

RAPPORT DE STAGE

Présenté à

**L'École Nationale d'Électronique et des
Télécommunications de Sfax**

Dans le cadre de la formation en

**Diplôme National d'Ingénieur en :
Ingénierie des Données et Systèmes Décisionnels**

Élaboré par : Rami AYOUNI

**Job Match : Outil intelligent d'aide au
recrutement basé sur l'LLM**

**Encadrant professionnel :
Mme AMMAR Sonda**

Projet réalisé au





REMERCIEMENT

*À la fin de cette expérience de stage enrichissante, je tiens à exprimer ma sincère gratitude à toutes les personnes qui ont joué un rôle important dans le succès de ce projet et dans mon expérience au sein du **CRNS**.*

Je remercie les membres du corps enseignant de l'École Nationale d'Électronique et des Télécommunications de Sfax ainsi que les professionnels impliqués pour leur encadrement théorique tout au long de ma formation.

*Je tiens également à exprimer ma profonde gratitude à mon encadrante professionnelle, Mme **Sonda AMMAR**, pour son accompagnement et sa supervision tout au long de mon stage. Son expertise et ses conseils ont été essentiels à la réussite de ce projet.*

*Enfin, Je remercie chaleureusement mon binôme de stage, **Atta Allah HILALI**, pour sa collaboration exemplaire, ses conseils éclairés et son soutien constant, qui ont largement contribué à la réussite de ce stage.*

Rami AYOUNI



TABLE DES MATIÈRES

LISTE DES FIGURES	iv
INTRODUCTION GÉNÉRALE	1
1 INTRODUCTION	2
1.1 Présentation de l'organisme d'accueil.	3
1.2 Contexte	4
1.3 Objectif	5
2 État de l'art et Méthodologie	6
2.1 Les Grands Modèles de Langage (LLMs)	7
2.2 Architecture des LLMs	7
2.2.1 Transformers	7
2.2.2 Fonctionnement des LLMs	9
2.3 Automatisation du Pré-sélection des candidats	9
2.3.1 Architecture Globale du Système	9
2.3.2 Pré-sélection avec LLMs.	10
2.3.3 Importance des LLMs pour la pré-sélection	11

3	Réalisation du Projet et Résultats	13
3.1	Missions Confiées	12
3.2	Architecture globale du système	13
3.3	Implémentation	14
3.3.1	Langages et Bibliothèques	14
3.3.2	Outils de Développement	15
3.3.3	Pré-traitement des Données (CV) et Prompt Engineering	16
3.3.4	Intégration et Fine-Tuning du Modèle (MistralAI-7B).	16
3.3.5	Le développement de l'interface.	16
3.3.6	Difficultés rencontrées	17

CONCLUSION GÉNÉRALE	19
----------------------------	-----------



LISTE DES FIGURES

1.1	Logo du Centre de Recherche Numérique de Sfax (CRNS).	3
2.1	Schéma représentant l'architecture générale d'un transformeur.. . . .	7
2.2	Logo de fonctionnement des LLMs.	9
2.3	Schéma de la structure générale du Projet.	10
3.3.2	Logo de VS Code	16



INTRODUCTION GÉNÉRALE

Dans un contexte de transformation numérique mondiale, le secteur des technologies de l'information connaît une croissance soutenue, entraînant une forte demande en profils techniques. En parallèle, le nombre de diplômés issus des filières informatiques ne cesse d'augmenter, ce qui contribue à intensifier la concurrence sur le marché de l'emploi. Cette situation génère un afflux important de candidatures pour chaque offre, rendant le processus de recrutement particulièrement complexe et chronophage.

Les professionnels des ressources humaines se retrouvent confrontés à la nécessité de traiter manuellement un volume considérable de CVs, souvent rédigés de manière hétérogène, avec des compétences exprimées différemment selon les candidats. Cela rend l'évaluation comparative difficile, augmente le risque d'erreurs ou de biais dans la présélection, et peut nuire à la qualité du recrutement.

Dans ce contexte, l'automatisation intelligente du traitement des candidatures apparaît comme une solution pertinente pour répondre à ces défis. Le projet mené dans le cadre de ce stage s'inscrit dans cette logique. Il consiste à développer un système complet permettant d'analyser automatiquement les CVs, d'extraire les informations essentielles, de comparer les profils aux offres d'emploi correspondantes, et de classer les candidats en fonction de leur adéquation.

Ce rapport est structuré en trois parties : l'état de l'art sur les LLMS, la Méthodologie adoptée pour la présélection des candidats au sein du département RH. Et enfin, les résultats obtenus, accompagnés des perspectives d'amélioration pour l'avenir.

1

Chapitre

INTRODUCTION

Sommaire

1.1	Présentation de l'organisme d'accueil.	3
1.2	Contexte	4
1.3	Objectif	5

1.1 Présentation de l'organisme d'accueil.

Le **Centre de Recherche Numérique de Sfax (CRNS)**, fondé en 2002, est un établissement de pointe dédié à la recherche avancée dans le domaine des technologies de l'information et de la communication (TIC). Depuis sa création, le CRNS s'est affirmé comme un acteur majeur dans le développement de solutions numériques innovantes. L'institution joue un rôle clé dans la promotion de l'innovation et le transfert de technologie, visant à répondre aux besoins technologiques croissants de divers secteurs.

Le CRNS se spécialise dans la recherche en technologies de l'information, avec un accent particulier sur l'intelligence artificielle, le traitement des images et des données, ainsi que les systèmes embarqués. L'institution développe des technologies avancées qui contribuent à l'évolution des TIC et à l'amélioration des processus dans ces domaines.

L'organisation interne du CRNS est structurée autour de plusieurs laboratoires de recherche, chacun dirigé par un chercheur principal. Ces laboratoires sont spécialisés dans divers domaines technologiques et collaborent régulièrement sur des projets interdisciplinaires. Cette organisation permet une synergie efficace entre les différentes équipes de recherche, favorisant l'innovation et l'échange de connaissances.

Le CRNS se distingue comme un leader en recherche appliquée dans le domaine des technologies numériques. L'institution est reconnue pour ses nombreuses collaborations internationales et ses partenariats avec des entreprises de haute technologie. Son objectif est de créer un écosystème axé sur l'innovation, exploitant les résultats de la recherche pour maintenir une position de pointe dans le secteur des TIC.



FIGURE 1.1 – Logo du Centre de Recherche Numérique de Sfax (CRNS).

1.2 Contexte

Le rapprochement entre l'offre et la demande d'emploi constitue un enjeu économique et social majeur, en particulier dans des marchés fragmentés où les intitulés de poste, les exigences implicites et les niveaux de séniorité varient fortement d'un secteur à l'autre. Dans ce contexte, les méthodes traditionnelles d'appariement par mots-clés peinent à saisir la richesse des profils et la spécificité des besoins entreprises, surtout lorsque les CV sont hétérogènes (formats PDF/DOCX, structures libres, langues multiples) et que les annonces ne suivent pas de standard commun.

L'essor récent de l'IA appliquée au recrutement embeddings sémantiques, recherche vectorielle, modèles de langage et RAG ouvre des perspectives nouvelles pour comprendre finement les compétences, relier des expériences à des missions et expliquer les recommandations. Toutefois, la performance de ces approches dépend fortement de la qualité, de la diversité et de la réalisme des données: offres publiées sur des pages dynamiques, contenus partiels ou bruités, informations dispersées (compétences, localisation, langues, contraintes légales) et CV très variés.

Les corpus existants sont souvent orientés vers des extraits propres ou des formats déjà normalisés, suffisants pour des démonstrations en laboratoire mais insuffisants face aux conditions réelles : pages qui se chargent au défilement, sélecteurs instables, doublons, champs manquants, ou encore compétences implicites qu'il faut inférer du contexte. Les systèmes entraînés ou évalués sur de telles données idéalisées perdent en robustesse lorsqu'ils rencontrent la complexité du terrain.

Dans ce cadre, JobMatch propose une approche intégrée et réaliste : (1) collecte d'offres via scraping robuste (y compris contenus JavaScript), (2) extraction et normalisation des informations clés côté annonces et CV, (3) indexation sémantique et retrieval de preuves, (4) scoring multi-critères (must-have/bonus, localisation, langue) et (5) explications traçables des recommandations. La coordination multi-agents (orchestrée via un protocole de communication tel que l'ACP) et une interface Streamlit assurent un fonctionnement modulaire

1.3 Objectif :

La question centrale de ce stage est la suivante :

Comment exploiter un LLM pour automatiser et améliorer la pré-sélection des CV tout en évitant les biais ?

Cette problématique s'inscrit dans le contexte du recrutement dans le domaine des technologies de l'information, où le volume croissant de candidatures et la diversité des profils rendent l'analyse manuelle des CV chronophage et parfois subjective.

Les objectifs spécifiques du stage sont :

1. **Analyser l'existant** : étudier les méthodes et outils actuels de pré-sélection des candidatures.
2. **Choisir et adapter un modèle de LLM** : sélectionner une architecture adaptée au traitement des CV et des offres d'emploi.
3. **Développer une preuve de concept (POC)** : mettre en place une solution modulaire capable d'extraire les informations clés des CV (compétences, expériences, formations), de les comparer sémantiquement avec les offres et de produire un score de pertinence.
4. **Évaluer les résultats** : mesurer la fiabilité, la cohérence et l'impartialité de la solution proposée.

L'objectif final est de fournir un outil intelligent d'aide au recrutement, accélérant la pré-sélection, améliorant la qualité de la décision et réduisant la sensibilité aux biais humains.

Dans un premier temps, nous présenterons l'état de l'art des approches existantes en matière de traitement automatique des CV et de recrutement assisté par IA. Nous détaillerons ensuite la réalisation de la solution proposée, depuis la conception de l'architecture logicielle jusqu'au développement du prototype. Enfin, nous discuterons des résultats obtenus et des perspectives d'amélioration.

État de l'art et Méthodologie

Sommaire

2.1	Les Grands Modèles de Langage (LLMs)	7
2.2	Architecture des LLMs	7
2.2.1	Transformers	7
2.2.2	Fonctionnement des LLMs	8
2.3	Automatisation du Pré-sélection des candidats	9
2.3.1	Pré-sélection avec LLMs.	9
2.3.2	Importance des LLMs pour la pré-sélection	10

2.1 Les Grands Modèles de Langage (LLMs)

Un **LLM (Large Language Model)**, ou **modèle de langage de grande taille**, est un modèle d'intelligence artificielle basé sur l'architecture **Transformer**. Il est entraîné en **auto-supervision** sur des volumes massifs de texte pour comprendre, générer ou compléter du langage naturel. Ces modèles sont capables de réaliser une variété de tâches : traduction, résumé, génération de texte, réponses aux questions, etc.

2.2 Architecture des LLMs

2.2.1 Transformers

Les Transformers, proposés en 2017, sont l'architecture clé du traitement automatique du langage. Leur objectif est de modéliser efficacement les dépendances contextuelles entre mots, quelle que soit leur distance, grâce à un mécanisme d'attention multi-têtes. Chaque bloc combine attention, réseau feed-forward et normalisations, assurant stabilité et parallélisation, ce qui a grandement amélioré la compréhension et la génération de texte.

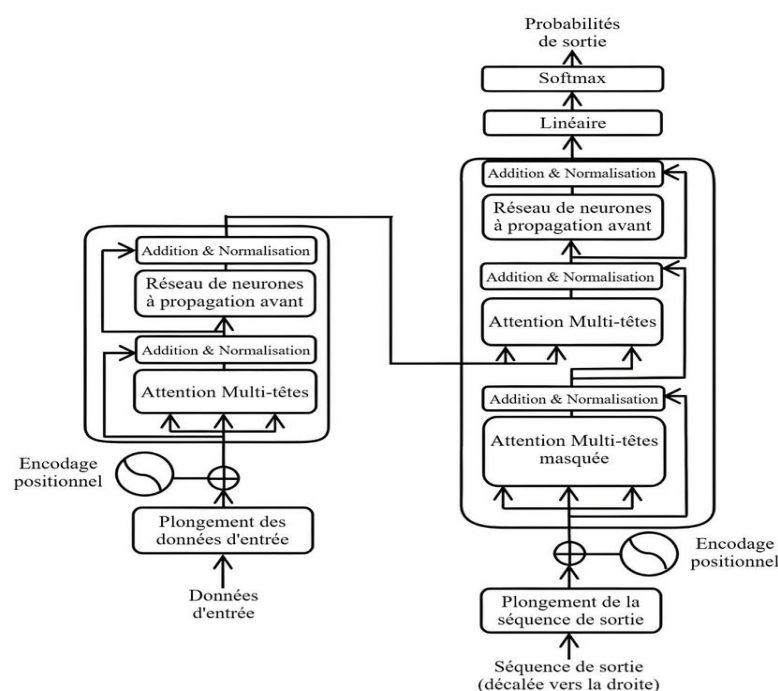


FIGURE 2.1 – Schéma représentant l'architecture générale d'un transformeur.

2.2.2 Fonctionnement des LLMs

Les grands modèles de langage (LLMs), majoritairement basés sur l'architecture des Transformers, sont des réseaux neuronaux profonds conçus pour traiter et générer du texte en langage naturel. Leur fonctionnement repose sur plusieurs composants et étapes techniques.

Ils traitent et génèrent du texte en plusieurs étapes. Le texte est d'abord découpé en *tokens* (mots ou sous-mots), puis chaque token est représenté par un vecteur (*embedding*) enrichi par des informations de position.

L'architecture repose sur le mécanisme d'**attention multi-têtes** qui capture les relations entre tokens, des **réseaux feed-forward** qui transforment les représentations, et des **couches de normalisation et connexions résiduelles** qui stabilisent et facilitent l'apprentissage.

L'entraînement se fait sur de vastes corpus avec des objectifs comme la prédiction du prochain token (GPT) ou le masquage de tokens (BERT), optimisé par des algorithmes tels qu'Adam, avec régularisation (dropout, pondération).

En phase d'inférence, le modèle génère une suite de tokens probables à partir d'un *prompt*, grâce à des techniques comme le sampling, le top-k ou le beam search, puis applique une fonction softmax pour produire des probabilités.

Les modèles peuvent ensuite être adaptés à des tâches précises via le **fine-tuning**, optimisés par le **prompt engineering**, et rendus plus efficaces grâce à la distillation ou la quantification.

L'objectif final est de construire un système capable de modéliser le langage humain de manière flexible et intelligente, utilisable dans des applications concrètes comme les assistants IA, chatbots ou outils de traduction.

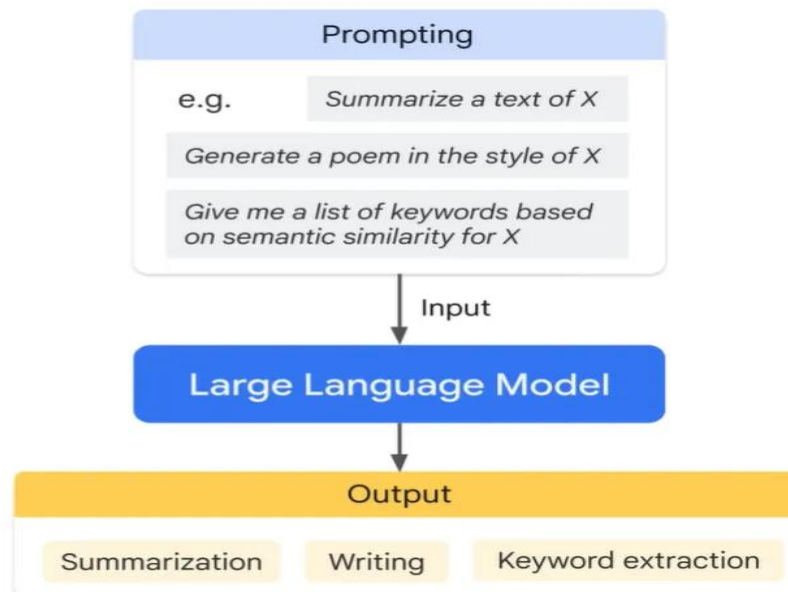


FIGURE 2.2 – Logo de fonctionnement des LLMs.

2.3 Automatisation Du Présélection des Candidats

Le recrutement est un processus chronophage, en particulier lors de la phase de présélection où il faut analyser manuellement un grand nombre de CVs. Ce projet a pour objectif de développer une solution d'automatisation de la présélection des candidats à l'aide de l'intelligence artificielle, et plus précisément des modèles de langage de grande taille (LLMs).

2.3.1 Pré-sélection Avec LLMs :

Dans le cadre de ce projet, la pré-sélection des candidats est assurée par le modèle *MistralAI-7B*, un modèle de langage de dernière génération. Ce LLM est chargé d'analyser les CVs en langage naturel, après extraction du texte brut, afin d'en extraire automatiquement les informations clés (identité, compétences, expériences, etc.). Cette étape repose sur l'utilisation d'un prompt soigneusement conçu, afin d'obtenir des résultats cohérents, structurés et exploitables.

Une fois les données extraites, un deuxième prompt permet au modèle de comparer les compétences du candidat avec celles exigées par le poste. Cette logique de comparaison repose sur

un raisonnement implicite du modèle, guidé par les critères métier définis en amont (compétences techniques, mots-clés, etc.).

Enfin, l'ensemble des résultats est affiché dans une interface utilisateur, développée à l'aide de bibliothèques Python adaptées (notamment Streamlit). Celle-ci permet de visualiser les scores de correspondance, les points forts du profil, ainsi que les éventuelles compétences manquantes.

2.3.2 Importance des LLMs pour la pré-sélection :

L'utilisation des LLMs est cruciale dans ce projet car ils constituent le cœur du système de pré-sélection automatisée des CV.

Contrairement aux méthodes traditionnelles de filtrage qui reposent sur des mots-clés ou des règles figées, les LLMs permettent de comprendre le contenu textuel de manière sémantique et contextuelle, ce qui est indispensable pour analyser des documents non structurés comme les CVs et les comparer efficacement aux descriptions de poste. Ils rendent possible l'extraction intelligente des informations clés (compétences, expériences, formations), la mise en relation fine avec les besoins exprimés dans une offre, et la génération d'un score de pertinence objectif.

Cette capacité d'analyse avancée renforce la fiabilité et la rapidité du processus de recrutement, tout en limitant les biais liés aux approches purement humaines ou aux systèmes trop simplistes.

Réalisation et implémentation

Sommaire

3.1 Missions Confiées	12
3.2 Architecture globale du système	13
3.3 Implémentation	14
3.3.1 Langages et Bibliothèques	14
3.3.2 Outils de Développement	15
3.3.3 Pré-traitement des Données (CV) et Prompt Engineering	16
3.3.4 Intégration et Fine-Tuning du Modèle (MistralAI-7B).	16
3.3.5 Le développement de l'interface.	16
3.3.6 Difficultés rencontrées	17

3.1 Missions Confiées

Dans le cadre de ce projet, plusieurs missions ont été réalisées pour atteindre les objectifs d'automatisation la Pré-Sélection :

- **Analyse du besoin fonctionnel** : La première mission consiste à formaliser les besoins des responsables RH et Techniques. Au but de Comprendre les critères de sélections généralement utilisés (compétences techniques, soft skills, expérience, formation) et de définir les éléments à extraire et valoriser dans un CV.
- **Étude de l'état de l'Art** : Une veille technique a été menée afin d'identifier les approches existantes dans le domaine du recrutement automatisé. Cette phase a porté sur les méthodes de parsing de CVs, ainsi que sur les modèles de langage pré-entraînés (LLMs) tels que GPT ou Mistral.
- **Mise en place d'un système de parsing de CV** : Un module d'extraction d'informations à partir de CVs a été développé afin d'identifier automatiquement les éléments essentiels tels que le nom, l'adresse e-mail, les compétences, les expériences professionnelles et les diplômes. Ce module s'appuie à la fois sur l'utilisation d'un modèle de langage génératif et sur des listes de mots-clés pour renforcer la précision des extractions.
- **Développement d'un moteur de Matching** : À partir des données extraites, un moteur de comparaison a été mis en place pour évaluer la correspondance entre un profil candidat et le profil nécessaire. Chaque CV se voit attribuer un score de compatibilité calculé sur la base des compétences détectées, des écarts identifiés par rapport aux exigences du poste, et de la cohérence globale du parcours. Le système génère également une explication textuelle accompagnant le score, mettant en lumière les compétences présentes et absentes.
- **Automatisation du processus** : Une interface simple a été développée à l'aide du Framework « Streamlit », afin de faciliter l'exploitation des résultats. Celle-ci permet de visualiser les profils les mieux classés et de consulter le détail du score attribué à chaque candidat.

- **Déploiement d’une interface utilisateur** : Les différents modules (parsing, évaluation, sélection) ont été intégrés dans un pipeline automatisé permettant de traiter l’ensemble des CVs de manière fluide et reproductible. Ce flux peut être déclenché automatiquement à la réception d’une nouvelle offre, garantissant un gain de temps et une standardisation du processus de tri.
- **Tests, validation et documentation** : Une phase de validation a été conduite afin de tester la robustesse et la cohérence des résultats produits. Les seuils de sélection ont été ajustés en fonction des cas observés, et une documentation technique détaillant l’architecture, le fonctionnement et les choix technologiques a été rédigée en vue de la maintenance et de l’évolution du système.

3.2 Architecture Globale du Système

Le système se compose des modules suivants :

- ✓ Extraction de texte depuis les CVs
- ✓ Parsing avec un LLM
- ✓ Matching des compétences et calcul de score.
- ✓ Interface utilisateur pour visualisation.

Ces modules sont réalisés avec une structure du projet qui contient les agents et les outils nécessaires.

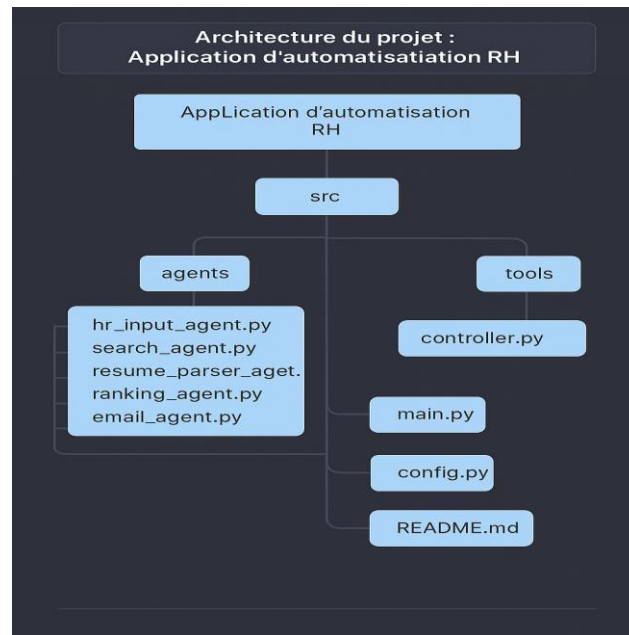


FIGURE 2.3 – Schéma de la structure générale du Projet.

3.3 Implémentation

3.3.1 Langages et Bibliothèques

- **Langages**

Le projet a principalement été développé en *Python*, en raison de sa flexibilité et de la richesse de son écosystème de bibliothèques pour la génération des textes et l'implémentation dans une interface.

- **Bibliothèques Utilisées**

- **Transformers** : Fournie par Hugging Face, permet d'utiliser des modèles de langage pré-entraînés (LLMs). Elle sert ici à charger un modèle (AutoModelForCausalLM) et son tokenizer, puis à exécuter des générations de texte via le pipeline.
- **Torch** : Bibliothèque de calcul tensoriel et base de nombreux modèles de deep learning. Elle est utilisée ici comme backend pour exécuter les modèles IA (via PyTorch).
- **Re** : Fournit des outils pour le **traitement de texte par expressions régulières**, utile ici pour extraire ou nettoyer des éléments spécifiques dans un CV (ex. : e-mail, numéro de téléphone).
- **PyPDF2** : Bibliothèque permettant l'extraction de texte à partir de fichiers PDF, utilisée ici pour récupérer le contenu des CVs soumis par les utilisateurs. CVs soumis par les utilisateurs.
- **OS** : Permet d'interagir avec le système d'exploitation, par exemple pour accéder à des variables d'environnement ou gérer des chemins de fichiers.
- **Dotenv** : Sert à charger des variables d'environnement depuis un fichier « .env », souvent utilisé pour stocker des clés API ou des chemins sensibles sans les exposer dans le code.
- **Streamlit** : Framework léger pour créer une interface web interactive en Python. Il est utilisé ici pour afficher les CVs analysés, les scores de correspondance, et permettre à l'utilisateur d'interagir avec l'application (ex. : chargement de fichiers, visualisation de résultats).

3.3.2 Outils de Développement

Dans le cadre de ce projet, l'environnement de développement *Visual Studio Code (VS Code)* a été utilisé pour le développement, l'organisation et le suivi du code source. Il s'agit d'un éditeur de code moderne, léger et extensible, largement adopté par la communauté des développeurs.



FIGURE 3.3.2 – Logo de VS Code.

3.3.3 Pré-traitement des Données (CV) et Prompt Engineering

Mise en place d'un pipeline d'extraction (PDF/DOCX → texte, OCR si nécessaire), nettoyage des documents, détection de la langue, normalisation des intitulés et anonymisation des informations sensibles afin de limiter les biais. Ce processus a été complété par la génération de représentations exploitables (features structurées et embeddings sémantiques) ainsi que la définition de prompts robustes avec des sorties structurées et validées automatiquement.

3.3.4 Intégration et Fine-Tuning du Modèle (MistralAI-7B)

Étude des options disponibles, entre API commerciales (ex. : OpenAI) et modèles open source/self-hosted (LLaMA, Mistral via Hugging Face).

Pour ce projet, l'intégration **locale** de **MistralAI-7B** a été privilégiée afin de mieux maîtriser les données. L'approche retenue repose sur la quantification pour optimiser les ressources, et un fine-tuning ciblé sur des jeux annotés. Les modèles ont été évalués selon des critères de précision, de latence, de coût et de respect de la confidentialité.

3.3.5 Le développement de l'interface :

Une interface utilisateur a été conçue (prototype sous Streamlit puis version web plus robuste) permettant d'afficher le classement des candidats, les scores détaillés et les

preuves textuelles. Des mécanismes de sécurité et de traçabilité (authentification, journaux d'audit, export CSV/JSON) ont été intégrés pour faciliter le contrôle RH.

3.3.6 Difficultés rencontrées :

Le développement du projet a présenté plusieurs défis techniques, mais aussi des résultats positifs :

- **Extraction incohérente des CVs** : Certains CVs présentaient des structures non standardisées (colonnes, tableaux, zones en image, etc.), rendant l'extraction du texte parfois incomplète ou désorganisée, même avec des bibliothèques comme PyPDF2.
- **Réponses imprévisibles du LLM** : Malgré un prompt bien formulé, le modèle MistralAI-7B ne retournait pas toujours un format JSON exploitable, ce qui a nécessité l'ajout de règles de validation et de nettoyage supplémentaires.
- **Dépendance à la qualité des données** : Certains CVs étant mal rédigés ou incomplets, cela pouvait fausser l'analyse automatique, surtout dans le cas des compétences implicites ou peu explicites.

3.3.7 Résultats obtenus :

- **Affichage clair des résultats dans l'interface** : plusieurs volets d'évaluation ont été mis en place, incluant des mesures opérationnelles (latence, throughput, gain de temps par CV traité) et des retours qualitatifs des recruteurs. Les résultats sont présentés sous forme de tableaux comparatifs (baseline mots-clés vs pipeline LLM), de courbes de performance, de captures d'écran de l'interface et d'extraits JSON illustrant la transparence des scores.

— **Amélioration continue du prompt** : Le prompt utilisé pour guider le modèle a été optimisé tout au long du projet. Plusieurs itérations ont été nécessaires pour obtenir des réponses structurées, cohérentes et adaptées aux besoins RH. Des ajustements ont été faits pour affiner la formulation, l'ordre des instructions, et la clarté des champs attendus.

— **Intégration d'un modèle de langage de nouvelle génération** : Le modèle MistralAI-7B a été intégré localement dans le projet afin d'éviter toute dépendance à des APIs externes. Sa capacité à comprendre le langage naturel et à générer des réponses précises a permis de traiter les CVs de manière intelligente et contextuelle, tout en conservant la maîtrise des données en interne.



CONCLUSION GÉNÉRALE

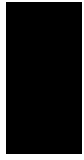
Ce stage a constitué une opportunité enrichissante pour explorer l'application concrète de l'intelligence artificielle dans un domaine en pleine transformation : les ressources humaines. Le projet mené avait pour objectif de concevoir une solution automatisée de présélection des candidats, capable de traiter un grand volume de CVs de manière fiable, rapide et intelligente.

Grâce à l'intégration d'un modèle de langage avancé (MistralAI-7B) et à une architecture modulaire bien définie, le système développé permet d'extraire les informations clés des candidatures, de les comparer automatiquement à une offre d'emploi, puis de classer les profils en fonction de leur pertinence. Ce processus, auparavant manuel et chronophage, a été considérablement simplifié et optimisé.

L'interface utilisateur développée facilite la visualisation des résultats et l'interaction avec les candidats, notamment via l'envoi automatique d'e-mails d'invitation à un entretien. Ce projet a ainsi démontré qu'il est possible d'associer intelligence artificielle et processus RH de manière efficace et cohérente, sans sacrifier la transparence ni la qualité des décisions.

En plus de l'acquisition de compétences techniques en développement Python, traitement du langage naturel et déploiement d'interface, ce stage a permis de mieux comprendre les besoins réels d'un service RH et les contraintes liées au recrutement dans un secteur concurrentiel comme l'IT.

Ce travail pose les bases d'une solution évolutive, qui pourrait être adaptée à d'autres contextes ou améliorée par l'ajout de nouvelles fonctionnalités. Il s'inscrit pleinement dans la dynamique actuelle de modernisation des outils RH, où la technologie devient un levier stratégique au service de la performance humaine.



Bibliographie