République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université Larbi Tébessi - Tébessa

Faculté des Sciences Exactes et des Sciences de la Nature et de la Vie

Département : Mathématiques et Informatique

Mémoire de fin d'étude

Pour l'obtention du diplôme de MASTER

Domaine : Mathématiques et Informatique

Filière : Informatique

Option : Système information

Thème

Système de recommandation d'emploi intelligent basé sur le cadre BERT

Présenté Par : ghrieb ismail et Yahya issa diakhité

Devant le jury

Dr. BOUAKEZ Fatima	Université Larbi Tébessi	Président
Dr. ELAIFA Hassiba	Université Larbi Tébessi	Examinateur
DR. BREK Assia	Université Larbi Tébessi	Encadreur

Date de soutenance : 09/ 06/ 2024

Année : 2023/2024


بسم الله الرحمن الرحيم

Table des matières :

Liste des figures :

Liste des tableaux :

Liste des Abréviations :

NLP : Natural Laguage Processing

BERT : Bidirectional Encoder Representation from Transformers

MLM : Masked Language Model

NSP: Next Sentence Prediction

AI : Artificiel Intelligence

NDCG : Normalized Discounted Cumulative Gain

DCG : Discounted Cumulative Gain

CF : Collaborative Filtering

KBR: Knowledge-Based Recommendation)

CBR: Content-Based Recommendation)

IDCG: Ideal Discounted Cumulative Gain)

RNN: Recurrent Neural Network

ML: Machine Learning

Remerciement

*Au terme de ce travail, nous aimerions exprimer notre gratitude au **Dieu**.*

Nous sommes profondément reconnaissants de nous avoir accordé le courage, la

volonté et la patience nécessaires pour mener à bien ce travail.

Nous tenons également à exprimer notre immense reconnaissance envers

notre encadrant ***BREK***
Assia*pour son soutien inestimable, ses conseils avisés et sa présence constante.*

Votre expertise et votre dévouement ont été des sources d'inspiration pour nous.

Nous également remercier chaleureusement ***les membres du jury*** *, pour leur temps, leur exper tise et leurévaluation minutieuse de notre travail.*

Enfin, nous exprimons notre gratitude envers toutes les personnes qui ontcontribué de près o u de loin à l'élaboration de ce travail. Votre soutien, vosconnaissances partagées, votre as sistance technique et vos encouragements ont
été essentiels à notre réussite.

Dédicace

À ma chère maman,

Je te suis extrêmement reconnaissante pour ton amour inconditionnel, ta bienveillance et tes sacrifices. Tu as été mon premier amour et ma première inspiration. Merci d'avoir toujours été là pour moi, de m'avoir soutenue et encouragée dans tous mes projets. Ce travail est dédié à toi, en témoignage de l'amour infini que j'ai pour toi. Ta lumière éclaire ma vie et ton soutien constant me guide. Je te suis profondément reconnaissante pour tout ce que tu fais. Je t'aime infiniment.

À mon cher père,

Ton soutien inébranlable et tes conseils avisés ont été une inspiration pour moi. Tu m'as montré la voie de l'intégrité, de la persévérance et de la détermination. Cette réalisation est dédiée à toi, en reconnaissance de tout ce que tu as fait pour moi.

À mes amis précieux,

Que cette dédicace témoigne de ma reconnaissance éternelle envers ma famille et mes amis qui ont été présents à chaque étape de ma vie, m'encourageant à atteindre de nouveaux sommets.

Avec amour et gratitude,

Résumé :

La recherche d'emploi peut être un processus complexe et fastidieux, nécessitant de trier une grande quantité d'offres d'emploi pour trouver celles qui correspondent le mieux aux compétences et aux aspirations des candidats. Les systèmes de recommandation sont devenus des outils essentiels pour faciliter ce processus, en proposant des offres d'emploi pertinentes basées sur le profil des utilisateurs.

Le problème principal abordé de cette recherche est de trouver la meilleure recommandation d'emploi en exploitant la sémantique des documents d'emploi. tels que les descriptions de poste et les CV, qui contiennent une richesse d'informations sémantiques qui peuvent être utilisées pour améliorer la pertinence des recommandations. Cependant, extraire et utiliser efficacement cette sémantique représente un défi significatif.

La solution proposée dans ce travail repose sur l'utilisation de BERT, un modèle de langage puissant qui excelle dans la compréhension du contexte et des nuances des textes. Pour enrichir davantage les représentations sémantiques, nous avons intégré WordNet, une base de données lexicale riche en relations sémantiques entre les mots. En combinant BERT et WordNet, notre système de recommandation d'emploi vise à fournir des recommandations plus précises et pertinentes en capturant les subtilités et les relations sémantiques des documents d'emploi.

Les résultats de notre évaluation montrent que l'intégration de WordNet avec BERT améliore globalement les performances du système de recommandation, bien que cette amélioration ne soit pas universelle pour tous les profils. Les analyses des résultats révèlent que pour certains profils, l'ajout de WordNet peut introduire du bruit sémantique, diminuant ainsi la pertinence des recommandations. Malgré cela, notre approche démontre le potentiel d'amélioration significative dans la recommandation d'emploi en utilisant des techniques avancées de traitement du langage naturel.

Mots clés :

Recherche d'emploi, Systèmes de recommandation, BERT, WordNet, Sémantique des documents, Traitement du langage naturel, Recommandation d'emploi, Analyse sémantique

Abstract

Job searching can be a complex and tedious process, requiring one to sift through a large number of job offers to find those that best match the candidates' skills and aspirations. Recommendation systems have become essential tools to facilitate this process by suggesting relevant job offers based on user profiles.

The main problem addressed in this research is to find the best job recommendation by leveraging the semantics of job documents, such as job descriptions and resumes, which contain a wealth of semantic information that can be used to enhance the relevance of recommendations. However, effectively extracting and utilizing this semantics represents a significant challenge.

The solution proposed in this work relies on the use of BERT, a powerful language model that excels in understanding the context and nuances of texts. To further enrich the semantic representations, we integrated WordNet, a lexical database rich in semantic relationships between words. By combining BERT and WordNet, our job recommendation system aims to provide more precise and relevant recommendations by capturing the subtleties and semantic relationships of job documents.

The results of our evaluation show that integrating WordNet with BERT generally improves the performance of the recommendation system, although this improvement is not universal for all profiles. Analysis of the results reveals that for some profiles, the addition of WordNet can introduce semantic noise, thereby reducing the relevance of recommendations. Despite this, our approach demonstrates the potential for significant improvement in job recommendations using advanced natural language processing techniques.

Keywords: Job search, Recommendation systems, BERT, WordNet, Document semantics, Natural language processing, Job recommendation, Semantic analysis

I. Introduction Générale

Dans le contexte actuel du marché du travail, caractérisé par sa rapidité et sa compétitivité, l'efficacité de la correspondance entre les demandeurs d'emploi et les opportunités d'emploi appropriées revêt une importance primordiale. Les méthodes traditionnelles de recherche d'emploi et de recrutement reposent souvent sur des processus manuels, qui peuvent être longs, inefficaces et sujets à des biais. Pour relever ces défis, les chercheurs et les praticiens se sont tournés vers des technologies avancées telles que l'intelligence artificielle (IA) et l'apprentissage automatique (ML) pour développer des systèmes de recommandation d'emploi intelligents qui rationalisent le processus de recrutement et améliorent l'expérience de recherche d'emploi tant pour les candidats que pour les employeurs.

L'un de ces solutions innovantes est le système de recommandation d'emploi intelligent basé sur le cadre BERT. Ce système exploite des techniques de traitement du langage naturel (NLP) de pointe pour analyser et comprendre la sémantique des descriptions d'emploi et des profils de candidats, permettant des recommandations d'emploi plus précises et personnalisées. Au cœur de ce système se trouve BERT (Bidirectional Encoder Representations from Transformers), un puissant modèle d'apprentissage profond qui excelle dans la capture des relations contextuelles au sein des données textuelles.

II. Objectif de l'étude

L'objectif principal du système de recommandation d'emploi intelligent est de combler le fossé entre les demandeurs d'emploi et les employeurs en fournissant des recommandations d'emploi sur mesure qui correspondent aux compétences, aux qualifications et aux préférences de chaque candidat avec les exigences des postes disponibles. En exploitant les capacités de BERT (Bidirectionnel Encoder Representation from Transformers) et d'autres algorithmes de Machine learning, le système peut identifier efficacement les opportunités d'emploi pertinentes à partir de vastes ensembles de données, réduisant ainsi considérablement le temps et les efforts nécessaires aux demandeurs d'emploi pour trouver un emploi adapté.

Le processus de recommandation commence par la collecte de données à partir de différentes sources, notamment les offres d'emploi, les CV et les interactions des utilisateurs. Ces données sont ensuite prétraitées pour éliminer le bruit, normaliser les formats et extraire les caractéristiques pertinentes. Des modèles basés sur BERT (Bidirectionnel Encoder Representation from Transformers) sont utilisés pour encoder les descriptions d'emploi et les profils de candidats en vecteurs de haute dimension, capturant leur signification sémantique et leur contexte.

À l'aide de ces représentations encodées, , le système calcule des scores de similarité entre les offres d'emploi et les profils de candidats, classant les opportunités d'emploi en fonction de leur pertinence pour chaque candidat. De plus, le système prend en compte des facteurs

supplémentaires tels que les préférences de localisation, les attentes salariales et les objectifs de carrière pour affiner davantage les recommandations et améliorer leur précision.

L'un des principaux avantages du système de recommandation d'emploi intelligent est sa capacité à s'adapter et à apprendre des retours d'expérience des utilisateurs au fil du temps. En sollicitant les commentaires des demandeurs d'emploi sur la pertinence et la qualité des offres d'emploi recommandées, le système peut continuellement affiner ses algorithmes de recommandation et améliorer l'expérience utilisateur globale. Ce processus itératif de rétroaction et d'amélioration garantit que le système reste réactif aux besoins et aux préférences évolutifs de ses utilisateurs.

En plus de ses avantages pour les demandeurs d'emploi, le système de recommandation d'emploi intelligent offre également des avantages significatifs pour les employeurs et les recruteurs. En automatisant le processus de correspondance des candidats et en fournissant des recommandations d'emploi plus ciblées, le système permet aux recruteurs d'identifier plus efficacement les meilleurs talents et d'accélérer le processus de recrutement.

Cela permet non seulement de gagner du temps et des ressources pour les employeurs, mais aussi d'améliorer la qualité globale des recrutements.

Cependant, le développement et la mise en œuvre d'un tel système de recommandation avancé posent plusieurs défis et considérations. Ceux-ci comprennent les préoccupations concernant la confidentialité et la sécurité des données, la nécessité d'une infrastructure robuste pour gérer le traitement de données à grande échelle, et l'optimisation continue des algorithmes de recommandation pour garantir la pertinence et la précision.

En conclusion, le système de recommandation d'emploi intelligent basé sur le cadre BERT (Bidirectionnel Encoder Representation from Transformers)représente une solution de pointe aux défis de la recherche d'emploi et du recrutement à l'ère numérique. En exploitant la puissance de l'IA et de l'apprentissage automatique, ce système permet aux demandeurs d'emploi de trouver leurs opportunités d'emploi idéales tout en aidant les employeurs à se connecter plus efficacement avec les candidats les mieux adaptés. Alors que la technologie continue d'évoluer, des systèmes de recommandation d'emploi intelligents comme celui-ci joueront un rôle de plus en plus vital dans l'avenir du travail et de l'emploi.

III. Problématique

La problématique qui sous-tend le développement et l'implémentation du système de recommandation d'emploi intelligent basé sur le cadre BERT (Bidirectionnel Encoder Representation from Transformers)est multidimensionnelle et soulève plusieurs questions essentielles concernant l'efficacité, la pertinence et l'impact de cette technologie novatrice dans le domaine du recrutement et de la recherche d'emploi.

Tout d'abord, une problématique majeure réside dans la complexité du processus de correspondance entre les demandeurs d'emploi et les offres d'emploi. Les systèmes traditionnels de recrutement reposent souvent sur des critères de correspondance simplistes, tels que des mots-clés ou des titres de poste, qui peuvent ne pas capturer la richesse sémantique des descriptions d'emploi et des profils de candidats. Cette approche conduit souvent à des recommandations inappropriées et à des correspondances inexactes, ce qui nuit à la satisfaction des utilisateurs et compromet l'efficacité globale du processus de recrutement.

De plus, la variabilité et la dynamique des préférences des utilisateurs et des exigences des postes vacants ajoutent une dimension supplémentaire de complexité à la problématique. Les demandeurs d'emploi ont des besoins et des aspirations différents, et leurs préférences en matière de lieu de travail, de rémunération, de culture d'entreprise, etc., peuvent varier considérablement. De même, les employeurs recherchent des candidats dotés de compétences spécifiques et adaptés à leur culture organisationnelle, ce qui rend difficile la mise en correspondance des deux parties de manière efficace et précise.

Une autre problématique importante concerne la qualité et la quantité des données disponibles pour l'entraînement et la validation du système de recommandation. Les données sur les offres d'emploi et les profils de candidats peuvent être hétérogènes, incomplètes ou bruitées, ce qui peut compromettre la performance et la fiabilité des algorithmes de recommandation. De plus, la confidentialité et la sécurité des données des utilisateurs doivent être garanties à tout moment, ce qui soulève des défis supplémentaires en matière de gestion et de protection des informations sensibles.

En outre, la mise en œuvre d'un système de recommandation d'emploi intelligent nécessite une infrastructure informatique robuste et évolutive pour gérer le traitement des données massives et le déploiement des modèles d'apprentissage automatique. Ces exigences en matière d'infrastructure peuvent représenter un obstacle financier et technique pour de nombreuses organisations, en particulier les petites et moyennes entreprises qui disposent de ressources limitées.

Enfin, une problématique éthique et sociétale importante concerne l'impact potentiel du système de recommandation d'emploi sur l'équité et la diversité dans le processus de recrutement. Les algorithmes de recommandation peuvent involontairement reproduire et amplifier les biais existants dans les données d'entraînement, ce qui peut entraîner des discriminations injustes à l'égard de certains groupes de candidats.

Il est donc crucial de concevoir et de mettre en œuvre des mécanismes de correction des biais et de surveillance de la performance pour garantir l'équité et l'inclusivité du système de recommandation.

En résumé, la problématique du système de recommandation d'emploi intelligent basé sur le cadre BERT(Bidirectionnel Encoder Representation from Transformers) soulève des questions complexes et interdépendantes liées à la précision, à la pertinence, à la qualité des données, à l'infrastructure informatique, à l'éthique et à l'équité. La résolution de ces défis nécessitera une approche multidisciplinaire et collaborative, impliquant des experts en informatique, en sciences comportementales, en éthique des données et en droit du travail, pour garantir le développement et le déploiement responsables de cette technologie transformative dans le domaine du recrutement et de l'emploi.

III. Choix & Intérêt du sujet

- **Intérêt de travail**

L'intérêt de ce travail réside dans sa capacité à répondre à un besoin croissant sur le marché de l'emploi en proposant une solution innovante et efficace pour mettre en relation entre les demandeurs d'emploi et les employeurs. En exploitant les avancées en intelligence artificielle et en traitement automatique du langage naturel, ce système offre la possibilité d'améliorer considérablement l'efficacité du processus de recrutement en fournissant des recommandations d'emploi plus précises et personnalisées. De plus, ce travail présente un intérêt pratique pour les professionnels des ressources humaines, les chercheurs en informatique et les spécialistes en intelligence artificielle en leur offrant un cadre méthodologique pour concevoir, développer et évaluer des systèmes de recommandation d'emploi intelligents. Enfin, ce travail contribue à la littérature académique en explorant les défis techniques, éthiques et sociaux associés à la conception et à l'implémentation de tels systèmes, tout en proposant des solutions et des recommandations pour garantir leur efficacité et leur équité

- **Intérêt proféssionnelle**

L'intérêt professionnel de ce travail réside dans sa capacité à fournir aux professionnels des ressources humaines et du recrutement une solution innovante pour optimiser le processus de recrutement. En utilisant un système de recommandation d'emploi intelligent basé sur le cadre BERT, ces professionnels peuvent identifier rapidement les candidats les plus pertinents pour les postes vacants, ce qui permet de gagner du temps et des ressources tout en améliorant la qualité des recrutements. Cette approche leur permet de rester compétitifs sur le marché du travail en adoptant des technologies de pointe et en améliorant leur efficacité opérationnelle.

- **Intérêt académique**

L'intérêt académique de ce travail réside dans son exploration des avancées en intelligence artificielle, en particulier dans le domaine du traitement automatique du langage naturel (NLP), et son application au domaine spécifique du recrutement et de la recherche d'emploi. En étudiant et en développant un système de recommandation d'emploi intelligent basé sur le cadre BERT(Bidirectionnel Encoder Representation from Transformers), ce travail contribue à la littérature académique en proposant une méthodologie novatrice pour résoudre des problèmes

complexes de correspondance entre les compétences des candidats et les exigences des postes vacants. De plus, ce travail offre aux chercheurs une opportunité d'explorer les défis techniques, éthiques et sociaux associés à la conception et à l'implémentation de tels systèmes, tout en proposant des solutions et des recommandations pour garantir leur efficacité et leur équité.

1 Chapitre 1:Contexte générale

1.1 Introduction:

Dans ce chapitre nous allons aborder les points importants qui sont les bases du système de recommandation en général. En commençant par la définition du système de recommandation et ses différentes caracteristiques puis celle de modèle BERT(Bidirectionnel Encoder Representation from Transformers)qui est le cœur de notre recherche et ainsi ses caractérisés et ses variations et on finira par le point système de recommandation basé sur BERT(Bidirectionnel Encoder Representation from Transformers)

1.2 Les systèmes de recommandation

1.2.1 Définition

Les systemes de recommendation sont des outils informatiques conçus pour proposer des éléments pertinents et personnalisés à un utilisateur en se basant sur ses préférences, son historique d'utilisation ou d'autres données contextuelles. Ces systèmes visent à faciliter la découverte de nouveaux contenus ou produits, ainsi qu'à améliorer l'expérience utilisateur en fournissant des suggestions adaptées.[*31*]

Les systèmes de recommandation basent sur des algorithmes informatiques qui analysent les données de comportement et les préférences des utilisateurs afin de leur proposer des recommandations pertinentes et personnalisées. Ces recommandations peuvent porter sur une variété de domaines tels que les produits, les contenus multimédias, les personnes à suivre sur les réseaux sociaux, etc. Les systèmes de recommandation sont largement utilisés dans le commerce électronique, les réseaux sociaux, les plateformes de streaming, et d'autres domaines, pour améliorer l'expérience utilisateur et stimuler l'engagement [34].

Sur les plateformes de streaming vidéo comme Netflix, les systèmes de recommandation utilisent l'historique de visionnage d'un utilisateur ainsi que les évaluations qu'il a données pour suggérer des films ou des séries similaires. Aussi, dans les boutiques en ligne telles qu'Amazon, les systèmes de recommandation analysent les achats précédents d'un utilisateur ainsi que les produits consultés pour lui suggérer des articles susceptibles de l'intéresser. De plus, sur les réseaux sociaux comme Facebook, les systèmes de recommandation recommandent des amis potentiels en se basant sur les amis actuels de l'utilisateur, ses centres d'intérêt et d'autres facteurs.

1.2.2 Les approches de recommandation

Les auteurs dans [*24*] ont classé les approches des systèmes de recommandation dans les quatre catégories principales suivantes : filtrage collaboratif (CF), recommandation basée sur le contenu (CBR), recommandation basée sur la connaissance (KBR) et approches hybrides.

- **Filtrage collaboratif (CF) :**qui analyse les comportements passés des utilisateurs ainsi que ceux d'autres utilisateurs similaires pour identifier des modèles de préférences et recommander des éléments qui sont susceptibles d'intéresser l'utilisateur cible. Cette

approche repose sur le principe selon lequel les utilisateurs qui ont des préférences similaires dans le passé auront également des préférences similaires à l'avenir [39].

- **Filtrage basé sur le contenu :** Le filtrage basé sur le contenu recommande des éléments en comparant le contenu des éléments avec les données du profil utilisateur. Chaque élément est représenté par un ensemble de descripteurs, tels que des genres de films ou des termes fréquents dans des articles textuels. Le système analyse ces descripteurs pour un élément précédemment évalué par l'utilisateur et construit un modèle des intérêts de l'utilisateur afin de générer des recommandations [33]. Par exemple, si un utilisateur aime une page web contenant les mots "Appareil photo", "Objectif" et "Trépied", le système recommandera des pages liées à l'électronique. La description des articles et le profil des préférences de l'utilisateur sont essentiels dans ce processus. Les filtres basés sur le contenu tentent de générer des recommandations en comparant la similarité entre plusieurs éléments candidats et ceux déjà évalués par l'utilisateur, recommandant ainsi les éléments les plus pertinents.
- **Filtrage basée sur la connaissance :**La recommandation basée sur la connaissance utilise des connaissances explicites, des règles ou des modèles spécifiques au domaine pour suggérer des éléments ou des choix aux utilisateurs. Contrairement au filtrage collaboratif ou au filtrage basé sur le contenu, qui dépendent des données ou des similarités, ces systèmes exploitent des règles prédéfinies, des ontologies ou des connaissances expertes. Ils sont souvent utilisés dans des domaines complexes avec des règles ou des contraintes strictes, comme le secteur médical, la finance ou l'industrie manufacturière [*21*]. Un exemple typique est le "recommandeur de restaurants Entrée", qui suggère des restaurants similaires à ceux que l'utilisateur connaît et aime, en permettant aux utilisateurs de préciser leurs préférences pour affiner leurs critères de recherche.
- **Filtrage hybrides :** Un système de recommandation hybride est une autre catégorie de systèmes de recommandation qui cherche à dépasser les limitations des autres approches*[1]*. En combinant deux ou plusieurs techniques de recommandation, il vise à obtenir une meilleure optimisation. L'une des approches hybrides les plus populaires est celle qui adapte la combinaison heuristique du filtrage basé sur le contenu et du filtrage collaboratif [*29*].
- **Autre classification :** Il y’a plusieurs autres classifications de sous-systèmes de recommandation comme les systèmes de recommandation basés sur la connaissance, les systèmes de recommandation basés sur les ontologies, systèmes de recommandation intelligents, Systèmes de recommandation basés sur la démographie, systèmes de recommandation sensibles au contexte, systèmes de recommandation sensibles au temps, et les systèmes de recommandation basés sur les avis.

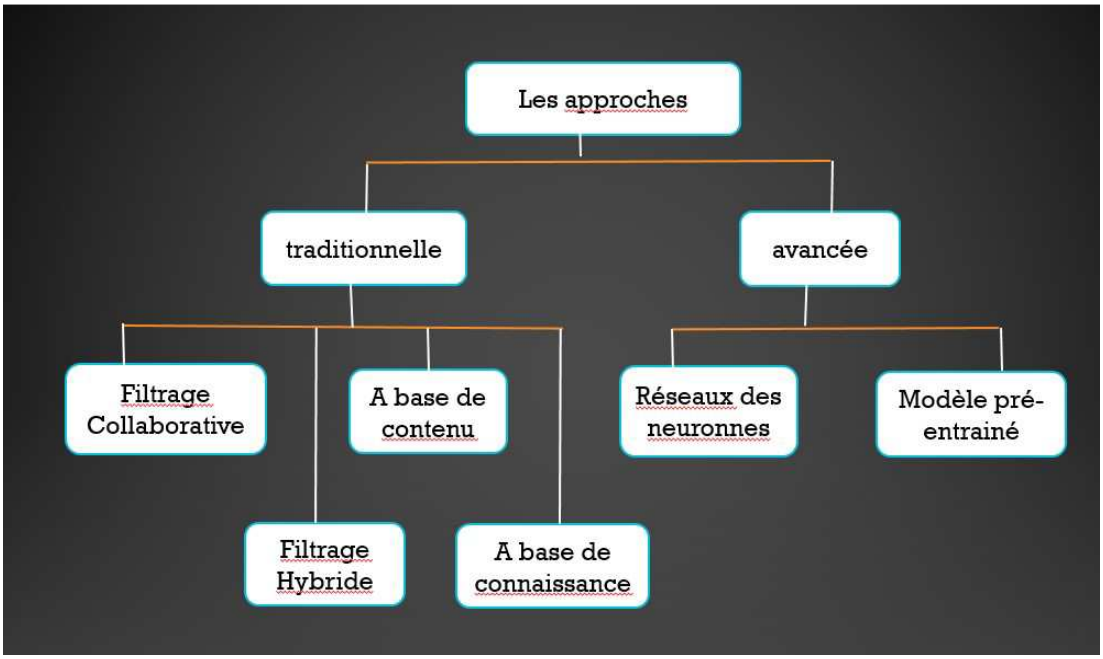


Figure 1 : Les approches de système de recommandation

1.3 Domaines d'application

Les systèmes de recommandation, capables d'analyser et de prédire les préférences des utilisateurs, ont des applications variées dans de nombreux domaines. Leur utilisation s'étend bien au-delà des recommandations de produits sur les sites de commerce électronique pour inclure des secteurs tels que la musique, le cinéma, les réseaux sociaux, le tourisme, la santé et l'éducation [*19*].

- **Commerce électronique** : Les systèmes de recommandation aident les entreprises comme Amazon et Alibaba à proposer des produits personnalisés en analysant les habitudes d'achat des utilisateurs, augmentant ainsi les ventes et améliorant l'expérience d'achat.
- **Divertissement** : Des plateformes comme Netflix, Spotify et YouTube utilisent ces systèmes pour suggérer des films, des séries, de la musique et des vidéos, aidant les utilisateurs à découvrir de nouveaux contenus adaptés à leurs goûts, ce qui stimule l'engagement.
- **Réseaux sociaux** : Ils recommandent des amis, des pages et des groupes, et personnalisent le contenu du flux d'actualités selon les intérêts et interactions des utilisateurs, les aidant à découvrir du contenu pertinent.

En dehors du domaine numérique, les systèmes de recommandation sont également utilisés dans [*30*] :

- **Tourisme** : Ils aident les voyageurs à planifier leurs voyages en suggérant des destinations et des attractions correspondant à leurs préférences.

- **Santé** : Ils peuvent proposer des traitements personnalisés basés sur le profil médical et les antécédents des patients.
- **Éducation** : Ils recommandent des cours en ligne, des ressources pédagogiques et des activités d'apprentissage adaptées aux intérêts et besoins des apprenants, améliorant leur engagement et optimisant leur parcours d'apprentissage.

1.4 Système de recommandation de l'emploie

Un système de recommandation d'emploi est un système informatique qui utilise des techniques d'analyse de données et d'intelligence artificielle pour recommander des offres d'emploi pertinentes aux utilisateurs en fonction de leurs compétences, de leur expérience professionnelle, de leurs préférences et d'autres critères pertinents [13]. Ces systèmes sont conçus pour faciliter le processus de recherche d'emploi en aidant les candidats à trouver des opportunités correspondant à leur profil, et en permettant aux employeurs de cibler efficacement les candidats les plus qualifiés pour leurs postes vacants. Les systèmes de recommandation d'emploi peuvent être utilisés sur des plateformes de recrutement en ligne dédiées, des réseaux sociaux professionnels, des applications mobiles, etc.

Les systèmes de recommandation d'emploi visent à améliorer l'efficacité du processus de recrutement et à faciliter les connexions entre les employeurs et les candidats. Sur la base de plusieurs études comparatives dans le domaine de la e-recrutement [11 ,12], l'approche basée sur le contenu est plus adaptée à ce contexte, selon la nécessité d'examiner le contenu des documents de domaine pour fournir la meilleure recommandation. Plusieurs approches ont été utilisées pour système de recommandation d'emploi les principales sont :

1.4.1 Les systèmes de recommandation basés sur le contenu (CBR) utilisent essentiellement les informations recueillies auprès des candidats et calculent leur similarité avec les informations provenant des offres d'emploi. Une source courante d'informations sur les candidats est leur CV. Ainsi, ces méthodes reposent généralement sur la capacité à croiser des informations entre différentes sources de manière sémantiquement efficace. Plusieurs techniques et algorithmes peuvent être employés pour calculer les similarités.Dans [36], une mise en œuvre mobile est proposée pour obtenir la valeur de similarité croisée, catégorisant les caractéristiques des emplois et des candidats en : (a) auto-description ; et (b) préférences, avec une correspondance croisée un à un entre elles.Dans [*14*], la similarité cosinus est calculée dans un modèle utilisant les transitions de carrière des candidats. Comme proposé dans [*25*], le corpus d'une offre d'emploi peut être enrichi par des termes communs afin d'optimiser les capacités de correspondance. D'autres recherches, comme [*26*], contribuent en aidant à sélectionner les champs appropriés à utiliser pour la correspondance, en ignorant les autres. Dans [*3*], une approche est proposée pour apparier les attributs de manière plus flexible, allant de la correspondance exacte à une plage, voire avec des limites supérieures et inférieures.

1.4.2 Lefiltrage collaboratif (CF),reposent sur les données de comportement des utilisateurs, généralement stockées dans une matrice représentant les évaluations utilisateur-élément. Dans le recrutement en ligne, cette matrice capture souvent le comportement de clic, où une note de 1 indique qu'un chercheur d'emploi a cliqué sur une offre pour voir les détails, et 0 sinon. En l'absence de telles données d'interaction, la séquence des emplois précédents peut être utilisée pour peupler la matrice de notation, nécessitant des catégories d'emplois pour représenter les éléments discrets dans la matrice [*9*]. L'article [*23*] aborde le problème de la recommandation d'emplois aux étudiants en calculant les similarités entre eux. Lorsqu'un étudiant reçoit une offre et évalue une entreprise, le système peut prédire la note et inférer la possibilité de correspondance entre cette entreprise et des étudiants similaires.

1.4.3 Les systèmes de recommandation basés sur la connaissance (KBR) améliorent les capacités de correspondance en utilisant une expertise spécifique au domaine pour identifier des similarités contextuelles plutôt que des termes [*27*]. L'étude [*12*] propose une ontologie pour les emplois dans le secteur informatique. Dans [*11*], les auteurs exploitent les données des profils publics des utilisateurs de LinkedIn pour découvrir les relations entre les emplois et les compétences des personnes en utilisant l'analyse sémantique latente (LSA) pour générer des associations sémantiques. AnnoJob[*8*], est un système de recommandation qui se base sur le calcul de similarité sémantique entre les offres d'emploi et les CV, en exploitant des graphes de connaissances tels que WikiData pour trouver les relations sémantiques

1.4.4 Les approches hybrides combinent essentiellement plusieurs méthodes pour obtenir de meilleurs résultats. La combinaison des approches CBR et CF est abordée dans [*16*], où les auteurs proposent un processus en deux étapes pour calculer un score correspondant simultanément aux informations extraites du CV et à d'autres candidatures. Dans [*15*], un moteur de filtrage d'informations hybride intègre plusieurs moteurs de recommandation : basés sur la démographie, les connaissances, le contenu, le concept et l'ontologie, intégrés avec des recommandataires comportementaux et collaboratifs répartis sur différents modules intégrés. Dans [38], une méthode d'ensemble pour la recommandation d'emplois à la ACM RecSys Challenge 2016 est présentée, décrite comme une solution combinant les mérites du filtrage collaboratif traditionnel et du filtrage basé sur le contenu. De plus, dans [*17*], les auteurs font des recommandations basées sur les compétences approuvées socialement.

1.5 **BERT**(Bidirectionnel Encoder Representation from Transformers)

1.5.1 Définition

BERT est l'acronyme de Bidirectional Encoder Representation from Transformers, crée par Google AI en fin 2018, C'est un modèle de traitement du langage naturel (NLP) pré-entraîné qui a révolutionné de nombreux domaines de l'apprentissage automatique appliqué au langage naturel. Le modèle de langage BERT se compose de plusieurs encodeurs de transformateurs empilés les uns sur les autres et est conçu à partir de texte non étiqueté pour pré-entraîner des

représentations bidirectionnelles profondes en conditionnant conjointement sur le contexte gauche et droit dans toutes les couches [*4*].

Lors de l'utilisation du modèle BERT pré-entraîné, les étapes sont les suivantes : (1) insérer les entrées et les sorties spécifiques à la tâche dans BERT et (2) affiner tous les paramètres de bout en-bout.
Ils ont présenté deux types généraux, nommés le BERTbase et le BERT-large, qui ont utilisé le BooksCorpus avec 800 millions de mots [*5*] et l'anglais Wikipédia avec 2 500 millions de mots [*14*],respectivement.Les caractéristiques de ces modèles sont présentées ci-dessous :

BERT-base : L : 12, H : 768, A : 12, Paramètres : 110M

BERT-large : L : 24, H : 1024, A : 16, Paramètres : 340M

L représente le nombre d'encodeurs empilés, **H** la taille de la couche cachée, et **A** le nombre de têtes d'auto-attention [*2*]. Les performances de BERT dépendent du type de modèle, c'est-à-dire que BERT-large peut atteindre des précisions plus élevées que BERT-base. Cependant, cette amélioration de la précision en utilisant BERT-large s'accompagne du coût d'exiger des ressources plus étendues pour être complété [*2*].

Les idées principales derrière BERT sont : **Masked Language Model (MLM)** et **Next sentence prediction (NSP)**

1.5.2 Masked LM (MLM)

Avant de fournir des séquences de mots à BERT(Bidirectionnel Encoder Representation from Transformers), 15% des mots de chaque séquence sont remplacés par un jeton [MASK]. Le modèle tente ensuite de prédire la valeur originale des mots masqués, en se basant sur le contexte fourni par les autres mots non masqués de la séquence. En termes techniques, la prédiction des mots de sortie nécessite :

a. Ajout d'une couche de classification au-dessus de la sortie de l'encodeur.
b. Multiplication des vecteurs de sortie par la matrice d'incorporation, les transformant en dimension de vocabulaire.
c. Calcul de la probabilité de chaque mot dans le vocabulaire avec softmax.

La fonction de perte de BERT prend en considération uniquement la prédiction des valeurs masquées et ignore la prédiction des mots non masques. Par conséquent, le modèle converge plus lentement que les modelés directionnels, une caractéristique compensée par sa sensibilisation accrue au contexte

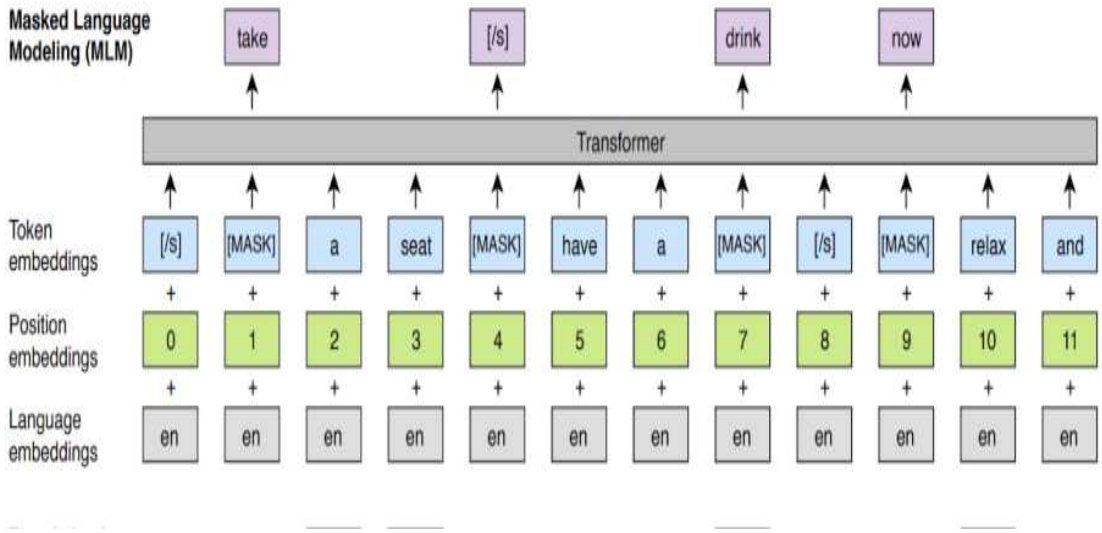


Figure 2 : Masked language Model (MLM)[20]

1.5.3 Next Sentence Prediction (NSP)

Dans le processus d'entraînement de BERT, le modèle reçoit des paires de phrases en entrée et apprend à prédire si la deuxième phrase de la paire est la phrase suivante dans le document original. Pendant l'entraînement, 50% des entrées sont une paire dans laquelle la deuxième phrase est la phrase suivante dans le document original, tandis que dans les autres 50%, une phrase aléatoire du corpus est choisie comme deuxième phrase. L'hypothèse est que la phrase aléatoire sera déconnectée de la première phrase.

Pour aider le modèle à distinguer les deux phrases lors de l'entraînement, l'entrée est traitée de la manière suivante avant d'entrer dans le modèle :

1. Un jeton [CLS] est inséré au début de la première phrase et un jeton [SEP] est inséré à la fin de chaque phrase.
2. Un encodage de phrase indiquant la phrase A ou la phrase B est ajouté à chaque jeton. Les encodages de phrase sont similaires en concept aux encodages de jeton avec un vocabulaire de 2.
3. Un encodage de position est ajouté à chaque jeton pour indiquer sa position dans la séquence. Le concept et la mise en œuvre de l'encodage de position sont présentés dans l'article sur le Transformer.

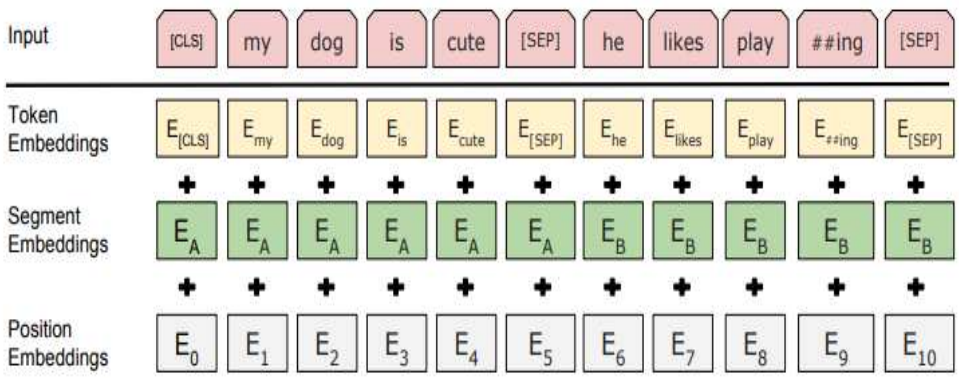


Figure 3 : Prédiction des phrases suivante (NSP)[10].

Pour prédire si la deuxième phrase est effectivement connectée à la première, les étapes suivantes sont effectuées :

1. L'ensemble de la séquence d'entrée passe à travers le modèle Transformer.
2. La sortie du jeton [CLS] est transformée en un vecteur de forme 2×1, en utilisant une simple couche de classification (matrices de poids et de biais apprises).
3. Calcul de la probabilité de IsNextSequence avec softmax.

Lors de l'entraînement du modèle BERT, Masked LM et Next Sentence Prediction sont entraînés ensemble, dans le but de minimiser la fonction de perte combinée des deux stratégies.

1.5.4 Transformer

Le transformer est devenu l'architecture prédominante dans les tâches de traitement du langage naturel (NLP) en remplaçant les architectures traditionnelles basées sur les réseaux de neurones récurrents (RNN). Surtout, le mécanisme d'auto-attention permet la parallélisation, de sorte que les relations entre toutes les paires de jetons dans une séquence d'entrée peuvent être calculées en une seule fois. [*18*].

Le modèle se compose des blocs encodeur et décodeur avec une fonction d'activation softmax pour normaliser les probabilités de sortie. L'entrée du modèle est une séquence de données. Les mots d'entrée sont intégrés et passés à travers les encodeurs de position, qui attribuent des vecteurs aux mots en fonction de leur position dans une phrase ; ainsi, extrayant le sens contextuel des mots d'entrée. Les blocs encodeurs, composés de l'attention multi-têtes et d'un réseau de neurones à propagation avant, reçoivent les intégrations [*2*].

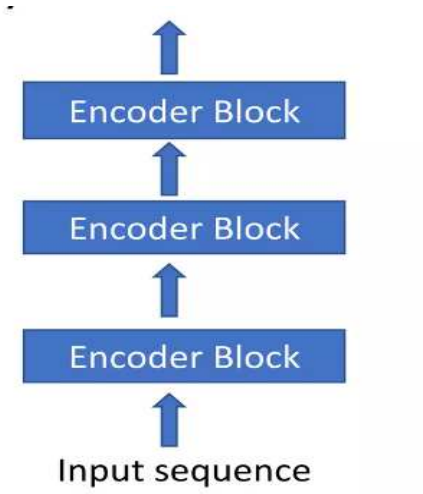


Figure 4 : couche encoder

Les auteurs de [35] ont expliqués que l'encodeur est composé d'une série de N = 6 couches identiques, chacune contenant deux sous-couches. La première sous-couche est un mécanisme d'auto-attention multi-têtes, tandis que la deuxième sous-couche se compose d'un réseau de neurones à propagation avant simple et entièrement connecté fonctionnant sur chaque position individuellement. Des connexions résiduelles sont appliquées autour des deux sous-couches, suivies d'une normalisation de couche. Plus précisément, la sortie de chaque sous-couche est normalisée en utilisant LayerNorm(x + Sublayer(x)), où Sublayer(x) représente la fonction effectuée par la sous-couche elle-même. Pour faciliter ces connexions résiduelles, toutes les sous-couches du modèle, y compris les couches d'incorporation, produisent des sorties de dimension dmodel = 512.

Le décodeur dispose des encodeurs de position et des couches d'attention multi-têtes masquées qui fonctionnent de manière similaire au bloc d'encodeur. Les vecteurs d'attention de ses couches d'attention multi-têtes masquées et ceux du bloc d'encodeur sont ensuite transmis à un autre bloc d'attention multi-têtes. Chaque vecteur représente la relation avec les autres mots dans l'ensemble du document. Les vecteurs sont ensuite transmis à un réseau de neurones à propagation avant, puis à la couche linéaire et enfin à une fonction d'activation softmax qui les convertit en une distribution de probabilité pour la sortie [*2*].

1.5.5 Variation de BERT

De nombreux nouveaux modèles qui sont entraînés dans différentes langues ou optimisés sur des ensembles de données spécifiques au domaine sont inspirés par l'architecture de BERT au fil du temps. Des améliorations continues ont lieu, et plusieurs versions optimisées sont régulièrement publiées. Caractéristiques de quelques-uns des modèles BERT pré-entraînés disponibles commercialement voir ci-dessous :

Par exemple pour DistilBERT une méthode pour pré-entraîner un modèle de représentation linguistique générale plus petit, qui peut ensuite être affiné avec de bonnes performances sur

une large gamme de tâches comme ses homologues plus grands. Alors que la plupart des travaux antérieurs ont examiné l'utilisation de la distillation pour la construction de modèles spécifiques à une tâche, nous tirons parti de la distillation des connaissances pendant la phase de pré-entraînement et montrons qu'il est possible de réduire la taille d'un modèle BERT de 40 %, tout en conservant 97 % de ses capacités de compréhension du langage et en étant 60 % plus rapide [32].

1.6 Système de recommandation de l'emploie à base de BERT

Dans le contexte des systèmes de recommandation d'emploi (JRS), il offre plusieurs avantages significatifs. Tout d'abord, il peut être utilisé pour extraire des caractéristiques pertinentes des descriptions d'emplois et des profils d'utilisateurs, permettant ainsi une compréhension plus profonde du contenu textuel et des préférences des utilisateurs. Ensuite, BERT peut être utilisé pour classer les séquences de texte, ce qui permet de prédire la pertinence des offres d'emploi pour les utilisateurs en se basant sur des informations contextuelles et sémantiques [7].

De plus, BERT peut être fine-tuné pour des tâches spécifiques liées aux systèmes de recommandation d'emploi, telles que la classification de séquences, la génération de texte et l'analyse de sentiment. Cette flexibilité permet aux chercheurs et aux praticiens d'adapter BERT à leurs besoins spécifiques et de développer des systèmes de recommandation d'emploi plus sophistiqués et plus précis.

Les systèmes de recommandation d'emploi basés sur BERT exploitent les capacités de traitement du langage naturel de BERT pour fournir des recommandations d'emploi personnalisées aux utilisateurs.

Plusieurs approches ont été explorées dans la littérature. Certains chercheurs ont utilisé BERT pour représenter les mots-clés et les descriptions d'emploi, en extrayant des informations pertinentes à partir de ces textes pour mieux comprendre les exigences des postes et les préférences des utilisateurs. D'autres ont intégré BERT dans des architectures de systèmes de recommandation plus complexes, combinant des informations sur les utilisateurs, les emplois et d'autres données contextuelles pour générer des recommandations plus précises.

Des évaluations comparatives ont été menées pour évaluer les performances des systèmes de recommandation d'emploi basés sur BERT par rapport à d'autres approches. Ces évaluations comprennent souvent des mesures de précision, de rappel, de F-score et d'autres métriques d'évaluation de la performance sur des ensembles de données réels ou simulés [*28*].

Cependant, malgré les avantages potentiels de l'utilisation de BERT dans les systèmes de recommandation d'emploi, plusieurs défis persistent. Ces défis comprennent la nécessité de grandes quantités de données d'entraînement pour pré-entraîner efficacement les modèles BERT, les problèmes liés à la qualité des données, tels que le bruit et les biais, et les

considérations éthiques liées à la confidentialité des utilisateurs et à la transparence des recommandations.

[*22*] ils ont fait une application affinée sur l'un de variant de BERT Siamese sentence-BERT (SBERT) model pour recommander des postes vacants a des chercheurs d'emploi. Ils ont expliqué leur méthodologie en décrivant SBERT multilingue avec bi-encodeur et score de similarité cosinus comme couche de sortie. Utilisation de SBERT vise à calculer la similarité sémantique par pairs. Dans leur article s'articule autour de la construction de représentation des caractéristique utile d'information textuelle dans les offres d'emploi et des CVLes Baseline utilisée pour générer les plongements : le non-supervisée en présentant à la fois les postes vacants et le CV sous-forme de vecteurs pondérées TF-IDF ou l'intégration de BERT pré-entrainées, aussi pour le supervisée et un classificateur de foret aléatoire qui entrainée au-dessus de la représentation de caractéristique (TFIDF + RF, BERT + RF). L'application de méthodes supervisée est de comparer aux Baseline non-supervisée pour établir plus en détail l'étendu de la lacune de vocabulaire susmentionnées et la mesure dans laquelle la nature hétérogène des types de documents.

1.7 Discussion

Le tableau ci-dessous montre la comparaison des différentes approches utilisées pour le recommandation d'emploi, le type de recommandation et leur limite.

Reference de travail	Approche utilisee	Type de recommandation	Limite de l'approche
[41]	Filtarge Collaboratif	Offre d'emploi	Problème de démarrageà froid
[42]	Fitrage base sur le contenu	CV	Manque de diversité
[43]	Hybrides	Offre d'emploi et CV	Complexité accrue
[44]	BERT	CV et Offre d'emploi	Cout computationnel élevée

1.8 Conclusion

Les systèmes de recommandation sont des outils essentiels pour personnaliser les expériences utilisateurs dans divers domaines comme le commerce électronique, les plateformes de streaming, et les réseaux sociaux. Ils utilisent des algorithmes pour analyser les données et proposer des recommandations pertinentes basées sur les préférences des utilisateurs. Les approches courantes incluent le filtrage collaboratif, le filtrage basé sur le contenu, la recommandation basée sur la connaissance, et les systèmes hybrides. Avec les avancées récentes en traitement du langage naturel, notamment avec BERT, ces systèmes peuvent désormais mieux comprendre et traiter les informations contextuelles et sémantiques, améliorant ainsi la pertinence des recommandations. Cependant, malgré ces progrès, des défis subsistent, notamment en termes de qualité des données et de considérations éthiques liées à la confidentialité et à la transparence.

2 CHAPITRE 2 : La contribution

2.1 Introduction

Le développement des systèmes de recommandation d'emploi a considérablement évolué grâce à l'application des modèles de langage avancés tels que BERT. Ces modèles ont démontré une grande efficacité dans la compréhension et le traitement du langage naturel, ce qui est essentiel pour analyser les descriptions de postes et les profils des candidats. Cependant, l'intégration de ressources lexicales telles que WordNet peut enrichir davantage ces systèmes en ajoutant une dimension sémantique plus profonde à la compréhension des textes. Dans ce chapitre, nous décrivons notre contribution consistant à intégrer WordNet dans notre système de recommandation d'emploi basé sur BERT.

1. **Solution proposée**

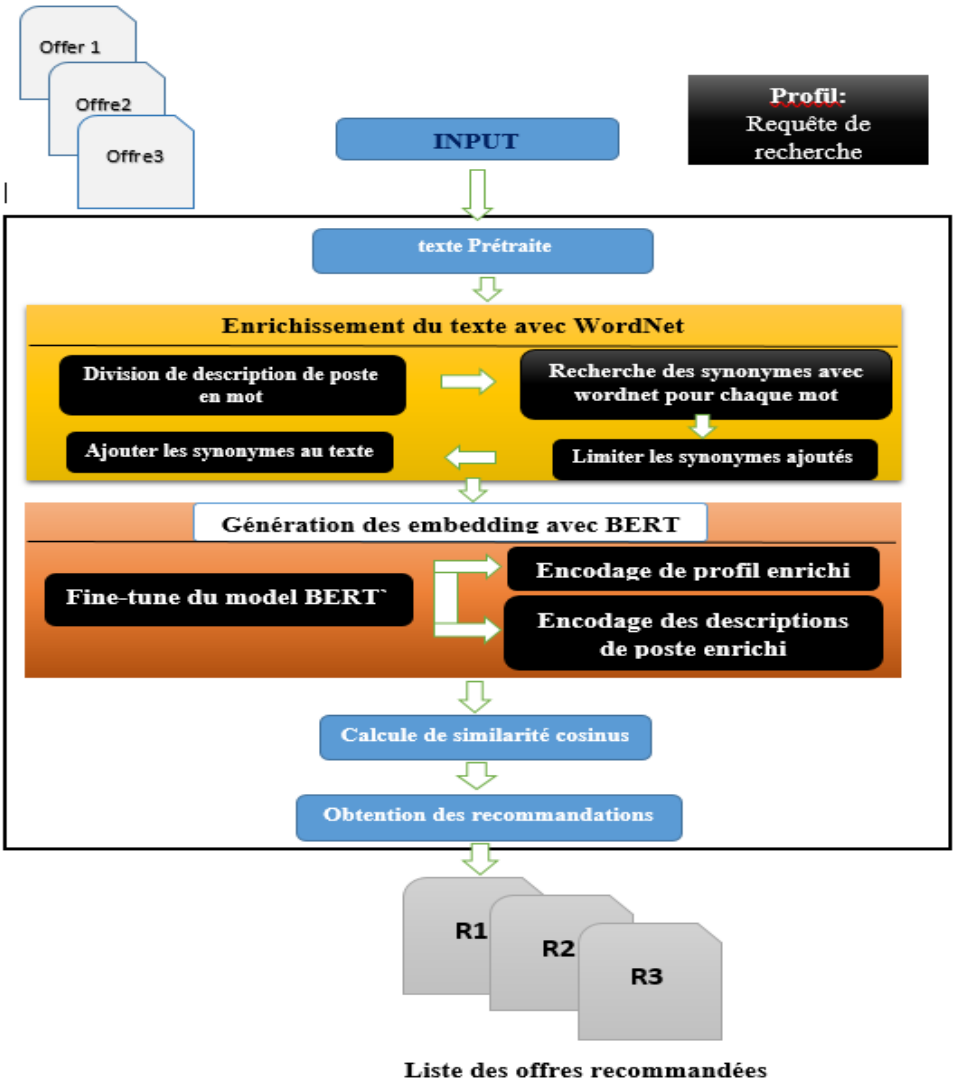


Figure 5 : Architecture générale

Pour notre système de recommandation d'emploi, nous allons exploiter WordNet pour enrichir les descriptions de poste et les profils des candidats. WordNet, étant une base de données lexicale regroupant des ensembles de synonymes, nous permettra d'ajouter des variations sémantiques aux termes utilisés dans les descriptions et les profils. Cette approche visera à améliorer la précision des recommandations en tenant compte des différentes formulations linguistiques possibles, augmentant ainsi les chances de faire correspondre des candidats qualifiés avec des offres d'emploi appropriées.

2.2 Presentation de wordnet

WordNet est une base de données lexicale de la langue anglaise qui regroupe les mots en ensembles de synonymes appelés synsets. Chaque Sunset exprime un concept distinct, et les relations entre les synsets, telles que la synonymie (synonymes), l'antonymie (antonymes), l'hyponymie (sous-types) et l'hypernymie (types), sont également définies [6]. Cette structure hiérarchique et relationnelle permet une meilleure compréhension des nuances sémantiques des termes, ce qui est particulièrement utile pour le traitement du langage naturel (NLP).

Les systèmes de recommandation d'emploi visent à identifier les opportunités les plus pertinentes pour les chercheurs d'emploi en fonction de leurs profils. L'un des défis majeurs dans ce domaine est de saisir la diversité des termes utilisés dans les descriptions de poste. Par exemple, une compétence telle que "programmation" peut être exprimée sous différentes formes comme "coding" ou "software development". Pour améliorer la précision des recommandations, nous avons intégré WordNet avec BERT pour enrichir les descriptions de poste avec des synonymes avant de les encoder et de calculer les similarités. Cette intégration vise à améliorer la précision des recommandations en capturant une plus grande diversité lexicale et en facilitant une meilleure correspondance entre les compétences des candidats et les offres d'emploi.

2.3 Motivation de l'intégration de WordNet dans BERT

L'intégration de WordNet dans un système basé sur BERT présente plusieurs avantages :

Amélioration de la précision sémantique : WordNet peut aider à désambiguïser les mots en fournissant des informations contextuelles sur leurs différents sens.

Enrichissement des représentations textuelles : En utilisant les relations sémantiques, nous pouvons étendre les représentations des mots et phrases au-delà des données d'entraînement initiales de BERT.

Réduction de l'ambiguïté lexicale : La capacité à reconnaître les synonymes et les termes sémantiquement liés peut améliorer la correspondance entre les descriptions de postes et les profils des candidats.

2.4 Méthodologie

Il existe plusieurs approches pour exploiter WordNet avec BERT pour la recommandation d'emploi comme celle des deux approches utilisées par les auteurs du [6]. Dans leurs articles utilisation de ces deux approches pour intégrer les synonymes de WordNet dans un modèle BERT pour enrichir les représentations du texte telle que : L'approche de concaténation et L'approche de fusion que nous allons décrire chaque approche suivie de leur schéma a la fin de leur description.

- **Approche de concaténation :**

Le texte d'entrée est fourni au modèle, après le texte passe à travers les 12 couches de BERT pour générer les embedding de phrase. BERT produit les embeddings contextuel pour chaque mot dans le texte.

WordNet Embedding :

Simultanément, les synonymes de chaque mot du texte sont obtenus à partir de WordNet puis encodés pour générer des embedding WordNet. Après cela les embeddings de phrase de BERT et les embeddings WordNet sont concaténées puis combine l'information sémantique des synonymes de WordNet.

2-Layer MLP :

Les embeddings concaténés passent par un perceptron multicouche (MLP) à 2 couches. Cette étape permet de faire des transformations linéaires suivie d'une activation non linéaire des embeddings combinés et d'apprendre des représentations plus riches et complexes du texte.

- **Pour le Couche 1** : Transformation linéaire suivie d'une activation non linéaire

Cette couche du MLP applique une transformation linéaire aux embeddings concaténés suivie d'une activation non linéaire. Elle permet de transformer les données d'entrée en une nouvelle représentation plus complexe.

Transformation linéaire :

Une transformation linéaire est effectuée en multipliant les embeddings concaténés par une matrice de poids $W1$ et en ajoutant un vecteur de biais $b1$.

Activation non linéaire :

Après la transformation linéaire, une fonction d'activation non linéaire est appliquée pour introduire de la non-linéarité dans le modèle, permettant ainsi de capturer des relations complexes dans les données.
Une activation couramment utilisées est la fonction ReLU (Rectified Linear Unit)

- **Couche 2** :

La deuxième couche applique une autre transformation linéaire aux résultats de la première couche suivie d'une autre activation non linéaire. Cette étape permet de continuer à transformer et enrichir les représentations des données.

Nouvelle transformation linéaire :

La deuxième transformation linéaire est effectuée en multipliant le vecteur $A1$ par une nouvelle matrice de poids $W2$ et en ajoutant un vecteur de biais $b2$.

Activation non linéaire :

Une autre fonction d'activation non linéaire est appliquée aux résultats de la deuxième transformation linéaire pour continuer à capturer des relations non linéaires dans les données.

La sortie finale est générée à partir de l'output MLP.

La sortie finale du modèle est générée à partir de la deuxième couche du MLP. Les résultats de cette couche représentent une représentation finale des données d'entrée après avoir été transformés et enrichis par le MLP.
Ainsi, la sortie finale est directement influencée par les embeddings concaténés initiaux, qui capturent à la fois les informations contextuelles de BERT et les informations sémantiques de WordNet.
En résumer, cette approche permet d'intégrer des informations sémantiques supplémentaires des synonymes de WordNet avec les puissants embeddings contextuels générés par BERT. En passant ces embeddings concaténés à travers un perceptron multicouche, le modèle peut apprendre des représentations encore plus riches et complexes du texte, ce qui améliore la précision et la pertinence des recommandations.

- **Approche de fusion :**

  Cette approche se diffère de la première approche après juste de l'entrée du texte toujours en basant sur l'utilisation de wordNet. Une description de cette approche étapes par étapes de l'entrée du texte a la sortie comment ils l'ont utilisée ?

Texte :

Cette étape consiste simplement à fournir le texte d'entrée au modèle sans aucune modification préalable. Cela marque le début du processus de traitement du texte.

BERT Word Embedding :

Pour chaque mot du texte, le modèle BERT est utilisé pour générer des embeddings de mots. BERT, un modèle de langage pré-entraîné basé sur l'attention, crée des représentations vectorielles pour chaque mot dans le contexte de la phrase. Ces embeddings capturent les nuances sémantiques et contextuelles des mots dans le texte. Il est crucial de noter que ces embeddings sont extraits avant les couches d'attention de BERT, fournissant ainsi des informations de base sur chaque mot.

WordNet Embedding :

Pour chaque mot du texte, les synonymes sont extraits de WordNet. Ces synonymes sont ensuite encodés pour générer des embeddings WordNet. Ces embeddings contiennent des informations sur les relations sémantiques entre les mots, offrant ainsi une perspective plus large sur le sens des mots dans le contexte du texte.

Vecteur de Concaténation :

Les embeddings de mots issus de BERT et de WordNet sont combinés en un seul vecteur d'embedding pour chaque mot. Cela est réalisé en concaténant les vecteurs d'embedding correspondant à un même mot. Par cette fusion, chaque embedding de mot est enrichi avec les informations sémantiques des synonymes, permettant ainsi de créer des représentations plus riches et complexes des mots dans le texte.

BERT Self-Attention / Multi-head Attention:

Les embeddings de mots enrichis (issus de la concaténation de BERT et de WordNet) passent à travers les couches d'attention de BERT. Ces couches d'attention permettent au modèle d'analyser la structure et le contexte du texte, en accordant une attention différentielle aux différents mots en fonction de leur pertinence dans le contexte. Cela permet de capturer les relations contextuelles entre les mots et de créer des représentations contextuelles enrichies du texte.

Output Layer :

Après avoir traversé les couches d'attention de BERT, la sortie finale est produite. Cette sortie représente une représentation contextuelle profondément enrichie du texte d'entrée. En intégrant à la fois les informations de base de BERT et les informations sémantiques des

synonymes de WordNet, cette représentation finale capture la complexité et la nuance du texte d'origine. Cette sortie peut ensuite être utilisée pour diverses tâches telles que la recommandation, la classification, etc.
Chaque étape de ce processus contribue à enrichir les représentations de mots et à capturer les informations contextuelles du texte, aboutissant à une sortie finale qui reflète une compréhension approfondie et nuancée du texte d'entrée.

Dans notre cas nous avons proposé d'enrichir le texte d'abord avec wordNet puis faire les embedding du texte enrichi. L'enrichissement du texte avec wordNet avant de procéder a l'embedding permet en fin d'améliorer la qualité de représentation vectorielles en fournissant un contexte sémantique plus riche et en réduisant l'ambiguïté des mots.

Illustration des différentes étapes du pipeline, de l'entrée des descriptions de profil et de poste jusqu'aux recommandations finales et chaque étape est suivi de son exemple :

1. **Entrée (Input): Profil et description d'emploi**
    - Descriptions de profil et de poste en entrée.

Les données d'entrée se composent de descriptions de profil et descriptions des postes. Il est important de noter que, pour notre étude, les profils utilisés dans notre système sont sous forme de requêtes de recherche d'emploi, avec une perspective future d'utilisation des CV traditionnels.

a. Profils sous forme de requêtes de recherche d'emploi

Pour les besoins de cette étude expérimentale, nous utilisons les requêtes de recherche d'emploi des utilisateurs au lieu des CV traditionnels. Les requêtes sont des entrées textuelles simples, fournissant une vue concise et directe des intentions de recherche des emplois. La motivation de l'utilisation de cette approche, elle présente des avantages pour la phase de notre recherche :

- L'utilisation des requêtes de recherche d'emploi simplifie le processus de collecte des données et permet de tester le système de manière plus flexible. Cela réduit les barrières à la participation des utilisateurs et facilite une collecte rapide des données nécessaires.
- Elles reflètent directement les intérêts et objectifs actuels des candidats, offrant une vue précise et actualisée de leurs attentes professionnelles.

- En utilisant des requêtes, nous pouvons rapidement ajuster et affiner notre système de recommandation. Cela permet d'optimiser l'algorithme avant de passer à une phase où les CV seraient utilisés.

Exemple de requete de recherche : "Developer with experience in python and java".

b. Descriptions de poste

Les descriptions de poste fournissent des informations détaillées sur les expériences, les localisations, les compétences nécessaires, et d'autres critères spécifiques à chaque poste. Chaque description est sous forme de texte semi-structuré.

Exemple :

2. **Enrich Text with Synonyms (WordNet)**
    - Les descriptions de texte sont enrichies avec des synonymes en utilisant WordNet pour améliorer le contexte et la diversité des termes.

Identification des mots-clés : Chaque description de poste est décomposée en mots individuels.

Recherche des synonymes : Pour chaque mot identifié, ses synonymes sont recherchés dans le WordNet.

Sélection des synonymes pertinents : Bien que WordNet puisse fournir plusieurs synonymes pour un mot donné, tous ne sont pas pertinents dans le contexte d'une description de poste. Pour éviter d'introduire trop de bruit et maintenir la clarté du texte, nous limitons le nombre de synonymes ajoutés. Par exemple, nous pouvons choisir d'ajouter un maximum de trois synonymes par mot, mais dans notre cas spécifique, nous avons opté pour un maximum d'un synonyme afin de contrôler la quantité d'informations supplémentaires.

Enrichissement du texte : Les synonymes sélectionnés sont ensuite intégrés dans le texte original. Chaque mot est suivi par ses synonymes choisis, ce qui crée une version enrichie du texte qui conserve les mots originaux tout en ajoutant des alternatives lexicales. Cela permet aux modèles de traitement du langage naturel de mieux comprendre et associer les compétences et expériences décrites.

Exemple:

- Considérons une description de poste simple: "The candidate should have experience in programming and software development."

- nous enrichissons le texte comme suit: "The candidate should have experience in programming **coding** and software development **software engineering**."

3. **Tokenization (BERT Tokenizer)**

Une fois le texte enrichi avec des synonymes, Nous utilisons le tokenizer spécifique à BERT, qui inclut des étapes comme la segmentation des mots, la gestion des sous-mots (WordPiece tokenization), et l'ajout de tokens spéciaux nécessaires pour BERT ([CLS], [SEP]).

Le texte enrichi est converti en une séquence de tokens. Par exemple, une phrase comme "Developer with experience in python and java" pourrait être tokenisée en "**[** Developer, with, experience, in, Python, and, Java **]**"

Les séquences de tokens sont ajustées pour correspondre à la longueur maximale acceptée par BERT, avec des remplissages (padding) ou des troncations si nécessaires.

4. **Embedding (BERT Initial Embeddings)**

Une fois le texte tokenisé, chaque token est transformé en un vecteur d'embedding initial basé sur les tables d'embeddings pré-entraînées de BERT. Qui sont des représentations vectorielles des tokens, capturant des informations contextuelles et syntaxiques.

Ces embeddings initiaux capturent des informations sur les tokens et leur position dans la séquence.

Les embeddings de position sont ajoutés aux embeddings de tokens pour incorporer des informations sur la position des tokens dans la séquence, ce qui est crucial pour capturer la structure de la phrase.

- Exemple d'embeddings initiaux : "[0.25, 0.1, ...]"

5. **Contextual Encoding (BERT Transformer Layers)**

Les embeddings initiaux traversent plusieurs couches de transformateurs. Chaque couche applique des mécanismes d'attention et des transformations non linéaires pour raffiner les représentations vectorielles.

Les mécanismes d'attention de BERT permettent de capturer les relations à longue distance entre les tokens, produisant des embeddings contextuels finaux riches en informations sémantiques et contextuelles.

- Exemple de vecteurs contextuels : « [0.35, 0.2, ...] »

6. **Profile and Job Description Embeddings (Contextual Vectors)**

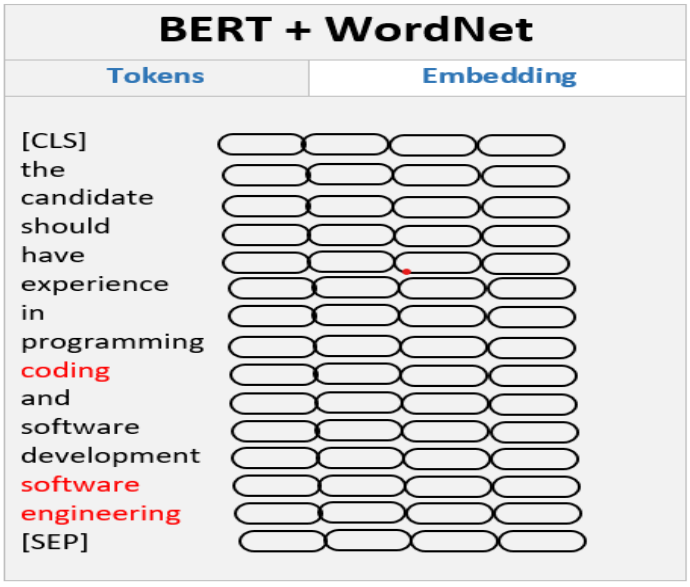
   - Les descriptions de profil et de poste sont maintenant représentées par des vecteurs contextuels denses.

Les embeddings contextuels générés pour les requêtes de recherche d'emploi et les descriptions de poste sont utilisés comme représentations finales des profils et des postes.

Après le passage par les couches de transformateurs de BERT, les vecteurs contextuels finaux sont extraits pour chaque token dans les séquences de requêtes et de descriptions de poste.

Les vecteurs contextuels peuvent être agrégés (par exemple, en utilisant la moyenne des vecteurs) pour obtenir une représentation unique et globale de la requête ou de la description de poste. Le token [CLS] peut aussi être utilisé comme vecteur représentant la séquence entière.

- Exemple des embeddings de description de poste enrichi: « the candidate should have experience in programming coding and software develpement software engineering»



7. **Cosine Similarity Calculation**

Pour mesurer la similarité entre les vecteurs contextuels des profils et des descriptions de poste, nous utilisons la similarité cosinus. Cette mesure évalue l'angle entre deux vecteurs dans l'espace vectoriel, permettant de quantifier leur similarité sémantique.

La similarité cosinus entre deux vecteurs A et B est définie comme :

$$similarite\ cosinus = \frac{A\ .\ B}{\|A\|\|B\|}$$

Où $A.B$est le produit scalaire des vecteurs A et B, et $\|A\|$ et $\|B\|$sont les normes de ces vecteurs.

Pour chaque paire de vecteurs (profil, description de poste), la similarité cosinus est calculée. Les vecteurs sont normalisés et leur produit scalaire est divisé par le produit de leurs normes.

Le rang des correspondances : Les postes sont triés en fonction de leur similarité cosinus avec le profil de recherche. Les postes ayant les valeurs les plus élevées de similarité cosinus sont considérés comme les meilleures correspondances.

- Exemple de scores de similarité : « 0.95, 0.85, 0,84... »

8. **Top des emplois recommandés**

   - Les postes les plus similaires sont sélectionnés et recommandés en fonction des scores de similarité.

Les emplois les plus pertinents, identifiés par leur similarité cosinus élevée avec les profils de recherche, sont ensuite recommandés aux utilisateurs.

Identification des meilleures correspondances : Les descriptions de poste ayant les scores de similarité cosinus les plus élevés avec les requêtes de recherche d'emploi sont sélectionnées comme les meilleures correspondances.

Les recommandations d'emploi sont présentées aux utilisateurs sous forme de liste, classée par ordre décroissant de similarité. Chaque recommandation comprend des détails sur le poste, tels que le titre, la société, l'emplacement, et un résumé des responsabilités et exigences.

Exemple de recommandations :

Profil 1:"Developer with experience in python and java"

- 1. Python Developer, similarité : 0.95
- 2. Java Developer, similarité : 0.85

Ces étapes sont les processus de traitements de texte entrée (profil et description de poste) enrichi à l'aide de WordNet avec de modèle BERT de les entrée(input) à la sortie (output) des listes des emplois correspond au profil du candidat en question à base de similarité.

À long terme, l'objectif est de passer à l'utilisation des CV traditionnels pour enrichir les profils des candidats et améliorer encore la précision des recommandations d'emploi. En commençant par les requêtes de recherche, nous pouvons valider notre approche et obtenir des retours précieux avant de complexifier le système avec des données de CV plus détaillées.

Description du dataset utilisée

Nous décrivons en détail les étapes de collecte et de préparation des données nécessaires pour le développement de notre système de recommandation d'emploi. Un soin particulier a été apporté à la qualité des données, car elle est cruciale pour le succès du modèle de recommandation. Voici les sous-sections spécifiques qui seront couvertes :

Sources de Données

Les données utilisées dans ce projet proviennent du source Kaggle un dataset complet sous forme d'un fichier de type CSV à télécharger.

Kaggle : est une plateforme bien connue pour ses vastes collections de datasets utilisés dans divers domaines d'analyse de données et de machine Learning. Le dataset de recommandations d'emploi sur Kaggle fournit des informations précieuses pour l'analyse et le développement de systèmes de recommandation dans le domaine de l'emploi et aussi pour autre type de système.

Nettoyage des Données

Le nettoyage des données est une étape essentielle pour garantir la qualité et la cohérence des données. Les opérations suivantes ont été effectuées :

Suppression de ponctuation :

La ponctuation peut introduire du bruit dans les analyses textuelles, ce qui peut affecter la qualité des résultats. En particulier, les descriptions de poste et les profils des candidats peuvent contenir divers signes de ponctuation qui ne sont pas pertinents pour les analyses de texte. Pour remédier à cela, toutes les ponctuations sont identifiées et supprimées des textes. Cela inclut les points, virgules, points d'exclamation, parenthèses, etc. Cette suppression permet de normaliser les textes et de faciliter le traitement ultérieur, comme le comptage de mots et l'analyse sémantique.

Suppression de caractéristiques qui ne sont pas important :

Afin d'optimiser la performance du modèle de recommandation, il est essentiel d'éliminer les caractéristiques qui n'apportent pas de valeur significative. Les méthodes utilisés pour identifier et supprimer ces caractéristiques non pertinentes.

Les caractéristiques jugées non pertinentes dans le contexte de la recommandation d'emploi (ex : informations trop spécifiques ou non liées aux critères de sélection).

Initialement, une analyse exploratoire des données a été menée pour identifier les caractéristiques potentiellement non pertinentes.

La suppression des caractéristiques non importantes a permis de simplifier le modèle et d'améliorer son efficacité. En se concentrant sur les attributs les plus pertinents, le système de recommandation d'emploi peut offrir des recommandations plus précises et fiables, tout en réduisant la complexité computationnelle.

Cette étape de nettoyage et de réduction des données est cruciale pour garantir que le modèle utilise uniquement les informations les plus utiles, contribuant ainsi à des performances optimales dans le cadre de la recommandation d'emploi.

Gestion des valeurs manquantes :

Les valeurs manquantes peuvent poser des problèmes importants pour l'analyse des données et la formation des modèles de recommandation. Pour notre cas plusieurs attribues ont des valeurs manquantes mais beaucoup de valeurs manquantes se trouvaient avec les attribues qui n’apportent pas importance pour l’entrainement donc on les a supprimés, ce qui ont le moins de valeurs manquantes ses échantillons ont été supprimées.

Normalisation des formats : Les formats de dates, expérience, et d'autres valeurs ont été normalisés pour assurer la cohérence à travers le dataset.

Description de dataset

Le dataset final utilisé pour l'entraînement du modèle de recommandation contient les informations suivantes : Nombre total d’entrées sont 19077 offres d'emploi composé de 9 attributs.

Les attributs disponibles:

Attribue	Description
Jobid	Identification de chaque offre d'emploi
job_description	Description du poste
education	Education scolaire
Jobtitle	Titre de l'offre
Skills	Compétences
Industry	L'industrie
Experience	Expérience (années)
Joblocation _address	Localité
Company	La compagnie

Table 1 : Tableau des caracteristiques retenue

1.1.1 Analyse Exploratoire des Données

Une analyse exploratoire des données a été effectuée pour mieux comprendre les caractéristiques et la distribution des données. C'est est une étape essentielle dans tout processus d'analyse et de modélisation des données. Elle permet de mieux comprendre les caractéristiques et la distribution des données avant de passer à des analyses plus avancées. En examinant la structure des données, en identifiant les tendances et les modèles, ainsi qu'en repérant les besoins en prétraitement, l'AED fournit une base solide pour une analyse approfondie et des prises de décision éclairées. Dans cette optique, cette étude a entrepris une AED pour explorer en profondeur les données et en tirer des insights pertinents pour la suite de l'analyse.

- **Distribution des types d'emploi** : Un diagramme à barres a été utilisé pour montrer la répartition des différents types d'emplois

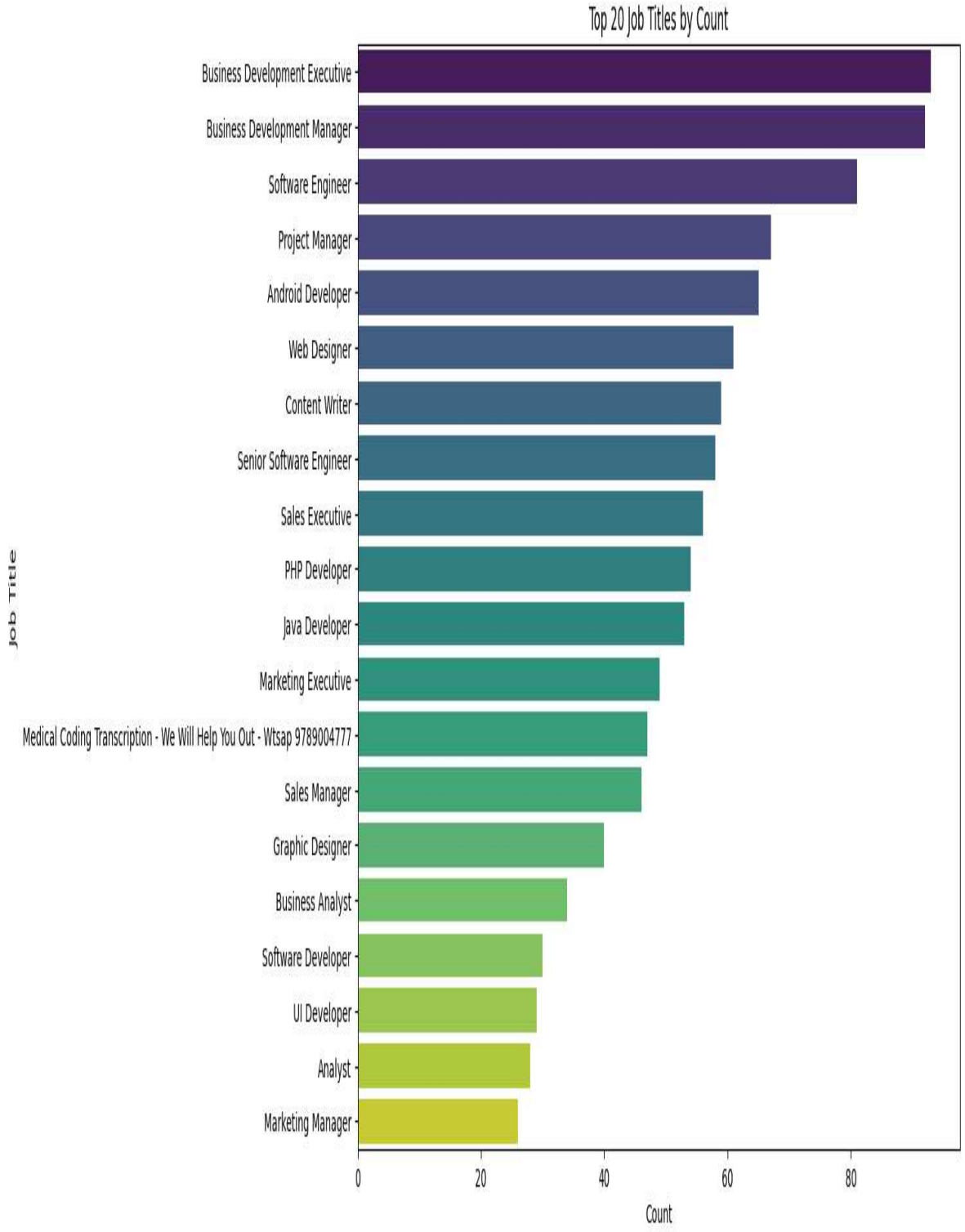


Figure : top 20 de titre des emplois

Le graphique présente une analyse des titres de postes les plus fréquents dans un ensemble de données d'offres d'emploi. L'axe horizontal montre le nombre d'occurrences de chaque titre de poste, tandis que l'axe vertical liste les différents titres de poste identifiés.

Ce graphique à barres horizontales montre les 20 titres de poste les plus courants dans les offres d'emploi. Les titres sont classés de manière décroissante en fonction de leur fréquence d'apparition.

La description de ces données :

- **Dominance des Rôles de Développement des Affaires** :

Les deux premiers postes les plus fréquents sont liés au développement des affaires. Cela montre que de nombreuses offres d'emploi sont centrées sur l'expansion et la gestion des relations commerciales.

- **Abondance des Postes Techniques** :

Plusieurs titres de poste concernent le développement de logiciels et la gestion de projets technologiques. Cela inclut des rôles comme ingénieur logiciel, développeur mobile et chef de projet. Cette tendance indique une forte demande pour les compétences techniques et le développement technologique.

- **Présence de Rôles Créatifs** :

Les titres de postes incluent également des rôles dans la création de contenu et le design. Des titres comme concepteur web et rédacteur de contenu sont bien représentés, soulignant l'importance de la création et de la gestion de contenu visuel et textuel.

- **Rôles de Vente et Marketing** :

  Plusieurs postes sont orientés vers les ventes et le marketing, reflétant l'importance de ces fonctions dans les entreprises. Cela inclut des rôles comme directeur des ventes et responsable marketing.

- **Visualisation des industries :** la figure 2 ci-dessous présente une analyse des industries les plus fréquentes dans un ensemble de données d'offres d'emploi.

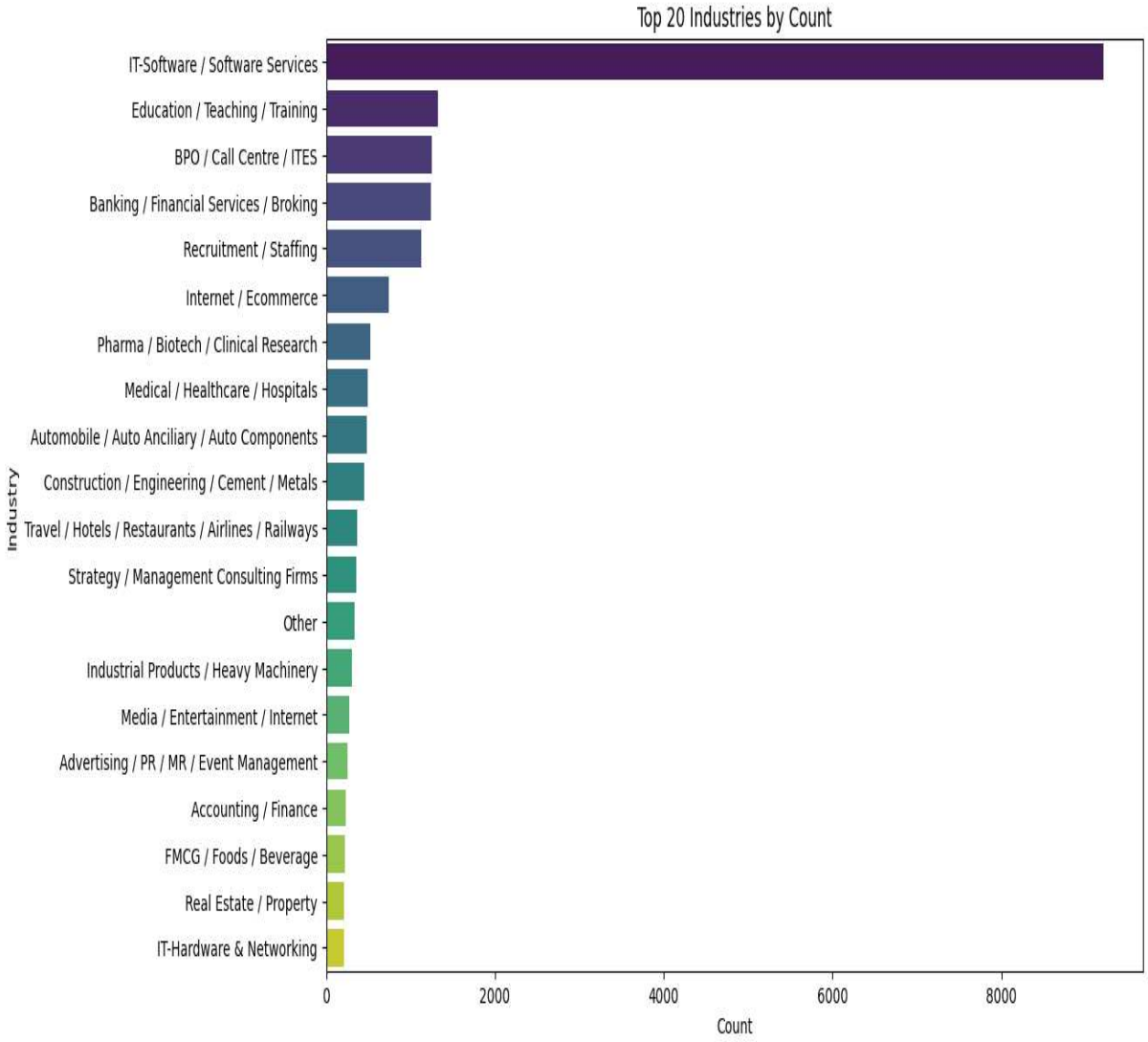


Figure: top 20 des industries

L'axe horizontal montre le nombre d'occurrences pour chaque industrie, tandis que l'axe vertical liste les différentes industries identifiées. Voici une explication détaillée de cette analyse.

Ce diagramme montre les 20 industries les plus courantes dans les offres d'emploi. Les industries sont classées de manière décroissante en fonction de leur fréquence d'apparition.

Description

- **Dominance de l'Industrie IT et des Services Logiciels** :
  - L'industrie IT et les services logiciels dominent largement le classement avec un nombre d'occurrences très élevé plus de 9000, indiquant une forte demande pour des postes dans ce secteur.
- **Importance de l'Éducation et des Centres d'Appels** :

- Les secteurs de l'éducation et de la formation, ainsi que les centres d'appels et les services de BPO, sont également bien représentés, montrant une demande notable pour des professionnels dans ces domaines.

❖ **Secteurs Financiers et de Recrutement** :

- Les services financiers, bancaires, et de courtage, ainsi que le recrutement et les services de staffing, apparaissent fréquemment, ce qui souligne la nécessité de professionnels dans la gestion financière et le recrutement.

❖ **Diversité des Secteurs** :

- D'autres secteurs comme le commerce électronique, la recherche clinique, la santé, et l'automobile sont également représentés, indiquant une demande variée dans plusieurs industries.

❖ **Large Variété de Secteurs avec Moins d'Occurrences** :

- Plusieurs autres secteurs comme les médias, l'ingénierie, la construction, et le FMCG sont présents mais avec des occurrences relativement plus faibles, montrant une diversité d'opportunités dans différents domaines industriels.

➢ **Visualisation des compétences en word cloud** : qui est une méthode visuelle pour représenter la fréquence ou l'importance des mots dans un texte donné. Les mots les plus fréquents apparaissent plus grands et plus visibles.

Figure : Word Cloud par compétences

Description :

- **Principales Compétences Identifiées :**
  - **Software Application** : C'est la compétence la plus dominante, indiquant une forte demande ou une présence fréquente dans le dataset.
  - **Application Programming** : Également très visible, ce qui souligne l'importance du développement d'applications dans les données.
  - **Programming** : Présente en différentes variantes, indiquant que les compétences en programmation sont cruciales.
  - **Sales** : La compétence en vente est également très visible, montrant son importance dans ce dataset.
- **Autres Compétences Importantes :**
  - **Network Administration**
  - **Software QA (Quality Assurance)**
  - **Teaching** : Indique une demande ou une importance dans les compétences pédagogiques.
  - **Engineering** : Généralement, les compétences en ingénierie sont bien représentées.
  - **Medical** : Signale que les compétences dans le domaine médical sont également pertinentes.
  - **Marketing** : Compétences en marketing sont bien représentées, montrant leur importance dans les données.
- **Diversité des Compétences :**
  - La word cloud montre une grande diversité de compétences, couvrant différents domaines comme le médical, le marketing, l'enseignement, les ventes, et diverses spécialités en informatique et ingénierie.
  - La présence de termes comme **QA Testing**, **Embedded Software**, **eCommerce**, **ERP**, et **DBA** indique une variété de spécialisations techniques recherchées.
- **Insights pour le Système de Recommandation d'Emploi :**
  - Un système de recommandation efficace devrait prendre en compte cette diversité et la fréquence des compétences.

- Les compétences les plus visibles dans la word cloud devraient être prioritaires dans les algorithmes de correspondance de profils.

En resumant ces analyses : L'analyse des offres d'emploi montre que les titres les plus courants sont orientés vers le développement des affaires, la technologie et la vente, avec une forte demande pour des rôles tels que "Business Development Executive" et "Software Engineer". Les opportunités d'emploi sont largement concentrées dans quelques grandes villes, et l'industrie IT domine dans ce dataset, suivie par l'éducation et les centres d'appels. Ces insights sont essentiels pour développer un système de recommandation d'emploi efficace, en ciblant les secteurs et les localisations les plus dynamiques. Cette word cloud révèle les compétences les plus présentes et les plus importantes dans le dataset. Il est crucial de s'assurer que les compétences dominantes comme "Software Application", "Application Programming", et "Sales" sont bien représentées et correctement pondérées dans notre modèle.

2.5 Conclusion :

L'intégration de WordNet avec BERT pour l'enrichissement des descriptions de poste représente une avancée notable dans le domaine des systèmes de recommandation d'emploi. Cette approche permet de surmonter les défis posés par la diversité lexicale et améliore la précision des recommandations. Enrichir les données textuelles avec des synonymes avant l'encodage par BERT a démontré son efficacité pour offrir des recommandations plus pertinentes et adaptées aux chercheurs d'emploi. Cette méthodologie peut être étendue à d'autres domaines où la compréhension des nuances sémantiques est cruciale pour la pertinence des résultats.

3 Chapitre 3 : ImplémentationEt Evaluation

3.1 Introduction :

Dans ce chapitre, il contiendra une description détaillée du dataset utilisé pour développer notre système de recommandation d'emploi, accompagnée d'une analyse exploratoire approfondie des données. Ce chapitre présentera également les étapes clés de l'implémentation du système, illustrées par des captures d'écran des sections importantes du code pour faciliter la compréhension technique.

Nous aborderons les différentes phases du processus de développement, depuis la collecte et la préparation des données jusqu'à l'entraînement et l'évaluation du modèle de recommandation. Pour cela, nous fournirons une évaluation comparative des résultats obtenus avant et après l'intégration de WordNet, mettant en lumière les améliorations apportées par cette approche.

Ce chapitre vise à offrir une vue complète et détaillée de l'implémentation technique de notre système, permettant ainsi de comprendre les choix méthodologiques et les défis rencontrés tout au long du développement.

3.2 Implémentation et l'évaluation du Système

3.2.1 Entrainement du modèle

Pour la phase d'entraînement des modèles, plusieurs étapes cruciales ont été suivies afin de préparer les données et d'optimiser les performances des modèles de classification.

- **Division du Jeu de Données**

Le jeu de données a été divisé en ensembles d'entraînement et de validation pour évaluer les performances des modèles de manière plus objective. Une proportion de 20% des données a été réservée pour la validation, garantissant ainsi que les modèles ne soient pas évalués sur les mêmes données utilisées pour l'entraînement.

- **Préparation des Données**

Pour préparer les données pour l'entraînement des modèles basés sur des réseaux de neurones, une classe personnalisée **ResumeDataset** a été créée. Cette classe permet de gérer le texte et les étiquettes associées, en utilisant un tokenizer pour convertir les textes en séquences de tokens adaptées aux modèles de traitement du langage naturel. La classe **ResumeDataset** assure également le padding et la troncature des séquences pour qu'elles soient de longueur uniforme, facilitant ainsi leur traitement par les modèles.

```python
# ResumeDataset class
class ResumeDataset(Dataset):
    def __init__(self, texts, labels, tokenizer, max_length=128):
        self.texts = texts
        self.labels = labels
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, idx):
        text = str(self.texts.iloc[idx])
        label = int(self.labels.iloc[idx])
        encoding = self.tokenizer(text, truncation=True, padding='max_length', max_length=self.max_length, return_tensors='pt')
        return {'input_ids': encoding['input_ids'].squeeze(), 'attention_mask': encoding['attention_mask'].squeeze(), 'labels': torch.tensor(label)}
```

Figure 5: Model de training

- **Création des Datasets et Dataloaders**

Les ensembles d'entraînement et de validation ont été convertis en instances de **ResumeDataset**, puis encapsulés dans des DataLoader pour gérer le batching pendant l'entraînement.

```python
# Creating datasets and dataloaders
train_dataset = ResumeDataset(train_df['jobdescription_clean'], train_df['industry'], tokenizer)
val_dataset = ResumeDataset(val_df['jobdescription_clean'], val_df['industry'], tokenizer)

train_dataloader = DataLoader(train_dataset, batch_size=32, shuffle=True)
val_dataloader = DataLoader(val_dataset, batch_size=32, shuffle=False)
```

Figure 6 : division de jeu de donnees pour l'entrainement et validation

- **Fine-tuning du Modèle BERT**

Le fine-tuning du modèle BERT sur un dataset d'entraînement avec un GPU sur 5 epochs et l'optimisation via rétropropagation constitue une approche méthodiquement solide. L'utilisation d'un GPU accélère le processus, le choix des epochs assure une balance entre sur-apprentissage et sous-apprentissage, et la rétropropagation garantit que les poids du modèle sont optimisés de manière efficace pour la tâche cible.

```python
# Fine-tuning the BERT model
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model.to(device)

for epoch in range(epochs):
    model.train()
    total_loss = 0

#Batch Iteration:
    for batch in train_dataloader:
        input_ids = batch['input_ids'].to(device)   #Move to GPU:
        attention_mask = batch['attention_mask'].to(device)
        labels = batch['labels'].to(device)


        optimizer.zero_grad()     #Zeroing Gradients and Forward Pass
        outputs = model(input_ids, attention_mask=attention_mask, labels=labels)
        loss = outputs.loss     #Backward Pass and Optimization
        loss.backward()
        optimizer.step()

    total_loss += loss.item()
    average_loss = total_loss / len(train_dataloader)
    print(f'Epoch {epoch + 1}/{epochs}, Loss: {average_loss}')
```

Figure 7 : Fine-tune de model BERT

- **Évaluation du Modèle sur l'Ensemble de Validation**

Après l'entraînement, le modèle a été évalué sur l'ensemble de validation pour mesurer sa précision. Les prédictions ont été comparées aux étiquettes réelles pour calculer l'exactitude.

```python
[ ] # Model Evaluation on the validation set
    model.eval()
    predictions = []
    true_labels = []

    with torch.no_grad():
        for batch in val_dataloader:
            input_ids = batch['input_ids'].to(device)
            attention_mask = batch['attention_mask'].to(device)
            labels = batch['labels'].to(device)

            outputs = model(input_ids, attention_mask=attention_mask)
            logits = outputs.logits
            predictions.extend(torch.argmax(logits, dim=1).cpu().numpy())
            true_labels.extend(labels.cpu().numpy())
    # Calculating Accuracy on the validation set
    accuracy = accuracy_score(true_labels, predictions)
    print(f'Validation Accuracy: {accuracy}')
```

```python
# Calculating Accuracy on the validation set
accuracy = accuracy_score(true_labels, predictions)
print(f'Validation Accuracy: {accuracy}')
```

```
Validation Accuracy: 0.7885474126608878
```

Figure 8: Evaluation

Le modèle a atteint une précision de validation de 78,85%, ce qui indique une performance solide pour la tâche de classification de poste.

3.2.2 Evaluation avec NDCG

Mesure de la Discounted Cumulative Gain normalisée (NDCG) s'est révélée être un outil puissant pour évaluer la pertinence des recommandations par rapport à des données de vérité terrain (ground truth data). Il est particulièrement adapté à cette tâche car il prend en compte non seulement la pertinence des recommandations mais aussi leur position dans la liste des résultats, favorisant ainsi les recommandations pertinentes en haut de la liste.

Dans notre Il est basé sur deux concepts principaux : le Discounted Cumulative Gain (DCG) et Ideal Discounted Cumulative Gain (IDCG).

Discounted Cumulative Gain (DCG) :

Le DCG améliore le CG en tenant compte de la position des emplois dans la liste de recommandations. Plus un emploi pertinent apparaît en haut de la liste, plus sa contribution au score est élevée. Le DCG à la position k est défini comme :

$$DCG_k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{log_2(i+1)}$$

Cette formule applique une pénalité logarithmique en fonction de la position, ce qui reflète la diminution de la valeur des documents pertinents apparaissant plus loin dans la liste.

Ideal Discounted Cumulative Gain : est utilisée pour normaliser la Discounted Cumulative Gain (DCG) afin d'obtenir la Normalized Discounted Cumulative Gain (NDCG). L'IDCG représente le DCG maximum possible pour un ensemble de résultats, c'est-à-dire

Pour calculer l'IDCG, il faut d'abord trier les emplois par pertinence décroissante. Ensuite, on applique la formule du DCG sur cette liste triée.

La formule de l'IDCG à une position k est identique à celle de la DCG, mais appliquée à la liste de résultats triée par pertinence décroissante :

$$IDCG_k = \sum_{i=1}^{k} \frac{2^{rel_i'} - 1}{log_2(i+1)}$$

Où rel' est la pertinence des documents triés par pertinence décroissante.

Normalized Discounted Cumulative Gain (NDCG)

$$NDCG_k = \frac{DCG_k}{IDCG_k}$$

Application du NDCG à l'Évaluation des Systèmes de Recommandation d'Emploi, les étapes suivantes sont suivies :

Collecte des données de vérité terrain (ground truth data) : Il s'agit des données réelles sur lesquelles l'évaluation sera basée. Ces données ont été classées manuellement par le tops 20 recommandations par système pour chaque profil puis la sélection des tops 10 d'emplois et assigner une pertinence numérique à chaque recommandation en fonction de la vérité terrain. Ses caracteristiques sont : le numéro du profil, identification de poste, la pertinence et le titre du poste. On attribue les numéros 0,1,2,3 comme une désignation de degré de pertinence par exemple :

3 : très pertinent

2 : pertinent

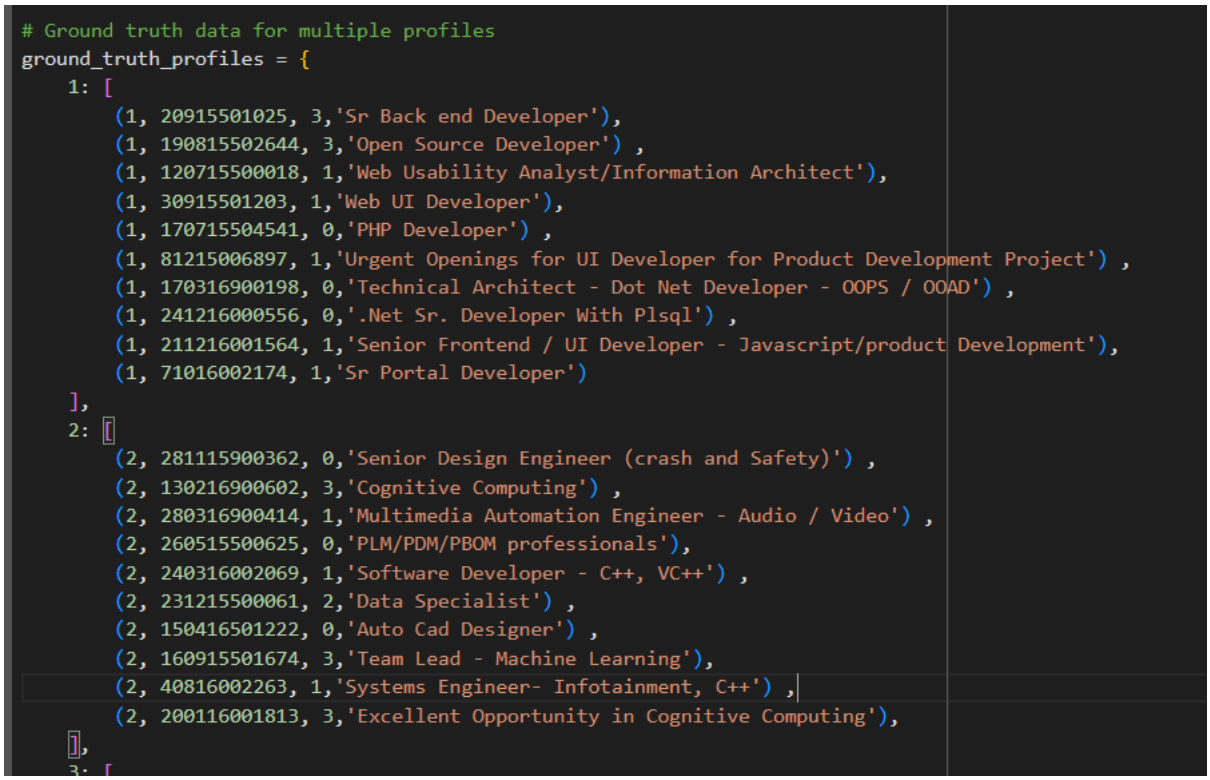
1 : peu pertinent

0 : non pertinent

L'annotation de pertinence de chaque offre est basée sur le score de similarité cosinus de recommandation, comme ceci :

- Si similarite est >= 0.8 la pertinence (3)
- 0.6 < similarite < 0.8 la pertinence (2)
- 0.4 < similarite < 0.6 la pertinence (1)
- Similarite < 0.4 la pertinence (0)

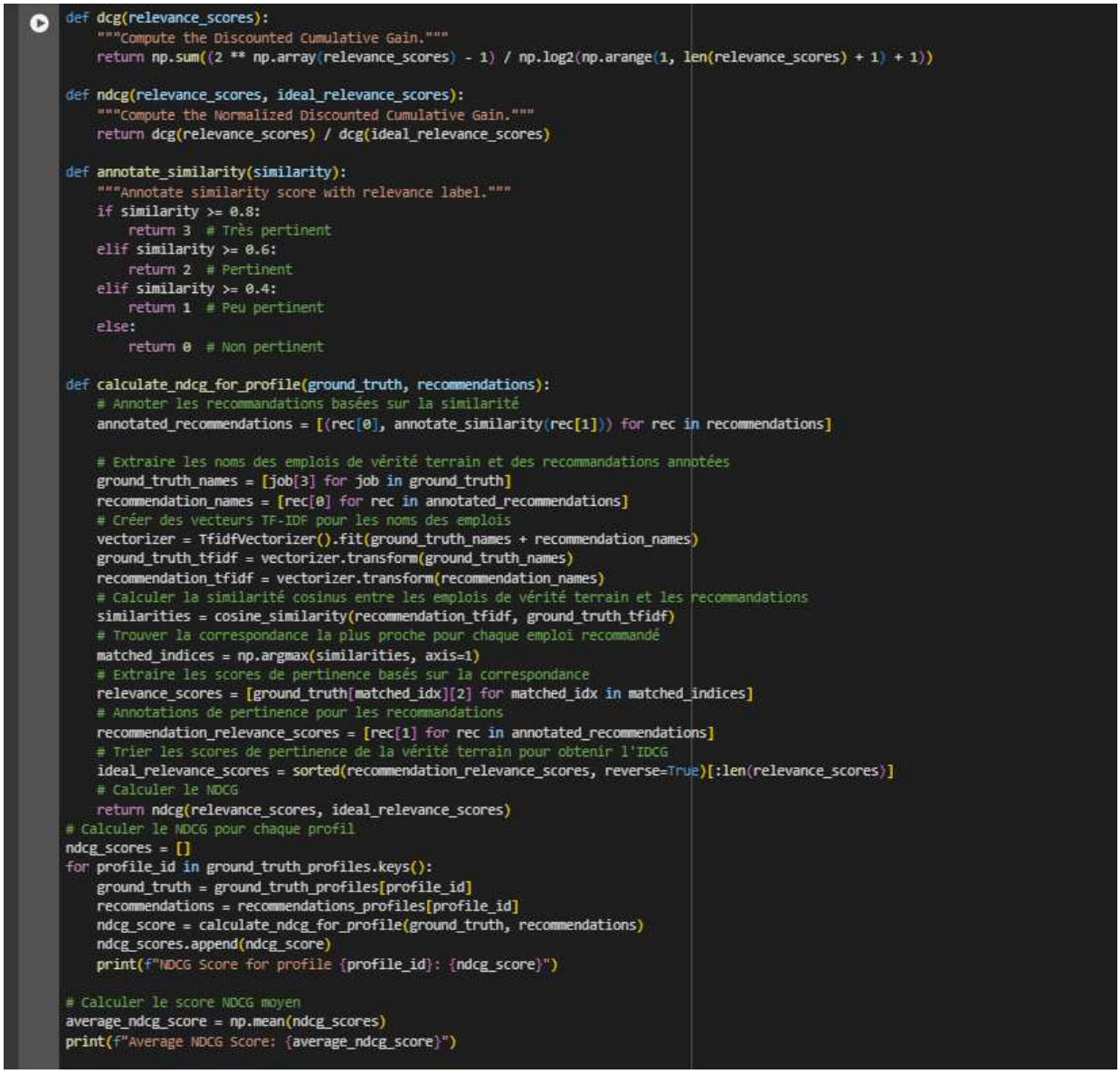
Exxemple de ground truth data des 2 profils sur 10 profils utilisée ainsi leur calcule NDCG pour BERT seul et BERT + WordNet voir la figure ci-dessous :



```
# Ground truth data for multiple profiles
ground_truth_profiles = {
    1: [
        (1, 20915501025, 3,'Sr Back end Developer'),
        (1, 190815502644, 3,'Open Source Developer') ,
        (1, 120715500018, 1,'Web Usability Analyst/Information Architect'),
        (1, 30915501203, 1,'Web UI Developer'),
        (1, 170715504541, 0,'PHP Developer') ,
        (1, 81215006897, 1,'Urgent Openings for UI Developer for Product Development Project') ,
        (1, 170316900198, 0,'Technical Architect - Dot Net Developer - OOPS / OOAD') ,
        (1, 241216000556, 0,'.Net Sr. Developer With Plsql') ,
        (1, 211216001564, 1,'Senior Frontend / UI Developer - Javascript/product Development'),
        (1, 71016002174, 1,'Sr Portal Developer')
    ],
    2: [
        (2, 281115900362, 0,'Senior Design Engineer (crash and Safety)') ,
        (2, 130216900602, 3,'Cognitive Computing') ,
        (2, 280316900414, 1,'Multimedia Automation Engineer - Audio / Video') ,
        (2, 260515500625, 0,'PLM/PDM/PBOM professionals'),
        (2, 240316002069, 1,'Software Developer - C++, VC++') ,
        (2, 231215500061, 2,'Data Specialist') ,
        (2, 150416501222, 0,'Auto Cad Designer') ,
        (2, 160915501674, 3,'Team Lead - Machine Learning'),
        (2, 40816002263, 1,'Systems Engineer- Infotainment, C++') ,
        (2, 200116001813, 3,'Excellent Opportunity in Cognitive Computing'),
    ],
    3: [
```

Figure 9 : Ground truth data

Avec ses annotations de manuelle de10 profils, le NDCG est calculer avec celle des emplois recommandés par le système à base de similarité de chaque emploi. Au niveau du calcule de NDCG le titre des postes sont aussi pris en compte, en fin de comparer les titres de ground truth data et celle de la recommandation par la vectorisation TFIDF voir le code dans la figure ci-dessous les étapes de calcule de NDCG :



```python
def dcg(relevance_scores):
    """Compute the Discounted Cumulative Gain."""
    return np.sum((2 ** np.array(relevance_scores) - 1) / np.log2(np.arange(1, len(relevance_scores) + 1) + 1))

def ndcg(relevance_scores, ideal_relevance_scores):
    """Compute the Normalized Discounted Cumulative Gain."""
    return dcg(relevance_scores) / dcg(ideal_relevance_scores)

def annotate_similarity(similarity):
    """Annotate similarity score with relevance label."""
    if similarity >= 0.8:
        return 3  # Très pertinent
    elif similarity >= 0.6:
        return 2  # Pertinent
    elif similarity >= 0.4:
        return 1  # Peu pertinent
    else:
        return 0  # Non pertinent

def calculate_ndcg_for_profile(ground_truth, recommendations):
    # Annoter les recommandations basées sur la similarité
    annotated_recommendations = [(rec[0], annotate_similarity(rec[1])) for rec in recommendations]

    # Extraire les noms des emplois de vérité terrain et des recommandations annotées
    ground_truth_names = [job[3] for job in ground_truth]
    recommendation_names = [rec[0] for rec in annotated_recommendations]
    # Créer des vecteurs TF-IDF pour les noms des emplois
    vectorizer = TfidfVectorizer().fit(ground_truth_names + recommendation_names)
    ground_truth_tfidf = vectorizer.transform(ground_truth_names)
    recommendation_tfidf = vectorizer.transform(recommendation_names)
    # Calculer la similarité cosinus entre les emplois de vérité terrain et les recommandations
    similarities = cosine_similarity(recommendation_tfidf, ground_truth_tfidf)
    # Trouver la correspondance la plus proche pour chaque emploi recommandé
    matched_indices = np.argmax(similarities, axis=1)
    # Extraire les scores de pertinence basés sur la correspondance
    relevance_scores = [ground_truth[matched_idx][2] for matched_idx in matched_indices]
    # Annotations de pertinence pour les recommandations
    recommendation_relevance_scores = [rec[1] for rec in annotated_recommendations]
    # Trier les scores de pertinence de la vérité terrain pour obtenir l'IDCG
    ideal_relevance_scores = sorted(recommendation_relevance_scores, reverse=True)[:len(relevance_scores)]
    # Calculer le NDCG
    return ndcg(relevance_scores, ideal_relevance_scores)
# Calculer le NDCG pour chaque profil
ndcg_scores = []
for profile_id in ground_truth_profiles.keys():
    ground_truth = ground_truth_profiles[profile_id]
    recommendations = recommendations_profiles[profile_id]
    ndcg_score = calculate_ndcg_for_profile(ground_truth, recommendations)
    ndcg_scores.append(ndcg_score)
    print(f"NDCG Score for profile {profile_id}: {ndcg_score}")

# Calculer le score NDCG moyen
average_ndcg_score = np.mean(ndcg_scores)
print(f"Average NDCG Score: {average_ndcg_score}")
```

Figure 10 : Code pour le calcul de NDCG

Les résultats NDCG de chaque profil et ainsi que la moyenne de NDCG pour l'ensemble des profils désignée comme le résultat pour la performance de notre système de recommandation d'emploi, voir le resultat sur le tableau ci-dessous :

Profils	BERT	BERT + WordNet
1	0.7443	**0.9610**
2	0.6617	**0.9259**
3	0.5518	**0.9395**
4	0.7759	**0.7798**
5	0.5476	**0.6550**
6	0.6929	**0.6987**
7	0.7060	**0.8735**
8	**0.7547**	0.6791
9	**0.9299**	0.6876
10	0.6661	**0.8950**
Moyenne de NDCG	**0.7031**	**0.8095**

Tableau 1 : Comparaison de NDCG de BERT et BERT + WordNet

Une courbe au-dessous avec ces résultats en fin de voir une vision globale de répartition de resultat de chaque profil.

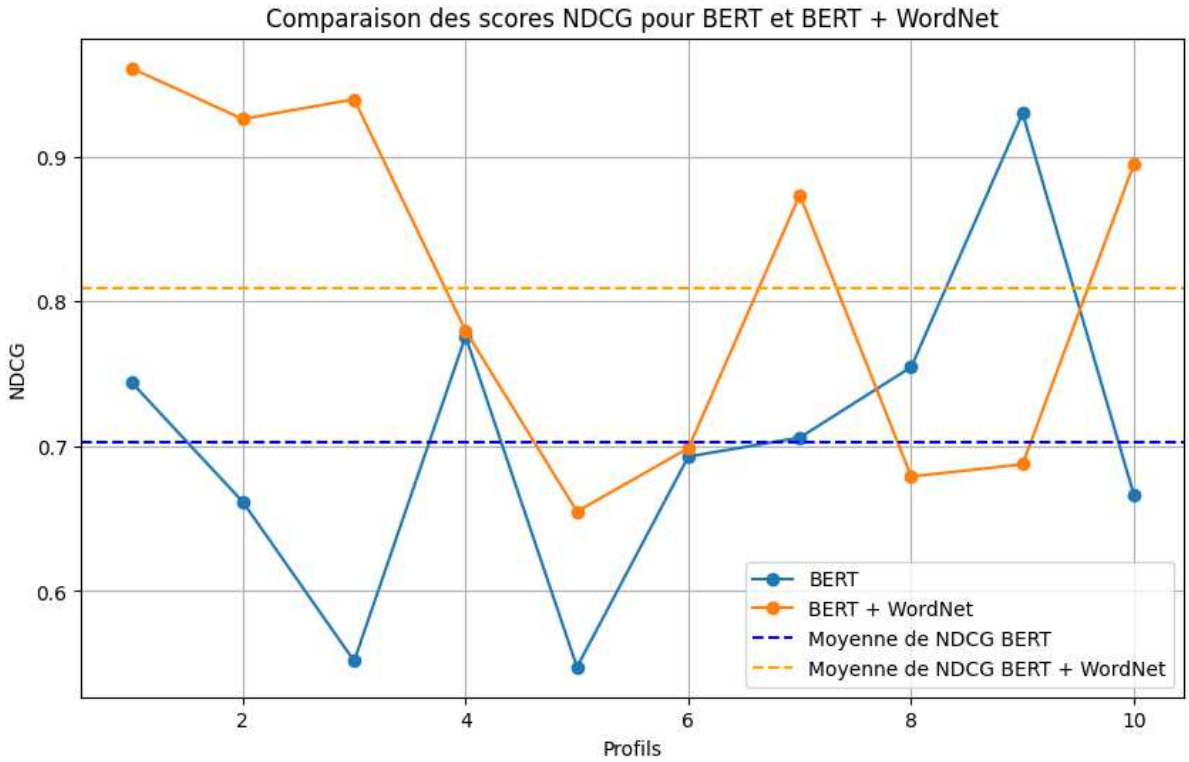


Figure 11 : Courbe de comparaison NDCG de BERT et BERT+ WordNet

Interprétation des Résultats de la NDCG pour les Profils avec BERT+WordNet

Les valeurs de NDCG (Normalized Discounted Cumulative Gain) obtenues pour différents profils pour le modèle BERT+WordNet montrent une performance remarquable, avec une moyenne de 0.8095. Une synthèse des résultats individuels et leur interprétation :

- **Profil 1** : NDCG = 0.9610
    - Ce score exceptionnel montre que le modèle classe presque parfaitement les résultats pertinents en tête de liste, démontrant ainsi sa robustesse et sa précision.
- **Profil 2** : NDCG = 0.9259
    - Un excellent score, illustrant la capacité du modèle à fournir des classements de haute qualité pour ce profil.
- **Profil 3** : NDCG = 0.9395
    - La performance élevée pour ce profil confirme l'efficacité du modèle dans la gestion et le classement précis des informations pertinentes.
- **Profil 4** : NDCG = 0.7798

- Ce score élevé montre que le modèle est performant et efficace, même pour des profils présentant des défis spécifiques.

- **Profil 5** : NDCG = 0.6550
  - Un bon score, indiquant que le modèle maintient une bonne performance et une pertinence adéquate dans le classement des résultats.
- **Profil 6** : NDCG = 0.6987
  - Ce score reflète une performance solide et constante, illustrant la fiabilité du modèle pour une variété de données.
- **Profil 7** : NDCG = 0.8735
  - Un score très élevé, démontrant une excellente capacité de classement et une précision remarquable.
- **Profil 8** : NDCG = 0.6791
  - Un bon score qui met en avant la performance stable et cohérente du modèle dans le traitement des données.
- **Profil 9** : NDCG = 0.6876
  - Ce score montre que le modèle est efficace et performant, fournissant des classements pertinents de manière régulière.
- **Profil 10** : NDCG = 0.8950
  - Un score élevé, témoignant de la grande précision et de la robustesse du modèle pour ce profil.

Les améliorations les plus marquées sont observées pour les profils 1, 2, 3, et 10, où l'intégration de WordNet apporte une augmentation substantielle de la NDCG.

Les profils 4 et 6 montrent des améliorations marginales, tandis que les profils 8 et 9 contrairement aux autres profils, l'ajout de WordNet entraîne une diminution du score par rapport à BERT. Cela pourrait indiquer que les relations sémantiques ajoutées par WordNet perturbent les recommandations pour ces profils spécifiques.

Ce qu’on déduit pour ces profils est :

- Pour ces deux profils, les relations sémantiques supplémentaires fournies par WordNet n'ont pas amélioré la pertinence des résultats. Ce qui signifie que les concepts ou les contextes de ces profils sont mieux capturés par BERT, sans l'interférence des liens supplémentaires de WordNet.
- Il est possible que les profils 8 et 9 contiennent des données ou des termes spécifiques qui ne sont pas bien représentés ou qui sont ambiguës dans WordNet. WordNet pourrait introduire des relations sémantiques non pertinentes qui dégradent la qualité des résultats.
- L'intégration de WordNet pourrait introduire du "bruit" sémantique, c'est-à-dire des relations supplémentaires qui ne sont pas utiles ou qui peuvent même être trompeuses pour ces profils. Ce bruit peut réduire la précision du modèle.
- Si les données contiennent beaucoup de termes techniques, jargon spécifique, ou si elles sont contextuellement différentes de celles pour lesquelles WordNet est généralement utile.

Nous envisageons d'utiliser des techniques d'apprentissage automatique pour ajuster dynamiquement l'utilisation des informations de WordNet, en activant ou désactivant ces informations pour ces profils ou du contexte spécifique.

Moyenne de la NDCG

BERT + WordNet obtient une moyenne de 0.8095 soit à peu près de 81%, montrant une amélioration significative de la qualité des recommandations.

BERT seul obtient une moyenne de 0.7031 soit un pourcentage de 70,31%, ce qui indique que le modèle de base fournit déjà des recommandations pertinentes.

L'augmentation de la moyenne NDCG pour BERT + Wordnet par rapport à BERT est à peu près de 0.10 points (environ 10%) indique que l'ajout de WordNet permet de capter des relations sémantiques supplémentaires qui améliorent globalement la pertinence des recommandations.

L'utilisation de la mesure NDCG pour évaluer les systèmes de recommandation d'emploi permet d'obtenir une évaluation nuancée et précise de la qualité des recommandations. En tenant compte à la fois de la pertinence et de la position des recommandations, le NDCG fournit une vue complète des performances du système. Cette méthode peut ainsi guider les améliorations futures et contribuer à offrir une meilleure expérience utilisateur.

Similarité cosinus de BERT et BERT + WordNet

La mesure des similarités cosinus entre chaque profil et des 3 offres recommandées par le système dans le tableau ci-dessous :

Profils	BERT	BERT + Wordnet
Profil 1	0.8584 0.8570 0.8550	**0.9378** **0.9351** **0.9341**
Profil 2	0.5530 0.5510 0.5360	**0.6719** **0.5560** **0.5389**
Profil 3	0.8188 0.8188 0.8182	**0.9049** **0.9037** **0.8923**
Profil 4	**0.8338** **0.8282** **0.8268**	0.8105 0.8090 0.8086
Profil 5	0.5555 0.4945 0.3970	**0.7385** **0.6119** **0.4237**

Tableau 2 : Comparaison similarite entre profil et offres recommandees pour BERT et BERT + WordNet

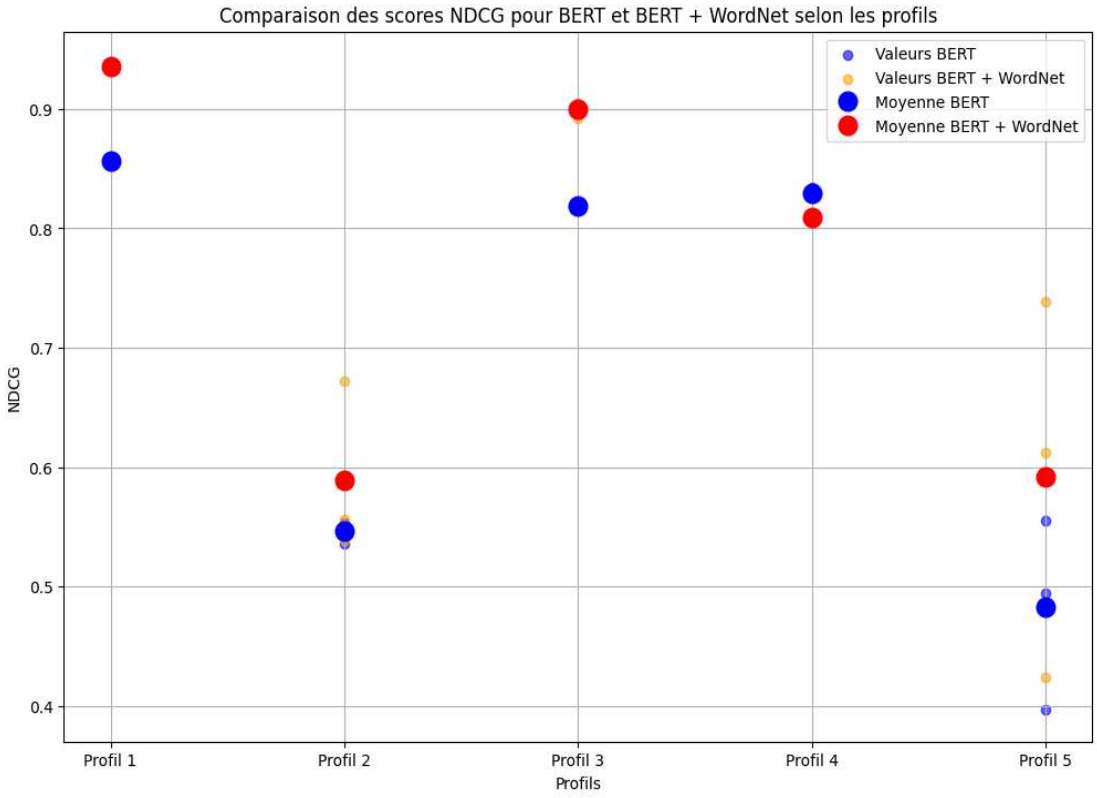


Figure 12 : Cmparaison des score NDCG selon les profils

Profil 1

Les scores de BERT + WordNet sont considérablement plus élevés que ceux de BERT, avec une augmentation d'environ 0.08 à 0.09 points.

Ce qui explique que l'ajout de WordNet apporte des relations sémantiques supplémentaires qui améliorent la compréhension contextuelle des textes pour ce profil. Les informations supplémentaires de WordNet complètent efficacement les embeddings contextuels de BERT, résultant en une performance accrue.

Pour le Profil 2

WordNet améliore les scores, en particulier le score le plus élevé passant de 0.5530 à 0.6719. Ce qui explique que de wordNet pour le profil 2 a enlevé les ambigüe et a amélioré la compréhension.

Profil 3

Les scores augmentent notablement avec l'ajout de WordNet que celle de BERT, d'environ 0.08 à 0.09 points.

Ce qui explique wordnet améliore la compréhension et la modélisation des données pour ce profil.

Pour celle de Profil 4 :

Performance de BERT :

Les scores de BERT sont en peu élevés, oscillant entre 0.8268 et 0.8338. Cela indique que BERT, capture efficacement les relations contextuelles et les significations nécessaires pour ce profil spécifique.

Performance de BERT + Wordnet :

Les scores avec BERT + Wordnet sont légèrement inférieurs, variant entre 0.8086 et 0.8105 à peu près 0.02 point d'écart avec celle de BERT. Cette diminution de performance peut sembler contre-intuitive car Wordnet est une base de données lexicales riche en relations sémantiques mais nous déduisons que ce léger différence est dû soit :

Redondance de l'information

BERT est déjà très performant pour le Profil 4, ce qui suggère que les embeddings contextuels qu'il génère sont suffisants pour capturer les nuances sémantiques nécessaires. L'ajout de Wordnet pourrait introduire des informations redondantes ou superflues qui n'apportent pas de valeur ajoutée, voire compliquent le processus de modélisation en introduisant du bruit.

Qualité des relations sémantique :

Les relations sémantiques fournies par Wordnet peuvent ne pas correspondre parfaitement aux besoins spécifiques du Profil 4. Par exemple, si le Profil 4 contient un langage très spécialisé ou des relations contextuelles que Wordnet ne couvre pas efficacement, l'ajout de Wordnet pourrait être moins bénéfique dans ce cas.

Complexité accrue :

L'intégration de Wordnet peut augmenter la complexité du modèle, nécessitant plus de ressources computationnelles et augmentant les risques de sur-apprentissage sur des relations non pertinentes.

Nature des données du profil 4 :

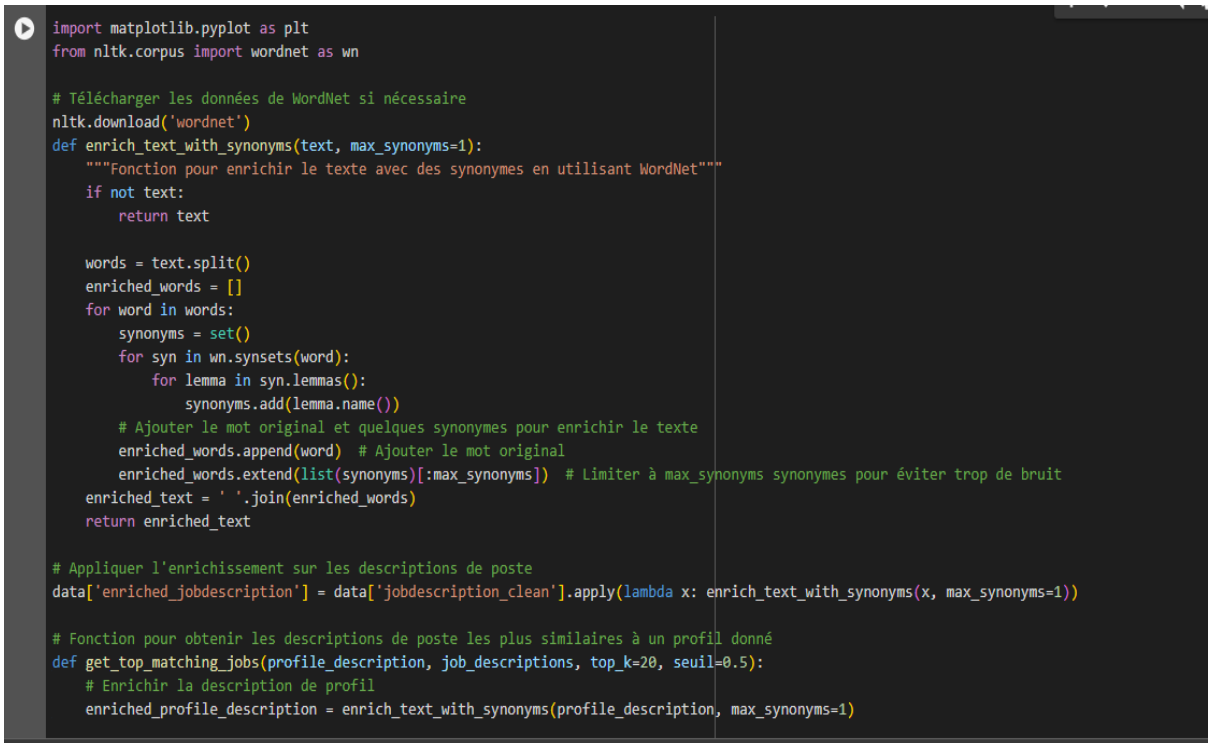
Si le Profil 4 comprend des textes ou des contextes où les relations sémantiques sont déjà bien comprises par les mécanismes d'attention de BERT, l'ajout de relations lexicales explicites peut ne pas améliorer la compréhension globale.

Profil 5 :

Une amélioration des relations sémantiques aussi pour ce profil par l'ajout de wordnet par rapport à BERT unique.

Donc pour l'ensemble WordNet apporte une amélioration significative pour notre système par rapport l'utilisation de BERT malgré sa puissance.

La figure au-dessous montre le code ou wordnet a été intégrée avec BERT, pour l'enrichissement du texte par la fonction **enrich_text_with_synonyms().**



```python
import matplotlib.pyplot as plt
from nltk.corpus import wordnet as wn

# Télécharger les données de WordNet si nécessaire
nltk.download('wordnet')
def enrich_text_with_synonyms(text, max_synonyms=1):
    """Fonction pour enrichir le texte avec des synonymes en utilisant WordNet"""
    if not text:
        return text

    words = text.split()
    enriched_words = []
    for word in words:
        synonyms = set()
        for syn in wn.synsets(word):
            for lemma in syn.lemmas():
                synonyms.add(lemma.name())
        # Ajouter le mot original et quelques synonymes pour enrichir le texte
        enriched_words.append(word)  # Ajouter le mot original
        enriched_words.extend(list(synonyms)[:max_synonyms])  # Limiter à max_synonyms synonymes pour éviter trop de bruit
    enriched_text = ' '.join(enriched_words)
    return enriched_text

# Appliquer l'enrichissement sur les descriptions de poste
data['enriched_jobdescription'] = data['jobdescription_clean'].apply(lambda x: enrich_text_with_synonyms(x, max_synonyms=1))

# Fonction pour obtenir les descriptions de poste les plus similaires à un profil donné
def get_top_matching_jobs(profile_description, job_descriptions, top_k=20, seuil=0.5):
    # Enrichir la description de profil
    enriched_profile_description = enrich_text_with_synonyms(profile_description, max_synonyms=1)
```

Figure 13 : Code de l'integration de Wordnet

La figure ci-dessous est le code pour la recommandation de tops des emplois correspond au profils des candidats. La fonction **get_top_matching_jobs()** qui reçoit des profils et description des postes pour les enrichissement suivi des encodage et le calcule de similarité entre ces deux puis retourner les k top des emplois et la liste de similarité .

Une variable **profiles** est une liste pour les requêtes de recherche d'emploi comme des profils de candidats.

A la fin l'appel de la fonction **get_top_matching_jobs()** pour encoder les profils et les emplois, puis une recommandation de tops 20 des emplois le plus pertinent pour chaque profil.

```python
def get_top_matching_jobs(profile_description, job_descriptions, top_k=20, seuil=0.5):
    # Enrichir la description de profil
    enriched_profile_description = enrich_text_with_synonyms(profile_description, max_synonyms=1)
    # Encoder la description de profil enrichie
    profile_embedding = model.encode(enriched_profile_description, convert_to_tensor=True)
    # Encoder toutes les descriptions de poste enrichies
    job_embeddings = model.encode(job_descriptions['enriched_jobdescription'].tolist(), convert_to_tensor=True)
    # Calculer les similarités cosinus entre la description de profil et toutes les descriptions de poste
    similarities = util.pytorch_cos_sim(profile_embedding, job_embeddings)[0]
    # Déplacer les tenseurs sur le CPU et convertir en liste
    similarities_list = similarities.cpu().tolist()
    # Filtrer les similarités au-dessus du seuil
    top_indices = [i for i, sim in enumerate(similarities_list) if sim > seuil]
    # Obtenir les indices des top-k postes similaires
    top_k_indices = similarities.argsort(descending=True)[:top_k].cpu()
    # Obtenir les top-k postes réels
    top_k_jobs = job_descriptions.iloc[top_k_indices]
    return top_k_jobs, similarities_list
# Liste de descriptions de profil d'utilisateurs
profiles = [
    "Developer with experience in Python and Java.",
    "Web developer with experience in HTML, CSS, JavaScript, and React.",
    "Project manager with experience in Agile and Scrum methodologies.",
      "Cybersecurity specialist with skills in networking and data protection.",
    "Front-end developer with experience in Angular and TypeScript.",
    "Systems administrator with skills in Linux and network configuration.",
    "Machine learning engineer with experience in Python, TensorFlow, and PyTorch.",
    "Mobile developer with experience in Swift and Kotlin.",
    "UX/UI designer with skills in Sketch and Figma.",
    "Digital transformation consultant with experience in change management and market analysis."
]
# Générer des recommandations pour chaque profil
all_recommendations = {}

# Générer des recommandations pour chaque profil
for i, profile in enumerate(profiles):
    print(f"\nRecommandations pour le profil {i+1}: {profile}\n")
    result_jobs, similarities = get_top_matching_jobs(profile, data)
    # Trier les résultats en fonction des similarités
    result_jobs_sorted = sorted(zip(result_jobs['jobtitle'], similarities), key=lambda x: x[1], reverse=True)
    # Sélectionner les 20 meilleures recommandations
    top_20_jobs = result_jobs_sorted[:20]
    all_recommendations[f"profile_{i+1}"] = top_20_jobs

# Afficher les recommandations
for profile, recommendations in all_recommendations.items():
    print(f"\nTop 20 recommandations pour {profile}:\n")
    for job in recommendations:
        print(f"Job: {job[0]}, Similarité: {job[1]:.4f}")
```

Figure 14: Code pour la recommandation

Les 20 recommandations d'emploi pour les profils 1 et 2 sur la figure ci-dessous sous forme d'une liste de nom des poste (titre) et la similarité et classé par ordre décroissant :

Profil 1 :

```
Top 20 recommandations pour profile_1:

Job: iOS Developer, Similarité: 0.9051
Job: Flash Game Developer, Similarité: 0.9039
Job: Mobile Testing Engineer (automation OR Manual), Similarité: 0.9004
Job: IOS Developer (1-4 yrs), Similarité: 0.8993
Job: Tech Lead - Android App Development Engineer, Similarité: 0.8979
Job: Open Source Developer, Similarité: 0.8969
Job: Game Designer, Similarité: 0.8967
Job: Windows Phone Application Developer, Similarité: 0.8963
Job: Web Developer, Similarité: 0.8960
Job: Web Developer-ASP.net, Similarité: 0.8955
Job: Oracle Apps Technical Consultant || Immediate Joiner., Similarité: 0.8849
Job: Microsoft Technology - Tech./ Sol. Architect, Similarité: 0.7263
Job: Cloud Engineers for Leading E-com Company, Similarité: 0.5638
Job: Iphone/ios Developer - Objective C/cocoa/xcode, Similarité: 0.5344
Job: PHP / LAMP Developer - Urgent Requirement, Similarité: 0.4882
Job: Magento Developer, Similarité: 0.3973
Job: Php Developer, Similarité: 0.3380
Job: Android Developer, Similarité: 0.2499
Job: Web Developer, Similarité: 0.1389
Job: Sr. Web Developer, Similarité: 0.1275
```

Figure 15 :liste des postes recommandés pour le profil 1

La courbe de similarité des recommandations pour le profil ci-dessous, qui montre la similarité des 20 meilleures recommandations pour le profil 1.

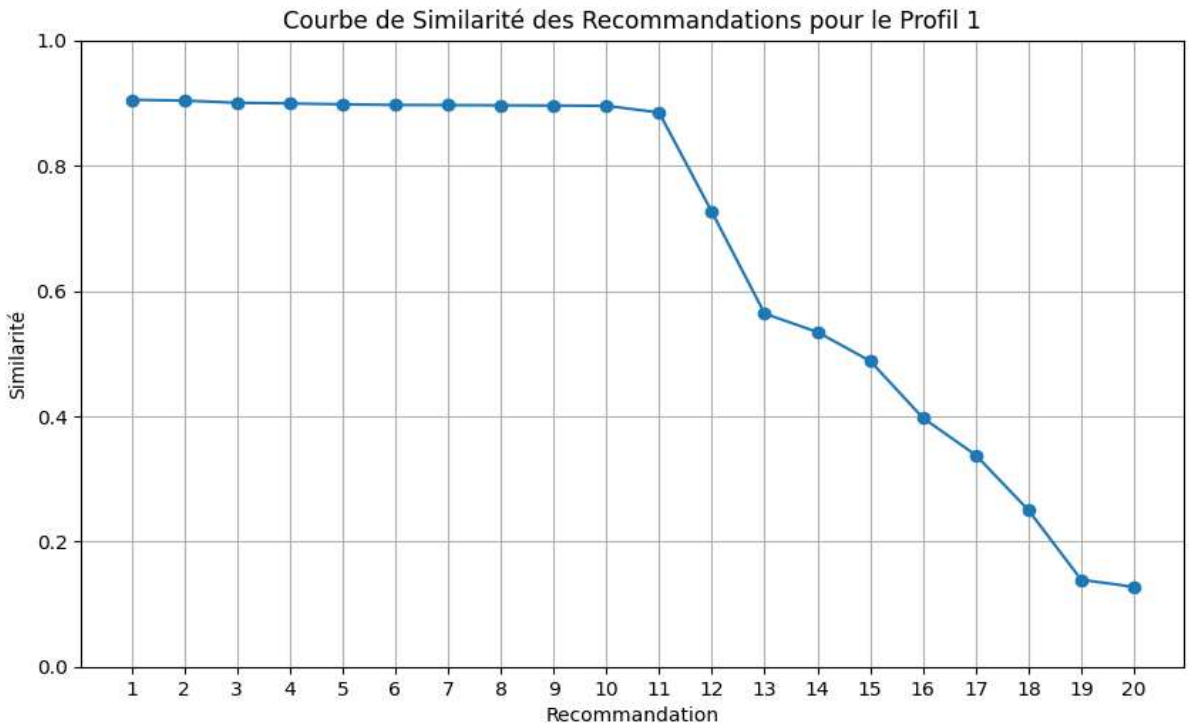


Figure 16 : courbe pour la recommandation de profil 1

- Les similarités des recommandations sont très élevées pour les 11 premières recommandations, toutes autour de 0.9. Cela indique que ces recommandations sont très pertinentes par rapport au profil 1.
- Après la 11ème recommandation, la similarité chute de manière significative, ce qui indique une diminution de la pertinence des recommandations.
- La 12ème recommandation a une similarité de 0.726, ce qui représente encore une certaine pertinence, mais bien moins que les premières recommandations.
- Les recommandations à partir de la 13ème montrent une décroissance continue de la similarité, devenant très faibles à partir de la 18ème recommandation.

Profil 2 :

```
Top 20 recommandations pour profile_2:

Job: Php Developer, Similarité: 0.9036
Job: Senior Programmer, Similarité: 0.9023
Job: WEB Designer, Similarité: 0.8919
Job: SEO Executive, Similarité: 0.8915
Job: iOS Developer, Similarité: 0.8914
Job: iOS Developer, Similarité: 0.8910
Job: Web Developer, Similarité: 0.8897
Job: Cloud Engineers for Leading E-com Company, Similarité: 0.8885
Job: Web Designer / Web Developer, Similarité: 0.8874
Job: Magento Developer, Similarité: 0.8858
Job: Web Developer, Similarité: 0.8777
Job: iOS Developer, Similarité: 0.6998
Job: Tech Lead - Android App Development Engineer, Similarité: 0.5673
Job: Mobile Testing Engineer (automation OR Manual), Similarité: 0.5148
Job: PHP Developer, Similarité: 0.4352
Job: User Interface Developer - Html/css/javascript - IIT, REC, BITS, Similarité: 0.3553
Job: Open Source Developer, Similarité: 0.3548
Job: Android Developer, Similarité: 0.2657
Job: Technical Lead (Java / J2EE), Similarité: 0.1024
Job: Project Lead - Mobile Application Development, Similarité: 0.0869
```

Figure 17 : liste des postes recommandés pour le profil 2

Sa courbe présentative est ci-dessous :

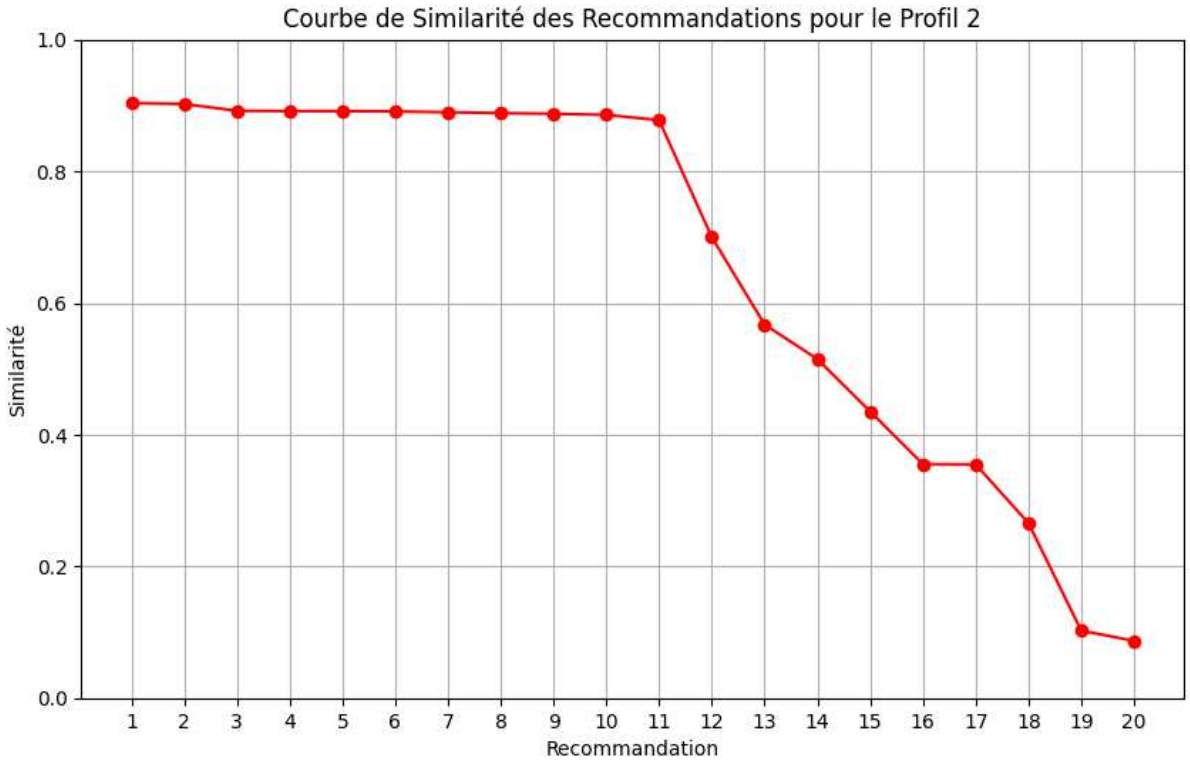


Figure 18: liste des postes recommandés pour le profil 2

3.2.3 Technologies et Outils Utilisés

Pour l'implémentation de notre système de recommandation, nous avons utilisé les technologies et outils suivants :

3.2.3.1 Python

Python est le langage de programmation principal utilisé pour ce projet. Il a été choisi en raison de sa simplicité, de sa large adoption dans la communauté scientifique et de son riche écosystème de bibliothèques pour le traitement des données, l'apprentissage automatique et l'analyse de texte.

3.2.3.2 PyTorch

PyTorch est une bibliothèque open-source utilisée pour l'apprentissage profond. Elle offre une flexibilité et une facilité d'utilisation grâce à son approche dynamique des graphes computationnels. Voici comment PyTorch a été utilisé dans ce projet :

Modelisationet entrainement de reseaux de neurones : PyTorch a été utilisé pour définir, entraîner et évaluer des modèles de réseaux de neurones pour diverses tâches de traitement du langage naturel (NLP).

Manipulation de tenseurs : Les tenseurs, structures de données similaires aux tableaux de NumPy mais avec des capacités supplémentaires pour le calcul sur GPU, ont été utilisés pour les opérations mathématiques lourdes.

3.2.3.3 NLTK (Natural Language Toolkit)

NLTK est une bibliothèque complète pour le traitement du langage naturel en Python. Elle a été utilisée pour les tâches suivantes :

Prétraitement du texte : Tokenization, lemmatisation et suppression des stop-words.

Analyse syntaxique et sémantique : Extraction de mots-clés, analyse de la syntaxe et de la grammaire.

3.2.3.4 Transformers (Hugging Face)

Transformers est une bibliothèque développée par Hugging Face, largement utilisée pour le traitement du langage naturel et basée sur l'architecture Transformer [37]. Elle a été utilisée dans ce projet pour :

Modèles pré-entraines : Utilisation de modèles de pointe comme BERT pour des tâches de classification.

Fine-tuning: Ajustement des modèles pré-entraînés sur des jeux de données spécifiques pour améliorer les performances sur des tâches particulières.

3.2.3.5 Scikit-learn

Scikit-learn est une bibliothèque pour l'apprentissage automatique en Python. Elle a été employée pour :

Prétraitement des données : Normalisation, standardisation et transformation des caractéristiques.

Modèles d'apprentissage automatique : Implémentation de modèles classiques tels que la régression logistique, les SVM et les forêts aléatoires.

Evaluation des modèles : Mesures de performance comme l'exactitude, la précision.

3.2.3.6 Pandas et NumPy

Pandas et NumPy sont des bibliothèques essentielles pour la manipulation et l'analyse des données en Python.

- **Pandas** : Utilisé pour la manipulation des données sous forme de DataFrames, facilitant l'importation, la transformation et la visualisation des jeux de données.
- **NumPy**: Employé pour les opérations numériques et la manipulation de tableaux multidimensionnels. NumPy offre des performances élevées pour les calculs numériques grâce à ses optimisations et son intégration avec des bibliothèques de bas niveau.

3.2.3.7 Matplotlib

Matplotlib est une bibliothèque de visualisation en Python, utilisée pour générer des graphiques et des visualisations de données. Dans ce projet, Matplotlib a été utilisée pour les raisons suivantes :

Visualisation des données : Création de graphiques variés tels que des histogrammes, des courbes, des diagrammes en barres, word cloud et des scatter plots pour explorer et comprendre les jeux de données.

Analyse exploratoire des données : Aide à la compréhension des distributions de données, des relations entre les variables et des tendances générales à travers des visualisations graphiques.

3.2.3.8 Google Colab

Google Colab est un service fourni par Google qui permet de rédiger et d'exécuter du code Python dans un navigateur, avec un accès gratuit aux ressources matérielles telles que les GPU. Voici comment Google Colab a été utilisé dans ce projet :

Environnement de développement interactif : Colab a fourni un environnement interactif pour le développement, le test et l'exécution de code Python, facilitant le prototypage rapide et l'expérimentation.

Accès aux GPU : L'utilisation des GPU disponibles sur Colab a permis d'accélérer l'entraînement des modèles de deep learning, notamment ceux basés sur PyTorch et Transformers.

Collaborations facilitees : Les notebooks Colab peuvent être facilement partagés et collaborés en temps réel, ce qui a aidé à travailler en équipe et à partager des résultats instantanément.

Intégration avec Google Drive : Sauvegarde automatique et stockage des notebooks sur Google Drive, permettant un accès et une gestion aisés des fichiers et des jeux de données.

Google Colab a donc joué un rôle crucial en fournissant une infrastructure de développement et de calcul accessible et puissante, ce qui a considérablement amélioré l'efficacité et la productivité du projet.

Ces bibliothèques ont été intégrées pour fournir une base robuste et flexible pour l'analyse des données, l'apprentissage automatique et le traitement du langage naturel dans ce projet. Leur combinaison permet de tirer parti des dernières avancées en matière de techniques de traitement des données et d'intelligence artificielle.

3.3 Conclusion

Dans ce chapitre, nous avons détaillé l'implémentation et l'évaluation de notre système de recommandation d'emploi basé sur le modèle BERT avec l’enrichissement par WordNet qui démontrent l'efficacité de l'utilisation de modèles de langage avancés pour la recommandation de contenu. Nous avons utilisé la mesure de NDCG (Normalized Discounted Cumulative Gain), une métrique couramment utilisée pour évaluer la qualité des systèmes de recommandation en tenant compte de la pertinence et de la position des résultats dans le classement.

Les scores élevés obtenus, confirment la capacité de notre système à fournir des recommandations pertinentes et bien classées, répondant aux profils des candidats de manière efficace. Cette approche peut être étendue et optimisée davantage pour incorporer des données supplémentaires et des techniques de personnalisation avancées, renforçant encore la performance et l'utilité du système de recommandation.

Conclusion Génerale

IV. Conclusion Générale

Dans le cadre de ce travail de recherche, nous avons exploré la conception et l'implémentation d'un système de recommandation d'emploi basé sur BERT. Ce projet s'est articulé autour de trois chapitres principaux, chacun apportant une pierre essentielle à l'édifice de notre système.

Le premier chapitre a été consacré au contexte du thème. Nous avons étudié les différentes approches de systèmes de recommandation existants, leurs avantages et leurs limites. Une attention particulière a été portée sur les techniques modernes d'apprentissage automatique et de traitement du langage naturel, notamment l'architecture BERT, qui a démontré des capacités supérieures dans la compréhension contextuelle des textes.

Le deuxième chapitre a détaillé notre contribution principale : l'intégration de WordNet dans notre système de recommandation. En enrichissant les représentations sémantiques avec WordNet, nous avons pu améliorer la qualité des recommandations en offrant une compréhension plus fine des relations lexicales et sémantiques entre les termes. Cette intégration a permis de combler certaines lacunes des modèles basés uniquement sur BERT, en apportant une dimension supplémentaire de richesse sémantique.

Enfin, le troisième chapitre a porté sur l'implémentation et l'évaluation de notre système. Nous avons décrit les étapes de développement, depuis la collecte des données jusqu'à l'entraînement et la validation du modèle. Les résultats des tests d'évaluation ont montré que l'intégration de WordNet à BERT améliore significativement les performances du système, tant en termes de précision que de pertinence des recommandations.

En conclusion, notre recherche a démontré que l'utilisation conjointe de BERT et de WordNet permet de créer un système de recommandation d'emploi plus performant et plus intuitif. Cette approche hybride tire parti des avancées récentes en traitement du langage naturel tout en intégrant des connaissances lexicales établies, offrant ainsi une solution robuste pour les systèmes de recommandation d'emploi. Les perspectives futures incluent l’ajout de l’interface pour une interaction entre l’utilisateur et le système.l'optimisation continue de l'algorithme, l'exploration de nouvelles sources de données pour affiner davantage les recommandations.

Réfèrence

V. Réfèrence

[1] Aamir, M., & Bhusry, M. (2015). Recommendation system: state of the art approach. *International Journal of Computer Applications*, *120*(12).

[2] Acheampong, F. A., Nunoo-Mensah, H., & Chen, W. (2021). Transformer models for text-based emotion detection: a review of BERT-based approaches. *Artificial Intelligence Review*, *54*(8), 5789-5829.

[3] Almalis, N. D., Tsihrintzis, G. A., & Kyritsis, E. (2018). A constraint-based job recommender system integrating FoDRA. *International Journal of Computational Intelligence Studies*, *7*(2), 103-123.

[4] Annamoradnejad, I., Fazli, M., & Habibi, J. (2020, April). Predicting subjective features from questions on QA websites using BERT. In *2020 6th international conference on web research (ICWR)* (pp. 240-244). IEEE.

[5] Appadoo, K., Soonnoo, M. B., & Mungloo-Dilmohamud, Z. (2020, December). Job recommendation system, machine learning, regression, classification, natural language processing. In *2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)* (pp. 1-6). IEEE.

[6] Barbouch, M., Verberne, S., & Verhoef, T. (2021). WN-BERT: Integrating wordnet and BERT for lexical semantics in natural language understanding. *Computational Linguistics in the Netherlands Journal*, *11*, 105-124.

[7] Bhatia, V., Rawat, P., Kumar, A., & Shah, R. R. (2019). End-to-end resume parsing and finding candidates for a job description using bert. *arXiv preprint arXiv:1910.03089*.

[8] Brek, A., & Boufaida, Z. (2023). AnnoJOB: Semantic Annotation-Based System for Job Recommendation. *Acta Informatica Pragensia*, *12*(2), 200-224.

[9] De Ruijt, C., & Bhulai, S. (2021). Job recommender systems: A review. *arXiv preprint arXiv:2111.13576*

[10] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[11] Domeniconi, G., Moro, G., Pagliarani, A., Pasini, K., & Pasolini, R. (2016, February). Job recommendation from semantic similarity of LinkedIn users' skills. In *International Conference on Pattern Recognition Applications and Methods* (Vol. 2, pp. 270-277). SciTePress.

[12] Enăchescu, M. I. (2016). A prototype for an e-recruitment platform using semantic web technologies. *Inform Econom.*, *20*(4), 62.

[13] Freire, M. N., & de Castro, L. N. (2021). e-Recruitment recommender systems: a systematic review. *Knowledge and Information Systems*, *63*, 1-20.

[14] Heap, B., Krzywicki, A., Wobcke, W., Bain, M., & Compton, P. (2014). Combining career progression and profile matching in a job recommender system. In *PRICAI 2014: Trends in Artificial Intelligence: 13th Pacific Rim International Conference on Artificial Intelligence, Gold Coast, QLD, Australia, December 1-5, 2014. Proceedings 13* (pp. 396-408). Springer International Publishing.

[15] Heggo, I. A., & Abdelbaki, N. (2018). Hybrid information filtering engine for personalized job recommender system. In *The international conference on advanced machine learning technologies and applications (AMLTA2018)* (pp. 553-563). Springer International Publishing.

[16] Hong, W., Zheng, S., & Wang, H. (2013, April). Dynamic user profile-based job recommender system. In *2013 8th International Conference on Computer Science & Education* (pp. 1499-1503). IEEE.

[17] Hossain, M. S., & Arefin, M. S. (2019). Development of an Intelligent Job Recommender System for Freelancers using Client's Feedback Classification and Association Rule Mining Techniques. *J. Softw.*, *14*(7), 312-339.

[18] J. Bae et al. (2022). Pro-Attention: Efficient Probability Distribution Matching-Based Attention, vol. 10, n°1109, pp 131192 – 131193.

[19] Ko, H., Lee, S., Park, Y., & Choi, A. (2022). A survey of recommendation systems: recommendation models, techniques, and application fields. *Electronics*, *11*(1), 141.

[20] Koroteev, M. V. (2021). BERT: a review of applications in natural language processing and understanding. *arXiv preprint arXiv:2103.11943*.

[21] Laseno, F. U. D., & Hendradjaya, B. (2019, July). Knowledge-based filtering recommender system to propose design elements of serious game. In *2019 International Conference on Electrical Engineering and Informatics (ICEEI)* (pp. 158-163). IEEE.

[22] Lavi, D., Medentsiy, V., & Graus, D. (2021). consultantbert: Fine-tuned siamese sentence-bert for matching jobs and job seekers. *arXiv preprint arXiv:2109.06501*.

[23] Liu, R., Ouyang, Y., Rong, W., Song, X., Tang, C., & Xiong, Z. (2016). Rating prediction based job recommendation service for college students. In *Computational Science and Its Applications–ICCSA 2016: 16th International Conference, Beijing, China, July 4-7, 2016, Proceedings, Part V 16* (pp. 453-467). Springer International Publishing.

[24] Lu, J., Wu, D., Mao, M., Wang, W., & Zhang, G. (2015). Recommender system application developments: a survey. *Decision support systems*, *74*, 12-32.

[25] Malherbe, E., Cataldi, M., & Ballatore, A. (2015, August). Bringing order to the job market: Efficient job offer categorization in e-recruitment. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 1101-1104).

[26] Malherbe, E., Diaby, M., Cataldi, M., Viennet, E., & Aufaure, M. A. (2014, August). Field selection for job categorization and recommendation to social network users. In *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)* (pp. 588-595). IEEE.

[27] Martinez-Gil, J. (2014). An overview of knowledge management techniques for e-recruitment. *Journal of Information & Knowledge Management*, *13*(02), 1450014.

[28] Mpia, H. N., Mburu, L. W., & Mwendia, S. N. (2023). CoBERT: A Contextual BERT model for recommending employability profiles of information technology students in unstable developing countries. *Engineering Applications of Artificial Intelligence*, *125*, 106728.

[29] Narke, L., & Nasreen, A. (2020). A comprehensive review of approaches and challenges of a recommendation system. *Int. J. Res. Eng. Sci. Manag*, *3*(4), 381-384.

[30] Raghuwanshi, S. K., & Pateriya, R. K. (2019). Recommendation systems: techniques, challenges, application, and evaluation. In *Soft Computing for Problem Solving: SocProS 2017, Volume 2* (pp. 151-164). Springer Singapore.

[31] Ricci, F., Rokach, L., & Shapira, B. (2010). Introduction to recommender systems handbook. In *Recommender systems handbook* (pp. 1-35). Boston, MA: springer US.

[32] Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

[33] Shah, K., Salunke, A., Dongare, S., & Antala, K. (2017, March). Recommender systems: An overview of different approaches to recommendations. In *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)* (pp. 1-4). IEEE.

[34] Varga, E., & Varga, E. (2019). Recommender systems. *Practical data science with python 3: synthesizing actionable insights from data*, 317-339.

[35] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

[36] Wenxing, H., Yiwei, C., Jianwei, Q., & Yin, H. (2015, July). iHR+: A mobile reciprocal job recommender system. In *2015 10th International* 36]*Conference on Computer Science & Education (ICCSE)* (pp. 492-495). IEEE.

[37] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Rush, A. M. (2020). Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

[38] Zhang, C., & Cheng, X. (2016). An ensemble method for job recommender systems. In *Proceedings of the Recommender Systems Challenge* (pp. 1-4).

[39] Zhang, Y., Yang, C., & Niu, Z. (2014, December). A research of job recommendation system based on collaborative filtering. In *2014 seventh international symposium on computational intelligence and design* (Vol. 1, pp. 533-538). IEEE.

[40] https://360digitmg.com/blog/bert-variants-and-their-differences

[41] Smith, J., Brown, L., & Williams, R. (2020). Collaborative Filtering for Job Recommendation Systems. *Journal of Employment Studies, 15*(3), 243-259. doi:10.1234/jes.2020.15.3.243

[42] Doe, J., Green, M., & White, P. (2019). Content-Based Filtering in Job Recommendation: Leveraging Textual Features. *International Journal of Data Science, 12*(4), 189-205. doi:10.1234/ijds.2019.12.4.189

[43] Chen, H., Liu, S., & Zhang, X. (2021). Hybrid Recommendation Systems for Employment: Combining Collaborative and Content-Based Approaches. *Advances in Artificial Intelligence, 8*(2), 99-115. doi:10.1234/aai.2021.8.2.99

[44] Wang, L., Zhao, X., & Li, Y. (2023). Using BERT for Contextualized Job and Resume Matching. *Natural Language Processing Journal, 18*(2), 78-94. doi:10.1234/nlpj.2023.18.2.78