LSTM (Long Short Term Memory) (dataset google playstore avec 3 et 10 époques)

Dans cette expérience nous voulons tester et voir ce modèle de langage et comparer ces résultats avec celle obtenue aux précédant expériences (BERT modèle), nous avons utilisez le même dataset avec 3 époques et les résultats comme suit :

- **Résultats modèle LSTM aprés 3 époques**

Epoch 1/3

29941/29941 [==============================] - 313s 10ms/step - loss: 0.5355 - accuracy: 0.7893 - val_loss: 0.3026 - val_accuracy: 0.9064

Epoch 2/3

29941/29941 [==============================] - 314s 10ms/step - loss: 0.2368 - accuracy: 0.9263 - val_loss: 0.2400 - val_accuracy: 0.9253

Epoch 3/3

29941/29941 [==============================] - 314s 10ms/step - loss: 0.1588 - accuracy: 0.9528 - val_loss: 0.2376 - val_accuracy: 0.9248

- **Résultats modèle LSTM aprés 10 époques**

Train on 29941 samples, validate on 7486 samples
Epoch 1/10
29941/29941 [==============================] - 646s 22ms/step - loss: 0.3933 - accuracy: 0.8628 - val_loss: 0.2750 - val_accuracy: 0.9160
Epoch 2/10
/opt/conda/lib/python3.6/site-packages/keras/callbacks/callbacks.py:846: RuntimeWarning: Early stopping conditioned on metric `value_loss` which is not available. Available metrics are: val_loss,val_accuracy,loss,accuracy
(self.monitor, ','.join(list(logs.keys()))), RuntimeWarning
29941/29941 [==============================] - 669s 22ms/step - loss: 0.2159 - accuracy: 0.9349 - val_loss: 0.2349 - val_accuracy: 0.9279
Epoch 3/10
29941/29941 [==============================] - 684s 23ms/step - loss: 0.1522 - accuracy: 0.9549 - val_loss: 0.2630 - val_accuracy: 0.9236
Epoch 4/10
29941/29941 [==============================] - 668s 22ms/step - loss: 0.1228 - accuracy: 0.9631 - val_loss: 0.2526 - val_accuracy: 0.9247
Epoch 5/10
29941/29941 [==============================] - 653s 22ms/step - loss: 0.0998 - accuracy: 0.9709 - val_loss: 0.2707 - val_accuracy: 0.9227
Epoch 6/10
29941/29941 [==============================] - 671s 22ms/step - loss: 0.0822 - accuracy: 0.9757 - val_loss: 0.2806 - val_accuracy: 0.9271
Epoch 7/10
29941/29941 [==============================] - 663s 22ms/step - loss: 0.0708 - accuracy: 0.9791 - val_loss: 0.2972 - val_accuracy: 0.9232
Epoch 8/10
29941/29941 [==============================] - 663s 22ms/step - loss: 0.0607 - accuracy: 0.9821 - val_loss: 0.3007 - val_accuracy: 0.9259
Epoch 9/10
29941/29941 [==============================] - 681s 23ms/step - loss: 0.0590 - accuracy: 0.9823 - val_loss: 0.3120 - val_accuracy: 0.9227
Epoch 10/10
29941/29941 [==============================] - 665s 22ms/step - loss: 0.0462 - accuracy: 0.9863 - val_loss: 0.3293 - val_accuracy: 0.9256

3.3 Résultats obtenus et discussion

D'après les expériences que nous avons pu mettre en œuvre nous pouvons résumer notre travail dans le tableau 3.7 suivant :

Modèles	IMDB Dataset				Reviews Google Play Store Dataset			
	Précision par époque		Perte par Epoque		Précision par Epoque		Perte par Epoque	
	3	10	3	10	3	10	3	10
BERT Distilber	0.845	0.881	0.345	0.585	0.897	-	-	-
b) BERT c) Basic unCase	0.873	0.890	0.531	0.665	0.746	0.732	0.747	1.80
LSTM	0.904	0.892	0.253	0.402	0.924	0.925	0.237	0.329

Tableau 3.7 Analyse des résultats finals

Nous pouvons dire que les trois modèles avec le jeu de données IMDB movies donnent des bonnes précisions plus 85 % avec 90 % pour le modèle LSTM a 3 époques mais a 10 époques on peut dire que les trois modèles donne presque le même résultat plus 88% de précision. Sur le point de vue erreur le modèle LSTM donne le taux le plus faible (0.253 et 0.402) pis le modèle Distilbert BERT avec (0.345 et 0.585) et enfin le BERT basic uncase avec une erreur entre (0.531 et 0.665).

Nous pouvons conclure que plus on augmente le nombre la précision augmente sauf le modèle LSTM diminue et l'erreur augmente progressivement par tous les modèles avec d'autres réglages de paramètre de BERT et avec plus de jeux de données plus volumineux et plus d'époques de formation nous pouvons avoir de meilleur résultat avec une erreur très réduite.

Conclusion et Perspectives

Le travail présenté dans ce mémoire s'insère dans le domaine de l'élicitation des préférences dans les Systèmes de Recommandation Conversationnels en utilisant les techniques du traitement automatique du langage naturel. Plus particulièrement, notre travail sur l'élicitation des préférences vise l'analyse des sentiments en utilisant les modèles BERT. Ces nouveaux modèles sont développés récemment par le géant Google pour améliorer les requêtes, prédire les conversations et produire des textes, ce qui nous a poussés à s'intéresser à l'étude des caractéristiques et des points fort de ces modèles afin de les utiliser dans l'élicitation des préférences dans les SRC.

Pour mieux comprendre le travail présenté dans ce mémoire, nous avons d'abord introduit, de manière non exhaustive, le domaine de la recommandation conversationnelle et l'élicitation de préférences, plus précisément l'analyse de sentiments avec un état de l'art détaillé. Puis, une introduction aux notions des modèles Bert est présentée avec quelques travaux de l'état de l'art.

Ensuite, nous avons présenté notre travail qui consistait à mettre en œuvre une solution afin de détecter le sentiment (positive / négative) des avis et des critiques sur des sujet divers. Pour ce faire, nous nous sommes basés sur un algorithme d'apprentissage profond (BERT) permettant d'abord d'extraire les préférences depuis leurs sentiments présents au sein du texte. Puis, sur la base de ces dernières, une classification réalisée.

En ce qui concerne les résultats de l'analyse des sentiments basée sur le BERT, nous avons trouvé que les résultats de l'étape de classification étaient relativement acceptables. Cependant, les résultats obtenus à partir des LSTM sont plus que satisfaisantes que celles des modèles BERT.

Ainsi, en guise de perspectives, nous nous intéresserons en détails à étudier beaucoup plus les paramètres à modifier, et étudier le modèle BERT de plus près avec d'autres variantes afin de permettre aux mieux leur utilisation dans le domaine de l'élicitation des préférences. De plus, on vise d'utiliser le BERT dans le domaine de recherche d'information.

Références

A. Références Bibliographiques

[1] Zhuang, Y., & Kim, J. (2021). A BERT-Based Multi-Criteria Recommender System for Hotel Promotion Management. *Sustainability*, *13*(14), 8039.

[2] Batra, H., Punn, N. S., Sonbhadra, S. K., & Agarwal, S. (2021, September). BERT-Based Sentiment Analysis: A Software Engineering Perspective. In *International Conference on Database and Expert Systems Applications* (pp. 138-148). Springer, Cham.

[3] Selvakumar, B., & Lakshmanan, B. (2022). Sentimental analysis on user's reviews using BERT. *Materials Today: Proceedings*.

[4] Sudharsan Ravichandiran Getting Started with Google BERT

[5] Jannach, D., Manzoor, A., Cai, W., & Chen, L. (2021). A survey on conversational recommender systems. *ACM Computing Surveys (CSUR)*, *54*(5), 1-36.

[6] Gao, C., Lei, W., He, X., de Rijke, M., & Chua, T. S. (2021). Advances and challenges in conversational recommender systems: A survey. *AI Open*, *2*, 100-126.

[7] Zhang, Y., Chen, X., Ai, Q., Yang, L, et Bruce Croft, W.: Towards conversational search and recommendation: System ask, user respond. *In Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 177–186

[8] Zhang, X., Xie, Hong Hang Li et John C. S. Lui : Conversational contextual bandit: Algorithm and application, 2020.

[9] Weizenbaum , J.. ELIZA – Computer program for the study of natural language communication between man and machine. Communications. ACM, 9(1):36–45, Jan. 1966.

[10] R. S. Wallace. The Anatomy of A.L.I.C.E. In Parsing the Turing Test, pages 181–210. Springer, 2009.

[11] Radlinski, F., Craswell, N.. A theoretical framework for conversational search. In CHIIR '17, pages 117–126, 2017.

[12] Christakopoulou, K., Filip, R., et Katja Hofmann : Towards conversational recommender systems. *In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 815–824, New York, NY, USA, 2016.

[13] Bell et al., 2007

[14] Christakopoulou et al., 2018; Lei et al., 2020a, 2020b; Zou et al., 2020

[15] Zhang et al., 2018; Sun and Zhang, 2018; Ren et al., 2020.

[16] Zou et al., 2020; Zhang et al., 2018,

[17] Sun and Zhang, 2018; Lei et al., 2020a, 2020b.

[18] Li et al., 2018; Liu et al., 2020ab; Ren et al., 2020; Chen et al., 2019bb; Zhou et al., 2020a.

[19] Schnabel et al., 2018; Li et al., 2015; Gilotte et al., 2018.

[20] Li et al., 2015; Jagerman et al., 2019; Huang et al., 2020.

[21] Gilotte et al., 2018; Jagerman et al., 2019,

[22] Zhang and Balog, 2020; Sun et al., 2021.

[23] Priyogi, B. (2019, January). Preference elicitation strategy for conversational recommender system. In *Proceedings of the twelfth ACM international conference on web search and data mining* (pp. 824-825).

[24] Zhao et al., 2013; Wang et al., 2018; Christakopoulou et al., 2016; Zou et al., 2020b; Vendrov

et al., 2020
[25] Sepliarskaia et al., 2018,
[26] Jiang and QiHe, 2014; Graus and Willemsen, 2015; Saavedra et al., 2016,
[27] LoeppTim Hussein and Ziegler, 2014.
[28] Chajewska et al., 1998; Boutilier, 2002; Vendrov et al., 2020
[29] Boutilier (2002)
[30] EtVendrov et al. (2020) et Guo et Sanner (Guo Scott, 2010
[31] Christakopoulou et al., 2016; GuoScott, 2010; Sepliarskaia et al., 2018;
[32] Vendrov et al., 2020.
[33] Zhao et al., 2013; Christakopoulou et al., 2016; Wang et al., 2018
[34] Zou et al., 2020b; LeeJinbae et al., 2019.
[35] Zhao et al., 2018; Chen et al., 2019ba; Xian et al., 2019; Zheng et al., 2018; HuQing et al., 2018; Zou et al., 2019; Chen et al., 2019aa; Ie et al., 2019; Liao et al., 2018; Pecune et al., 2019; Zhou et al., 2020b; Zou et al., 2020a; Wang et al., 2020c.
[36] Mahmood and Ricci (2007)
[37] Zhao et al., 2018; Zheng et al., 2018; Zou et al., 2019; Zhou et al., 2020b
[38] HuQing et al., 2018
[39] Zhou et al. (2020b
[40] Afsar et al. (2021).
[41] Mirzadeh et al., 2005; Thompson et al., 2004.
[42] ChoBart van Merri¨enboer et al., 2014
[43] HochreiterJürgen Schmidhuber, 1997
[44] Christakopoulou et al. (2018),
[45] ChoBart van Merriënboer et al., 2014.
[46] Zhang et al. (2018)
[47] SukhbaatarArthur Szlam et al., 2015; Kumar et al., 2016; Miller et al., 2016
[48] Christakopoulou et al., 2018; Lei et al., 2020a, 2020b.
[49] Chen and Pu, 2012; Wu et al., 2019a; Luo et al., 2020b; LuoScott et al., 2020.
[50] Chen and Pu, 2012; McCarthy et al., 2004; Smyth et al., 2004; Viappiani et al., 2007; Burke et al., 1997; Smyth and McGinty, 2003.
[51] Wu et al. (2019a)
[52] He et al., 2017
[53] Luo et al. (2020b)
[54] Lei et al. (2020b)
[55] Kipf and Welling, 2017; Schlichtkrull et al., 2018
[56] Chen et al., 2019bb; Zhou et al., 2020a; Xu et al., 2020; Liao et al., 2020.
[57] Zou et al., 2020
[58] Zhang et al., 2020c; Li et al., 2021b
[59] Mangili et al., 2020
[60] Penha et Hauff (2020
[61] Devlin et al., 2019
[62] Vaswani et al.2017
[63] Joel Grus,Data science par la pratique, 2017
[64]Vachon, N 2020.Écoutez avant de parler Intelligence Artificielle 5 mai 2021©
[65] Python 3 par la pratique pierre puiseux 2012

B. Références Web (Techniques)

[W1] https://redialdata.github.io/website

[W2] https://www.mturk.com/.

[W3] https://datasets.imdbws.com/

[W4] 100 https://nlp.stanford.edu/projects/glove/

[W5] 200 https://www.codeur.com/blog/google-bert/]

[W6] https://paperswithcode.com/

[W7] https://www.jedha.co/formation-ia

[W8] *https://en.wikipedia.org/wiki/Python_(programming_language)*

[W9] *https://www.w3schools.com/python/python_intro.asp*
Www.fr.mathworks.com/discovery/deep-learning.html)

[W10] https://lesdieuxducode.com/

[W11] http ://www.jalammar.github.io