**Institut National des Langues et Civilisations Orientales**

Département Textes, Informatique, Multilinguisme

# Optimisation du processus de recrutement par utilisation de méthodes d'intelligence artificielle

## MASTER

## TRAITEMENT AUTOMATIQUE DES LANGUES

*Parcours :*

*Ingénierie Multilingue*

par

**Suhaib ETARHUNI**

*Directeurs de mémoire :*

SARRA EL AYARI
DAMIEN NOUVEL

*Encadrante :*

LAMIAE MKADMI

Année universitaire 2018/2019

# Remerciements

Mes premiers remerciements vont évidemment à Lamiae Mkadmi (Data Scientist), ma tutrice de stage au sein d'Expleo Group, pour sa confiance, son temps et ses conseils tout au long de mon projet.

Je tiens à remercier également mes collègues au sein du pôle Data Science d'Expleo Group pour leur accueil sympathique et leur soutien permanent.

Je remercie mes encadrants de mémoire Sarra El ayari et Damien Nouvel pour leur suivi, leurs disponibilités et leurs encouragements tout au long de la rédaction de ce mémoire. Mes vifs remerciements vont également à toute l'équipe pédagogique de pluriTAL notamment Serge Fleury, et Jean-Michel Daube pour leur attention, leur patience et leur aide permanente tout au long du master.

Enfin, mes plus profonds remerciements vont à mes parents et toute ma famille pour leur amour, leurs conseils ainsi que leur soutien inconditionnel qui m'a permis de réaliser les études que je voulais et par conséquent ce mémoire.

# Sommaire

# Liste des figures

# LISTE DES TABLEAUX

# RÉSUMÉ

La croissance exponentielle des données a permis l'évolution des nouvelles technologies dans de nombreux secteurs. Parmi eux, le marché du travail, qui a connu le développement d'un grand nombre de sites d'offres d'emploi (*Jobboards*) comme par exemple LINKEDIN [1], MONSTER [2] qui ont permis de développer un marché de recrutement en ligne.

Notre sujet porte sur le développement d'un outil permettant d'optimiser le temps de traitement des données lors du processus de recrutement. Ceci est en automatisant la présélection et l'évaluation des profils des candidats. L'objectif est de fournir un score de compatibilité entre une offre et chaque candidat, ce score reflète à quel point son profil correspond aux spécifications du poste.

Notre travail se concentre sur l'analyse de CVs ainsi que les besoins métier du service de recrutement. Nous présentons d'abord un système capable de lire en profondeur les CVs des candidats et les segmenter grâce aux modèles d'apprentissage automatique. Enfin, ce système est capable également de proposer les postes les plus pertinents pour le ou les candidats de l'entreprise en se basant sur la similarité des informations extraites du CV et les besoins RH.

**Mots-clés** : Recrutement, apprentissage automatique, traitement automatique des langues, classification de textes, scoring, ordonnancement de candidats, ranking candidates

1. www.linkedin.com
2. www.monster.com

# INTRODUCTION

## Contexte et objectifs des travaux

Ce mémoire a été rédigé en parallèle du stage de fin d'étude au sein du pôle Data Science de l'entreprise Expleo Group [3]. Cette dernière est une société mondiale d'ingénierie et de conseil stratégique pour la transformation digitale. Expleo Group est présente dans 25 pays avec plus de 15 000 collaborateurs sur différents secteurs notamment l'aéronautique, l'automobile, le ferroviaire, et le secteur de la défense.

Mon sujet de stage fait partie d'un projet interne d'Expleo nommé « *Digital Human Resources* ». Ce projet, lancé en 2018, est destiné aux différents services ressources humaines afin de développer des applications de monitoring, de visualisations, de suivi, en intégrant des méthodes d'intelligence artificielle.

Le but de mon projet, qui est en réalité un sous-projet de « DIGITAL HR », est de développer un outil d'aide à la décision. Il est destiné spécifiquement au service recrutement pour traiter automatiquement les données des candidats, et mettre en correspondance le profil du candidat avec le ou les offres d'emploi adéquates. Cet outil va prioriser les profils à étudier et gérer les flux importants de demandes d'emplois de manière plus rapide.

Les technologies de l'information et de la communication ont introduit de nouvelles pratiques dans les fonctions de gestion des ressources humaines telles que le recrutement électronique. Les demandeurs d'emploi soumettent leur curriculum vitae (CV) sur le Web ou ils l'envoient directement à une entreprise. L'enjeu est donc d'accélérer le processus de traitement des candidatures. En effet, chez Expleo on compte entre 60 et 80 entretiens physiques par semaine, et plus de 200 candidatures par offre publiée sur Internet. Ce chiffre peut varier selon le secteur car il peut vite augmenter surtout quand il s'agit d'une offre de poste technique telle que Data Scientist, ingénieur mécanique etc.
Un chargé de recrutement doit gérer les candidatures reçues des différents sites d'emplois, les candidatures spontanées, mais il doit aussi être capable d'identifier et prendre contact avec des candidats qui lui semblent intéressants même s'ils n'ont pas postulé. Cette dernière action est appelée le « sourcing des candidats ».

3. https://expleogroup.com/fr/

Le traitement de données des candidats est un travail fastidieux et coûteux en temps. Le chargé de recrutement doit en effet dans un premier temps traiter tous les CVs un par un, puis les filtrer selon leur pertinence. Ensuite il faut extraire les informations et les entrer manuellement dans la base de données interne. Ce processus est nommé « Présélection des candidats ». Il va être suivi d'autres processus comme passer un entretien physique, prendre une décision, faire une proposition, etc.

C'est à partir de ce processus traditionnel qu'est née l'idée de développer un outil capable d'analyser automatiquement les CVs et en extraire les informations pertinentes. Le filtrage de ceux qui sont en adéquation avec les demandes de l'entreprise a également pour but d'être automatisé afin de simplifier autant que possible le travail des chargés de recrutement.

# Première partie

# Contexte général

CHAPITRE 1

# ÉTAT DE L'ART

Dans ce chapitre, nous présentons les travaux réalisés concernant les méthodes de traitement automatique de la langue (TAL) ainsi que les méthodes de *Machine Learning* appliquées aux besoins du service de ressources humaines, et plus particulièrement aux services recrutement et gestion de carrière. Différentes approches existent et des méthodes variées sont appliquées. A l'heure actuelle, malgré l'existence de plusieurs prototypes et de programmes pour automatiser le processus de recrutement, peu de systèmes sont mis en production pour automatiser la présélection des candidats.

Il existe des entreprises comme Riminder[1], Smartrecruiters[2], et Digitalrecruiters[3] qui proposent des outils pour automatiser la gestion de ressources humaines. Ils accompagnent leurs clients pour transformer leurs données RH et créer des applications d'aide à la décision, de diffusion d'offres d'emplois, de gestion de candidatures etc. La transformation digitale est une phase très complexe, surtout quand il s'agit du domaine des ressources humaines dans lequel les données ne sont pas forcément faciles à traiter. En effet, le domaine des ressources humaines est un domaine très riche, il recouvre l'ensemble des pratiques mises en œuvre pour administrer et développer le capital humain de l'entreprise (17).

Nous allons nous concentrer dans ce mémoire sur le traitement automatique des données d'une branche du domaine des ressources humaines qui est le recrutement électronique.

## 1.1 Le e-recrutement

Le terme « recrutement » est défini dans le dictionnaire de la manière suivante : « Action de choisir et de sélectionner une personne pour intégrer une structure, une association, une entreprise »[4]. (3) définit le recrutement comme étant le processus de recherche et d'engager les personnes dont l'organisation

1. www.riminder.net
2. www.smartrecruiters.com
3. www.digitalrecruiters.com
4. www.linternaute.fr

a besoin. (4), quant à lui, définit le recrutement électronique, (*e-recruitment*), comme étant un ensemble d'outils qui utilise différentes techniques, telles que le traitement automatique du langage naturel, et l'intelligence artificielle, afin de traiter automatiquement les informations contenues dans les curriculum vitae et les offres d'emploi.

Dans l'approche traditionnelle, les agences de ressources humaines doivent affecter des chargés de recrutement pour examiner manuellement les candidatures. L'objectif est d'évaluer l'adéquation des candidats aux postes à pourvoir.

Aujourd'hui, il existe de nombreux sites Internet consacrés au recrutement. Le recruteur peut publier des offres d'emploi sur des sites Internet regroupés sous l'appellation de « *jobboards* ». Mais il peut aussi rechercher des profils de candidats inscrits sur ces sites, comme « monster », ou sur les réseaux sociaux professionnels tels que « linkedIn » où les candidats peuvent rendre leur CV disponible en ligne. Les réseaux professionnels sont désormais très utilisés et permettent également aux recruteurs de voir les réseaux professionnels des candidats. Le *sourcing* peut se définir comme étant « l'identification des profils précis répondant à une liste de critères de sélection pour un poste professionnel donné. ». Le *sourcing* utilisant un support numérique et largement utilisé est connu sous le nom de « e-recrutement » (13).

Selon (3) les processus de recrutement et de sélection peuvent être divisés en 10 étapes :

1. définir les critères,
2. attirer les candidats,
3. présélection,
4. entretien des candidatures,
5. test,
6. évaluer les candidats,
7. obtenir des références,
8. vérifier les candidatures,
9. faire une proposition d'embauche,
10. suivi.

(4) montre que les deux premières étapes concernent le processus de recrutement, alors que les huit dernières concernent la partie de sélection. C'est un processus coûteux en temps. Cependant le recrutement électronique peut automatiser le processus de présélection des profils de candidats avec une efficacité maximale et un faible coût. Ceci est prouvé dans les travaux de (16) pour SAT telecom [5] où le temps de traitement a diminué de 70 jours à 37 jours pour un poste donné en utilisant un programme automatique de présélection.

5. www.sat-telecom.net

Selon (8) la partie sélection commence par le traitement du CV du candidat. Le traitement des données du candidat est la base pour tous les projets qui visent l'automatisation du processus de recrutement. En effet pour identifier les meilleurs candidats, il faut que l'on dispose d'une méthode robuste pour analyser et extraire les informations pertinentes du CV. Or ce dernier n'a pas un format universel. Chaque CV a un format qui lui est propre, ce qui rend l'analyse et l'extraction très complexes.

Le e-recrutement propose plusieurs fonctionnalités. Dans ce mémoire, nous nous intéressons au classement des offres d'emploi en fonction du profil du candidat.

## 1.2 Travaux visant à identifier les candidatures les plus adéquates selon une offre d'emploi

Divers travaux ont été publiés afin de classer les candidatures reçues de la plus à la moins pertinente, ce qui est appelé le « ranking candidates ». Il existe plusieurs techniques pour le classement des candidats. Dans ce mémoire, nous nous concentrons sur les méthodes de traitement automatique des langues et des méthodes de *Machine Learning* utilisées pour analyser le CV du candidat et l'offre d'emploi.

Tout système de recrutement en ligne a besoin de deux éléments au minimum : le curriculum vitae et une offre d'emploi. Le CV représente le candidat et les offres d'emploi représentent le recruteur. Ils contiennent différents types d'informations et de points de vue qui peuvent être utilisés pour créer des systèmes permettant de classer les candidats en fonction d'une offre d'emploi. Parmi ces systèmes, le *Prospect* proposé par (19) extrait les compétences et la formation du candidat en les présentant sous forme facette pour ensuite proposer un classement selon une offre d'emploi.

Une autre approche présentée par le système *EXPERT* de (12) ordonne les candidats en trois étapes. La première consiste à définir une ontologie construite à partir des informations du candidat. La deuxième étape s'appuie sur une ontologie pour représenter l'offre d'emploi. Enfin, pour la dernière étape, le système met en corrélation les deux représentations et propose un classement avec les candidats les plus adéquats.

Les ontologies ont aussi fait l'objet des travaux de (14) en proposant *LO-MATCH* qui est une plate-forme développée pour le projet européen *MATCH1*. Son objectif était de faire « matcher » les curriculum vitae et les offres d'emploi par l'analyse des compétences professionnelles. Le système

LO-MATCH est basé sur l'utilisation d'ontologies pour enrichir les informations contenues dans les CV et les offres d'emploi. L'objectif est de faciliter le processus de recherche, respectivement pour les demandeurs d'emploi et les recruteurs. Pour être plus précis, LO-MATCH utilise une ontologie créée par des experts du domaine et la base de données lexicale WordNet8 pour classer les offres d'emploi et les CVs. Parmi les caractéristiques qui rendent ce système remarquable, il y a la possibilité d'indiquer aux candidats des compétences manquantes ou peu développées. De plus, il propose aux recruteurs des compétences alternatives pour leur offre d'emploi.

La notion d'ontologie est présente aussi dans les travaux de (21) qui adoptent l'annotation sémantique des documents pour automatiser le processus de e-recrutement. L'idée consiste à modéliser le contenu sémantique de ces documents en termes de leurs acquis/requis en utilisant une ontologie commune.

D'autres auteurs ont choisi les techniques de traitement automatique des langues, notamment le système *E-Gen* proposé par (11). Il utilise la représentation vectorielle des textes (*WORDS EMBEDDINGS*) pour ensuite mesurer la similarité entre le vecteur du CV et le vecteur de l'offre d'emploi afin de proposer un classement des candidats. Les auteurs ont décidé d'intégrer un *relevance feedback* du recruteur, c'est-à-dire que les recruteurs ont été invités, par une simulation, à donner leur avis sur un petit ensemble de CV. Ensuite, l'entrée donnée par le recruteur a été utilisée pour affiner le classement des CVs. Les résultats obtenus ont été améliorés avec le feedback du recruteur, le système atteint 66% d'AUC [6].

(6) mettent en place des automates capables d'identifier des catégories de CV, de profils de candidats et/ou de postes. La première approche consiste à catégoriser des CVs de cadres et de non-cadres en extrayant des termes spécifiques permettant une classification et un modèle à base d'analyse discriminante. Cette méthode met en avant la spécificité de certains termes (niveau d'étude, compétences, expériences professionnelles, etc) afin d'effectuer une classification. Les résultats restent modestes environ 50% à 60% des CVs sont correctement classés.

Le système proposé par (23) se base aussi sur un découpage en sections pour extraire les informations personnelles, la formation, l'expérience, ainsi que la liste de compétences des candidats. Le système passe ensuite par une phase de filtrage qui affine les listes de concepts. Le système supprime les concepts ayant une faible pondération tf-idf, et ceux qui ne contribuent pas à la sémantique de chaque segment. Les auteurs intègrent une base de connaissances

6. AUC (Area under curve) signifie "aire sous la courbe ROC". Cette valeur mesure l'intégralité de l'aire à deux dimensions située sous l'ensemble de la courbe ROC (par calculs d'intégrales) de (0,0) à (1,1)

qui combine deux ressources sémantiques principales : DICE1 [7] le centre de compétences et O * NET2 [8], une hiérarchie standardisée de catégories professionnelles connue sous le nom de réseau d'information professionnelle (O * NET)

Leur module de *matching* (correspondance CV et offre d'emploi) prend en entrée la liste de compétences issues à la fois de curriculum vitae et de l'offre d'emploi et déduit la relation sémantique pour proposer une similarité sémantique à l'aide des ressources mentionnées.

Une autre approche qui a été proposée dans les travaux de (2) consiste à transformer les CVs en format XML structuré en utilisant GATE [9]. Produit par l'Université de Sheffield, GATE est un environnement de développement qui permet aux utilisateurs de développer et de déployer des composants et des ressources d'ingénierie linguistique. Il contient différents modules permettant de traiter des documents texte en différents formats. Les auteurs se basent sur les modules de GATE pour analyser les CVs, notamment le « Tokeniser », « Sentence Splitter », « Pos Tagger », « Named Entity » ainsi que des règles linguistiques d'extraction nommées JAPE. Les résultats de la classification varient entre 0.79 et 0.97 du F-mesure (mesure d'évaluation très courante en apprentissage automatique).

L'idée que nous développons ici est proche des travaux de (22). Ils ont mis en place un système de correspondance (Matching) entre CV et offres d'emplois en se basant sur l'extraction d'informations. Ce système utilise les algorithmes de classification de Hidden Markov Model (HMM) et l'algorithme de Support Vector Machine (SVM) afin d'annoter les segments de curriculum vitæ avec la catégorie appropriée.
En conséquence, les CVs passent à travers deux couches. Dans la première couche un HMM est appliqué pour segmenter le CV en blocs consécutifs et chaque bloc est annoté avec une catégorie telle que Informations personnelles, Éducation et Expérience, etc. Dans la deuxième couche, un modèle SVM est appliqué afin d'extraire les informations détaillées des blocs étiquetés. Néanmoins les auteurs ont mentionné une faible précision car le processus d'extraction d'informations passe par deux étapes faiblement rattachées, à savoir la première avec les modèles HMM pour segmenter le CV, et la deuxième avec le modèle SVM.

Tout système de classement de candidats est divisé généralement en deux grandes étapes. La première traite le CV et la deuxième traite l'offre d'emploi. Néanmoins la première étape est extrêmement difficile car comme nous l'avons mentionné, chaque CV est différent en répondant à une structure conceptuelle propre à chaque individu et difficile à modéliser. L'enjeu principal lorsqu'on crée un analyseur automatique de CVs est d'identifier les différentes

7. https://www.dice.com/skills
8. https://www.onetcenter.org/taxonomy/2010/list.html
9. https://gate.ac.uk/

sections mentionnées (Éducation, Compétences, etc.) sans perdre d'informations.

## 1.3 Les curriculums vitaes

Un CV est généralement composé de cinq sections comme plus haut la liste ci-dessous :

1. Formations ou diplômes,
2. Expériences professionnelles,
3. Compétences,
4. Langues,
5. Loisirs.

L'idée des systèmes de e-recrutement est de réunir les candidats et les recruteurs. Néanmoins le résultat d'un système destiné au chargé de recrutement sera différent de celui qui sera destiné au candidat. Le premier proposerait un classement des candidats selon une offre d'emploi avec un score de compatibilité par exemple, alors que le deuxième proposerait les offres d'emploi qui sont en adéquation avec le profil du candidat.

Dans tous les cas, le profil du candidat doit être analysé en premier lieu. L'analyseur automatique permet de passer le profil du candidat au crible. Par exemple, les compétences indiquées seront analysées et en quelques secondes s'afficheront tous les postes qui correspondent aux candidats. L'analyse du CV commence par une première étape qui est le découpage en sections. Cette étape est la plus importante car c'est celle sur laquelle vont se baser toutes les autres étapes. Notamment l'étape de l'extraction d'information, ainsi que l'étape de classement. Par exemple, pour extraire le nom de l'école du candidat, il faut d'abord parvenir à identifier le bloc « Formation & diplômes ».

Nous allons présenter différentes méthodes de découpage en sections utilisées. L'analyseur automatique « SegCV » de (7) met en oeuvre deux approches. La première est basée sur la structure du CV : les titres, les sous-titres ou les débuts des lignes. Les règles de découpage sont basées sur 94 expressions régulières. La deuxième approche tente d'améliorer le découpage en comparant la taille de chaque section avec la taille du CV. Si la taille de la section est anormalement grande (ou petite), ils font appel aux mots-clés pour restructurer les sections. Par exemple, si un fragment de texte dans la section « Compétences » contient les mots « célibataire » ou « situation de famille » ce fragment sera déplacé vers la section « Identité ».

Nous nous sommes inspiré de ces travaux pour mettre en place une approche symbolique de découpage en sections des CVs structurés que l'on expliquera en détail dans le chapitre suivant.

Une autre approche basée sur les techniques de traitement automatique des langues est présentée dans les travaux de (18). Ils s'appuyaient sur un corpus de CVs fourni par la société « Vedior Bis » afin d'extraire des patrons morpho-syntaxiques et de les classer en fonction d'une ontologie spécialisée. L'étude menée consistait à définir des collections que l'on peut trouver dans un CV pour ensuite extraire les patrons « Nom, Verbe, et Adjectif » et calculer la performance de la classification selon la taille de la collection.

Nous nous sommes également inspiré des travaux de (23) qui utilisent les techniques de TAL comme les n-gram, la tokenization, la suppression les mots vides« *stop words* », l'annotation en parties de discours (*Part-of-Speech-Tagging*) et la reconnaissance des entités nommées, afin de repérer les caractéristiques de chaque section. Par exemple, on peut penser que la section expériences professionnelles contiendra plus de verbes que les autres sections car les candidats ont tendance à détailler leurs expériences. On imagine aussi que la section Formation contiendra plutôt des noms communs ou propres car les gens renseignent leurs écoles, leurs diplômes etc.

Nous avons présenté, au cours de ce chapitre, un tour d'horizon des différentes approches de la littérature afin de résoudre les problématiques auxquelles nous sommes confrontés. Nous nous sommes inspirés d'une partie de celles-ci afin de développer les différentes parties du projet « Smart HR » comme nous le verrons dans les chapitres suivants où nous décrirons les différentes techniques utilisées.

CHAPITRE 2

# MÉTHODES

Les travaux mentionnés dans le chapitre précédent montrent bien les difficultés liées au traitement des documents non-structurés. Notre projet se divise donc en deux grandes parties :

1. La première vise à structurer les données provenant des CVs des candidats. Ceci signifie de pouvoir extraire et séparer automatiquement et intelligemment leurs sections les plus pertinentes.
2. La deuxième consiste à proposer un classement des offres d'emploi les plus pertinentes vis-à-vis du profil d'un candidat intéressé par l'entreprise Expleo.

## 2.1 Structuration des CVs

Chaque CV est unique, il n'existe pas véritablement de règles quant à sa structure ni aux informations à y faire figurer. Analyser un CV, est donc identifier automatiquement la limite de ses sections. L'objectif est d'extraire les informations pertinentes dans chaque section identifiée.

Nous présentons dans cette section les deux méthodes utilisées pour la structuration des CVs.

### Méthodes symboliques

L'approche symbolique consiste à définir des règles manuelles permettant l'extraction des différentes sections des CVs. Selon (10), un système symbolique repose sur deux éléments : ressources et règles. Les ressources sont des listes de termes (par exemple une liste de mots-clés) ou tout moyen externe qui ajouteraient des informations supplémentaires au corpus. Les règles, quant à elles, sont principalement des expressions régulières (voir figure 2.1). Les expressions régulières, ou regex (contraction de *regular expression*), sont des outils de recherche de texte dans un document, dans le but détecter, remplacer, extraire, ou supprimer.

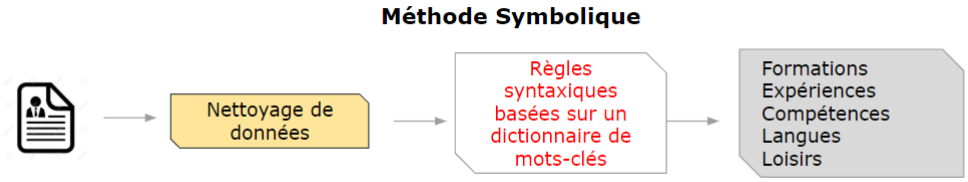



FIGURE 2.1 – Schéma de la méthode symbolique

Après l'observation de 30 CVs issus de différentes sources (sites d'emploi et base de données Expleo), nous avons construit une liste contenant 86 mots-clés. La taille de cette liste a vite accrue car une section peut être représentée par différents mots clés. Par exemple, la section « Formation » peut être définie par les termes suivants : « Formation », « Diplômes », « Cursus universitaire », « Éducation », « Formations et Diplômes » etc. Notons que cette liste est amenée à croître rapidement afin de couvrir tous les mots clés possibles.

Une fois ces mots clés définis, nous avons mis en place un programme informatique qui prend en entrée le texte issu d'un CV et ensuite le parcourt en repérant les mots clés et leurs emplacements. Par exemple, si le programme identifie le mot "formation", il va récupérer le texte qui suit et va s'arrêter lorsqu'il va rencontrer un mot clé définissant une section autre que formation.

Pour aller plus loin, nous avons établi d'autres règles pour extraire quelques informations qui correspondent aux éléments sur lesquels se basent les recruteurs d'Expleo dans leur processus de recrutement, à savoir :

— le nombre d'années d'expérience,
— le code postal,
— l'adresse mail,
— le numéro de téléphone.

Pour finir, cette méthode est propre aux CVs ayant une structure linéaire tels que les CVs LinkedIn (voir figure 2.2), Monster ou encore Indeed, car ils respectent une structure bien précise qui permet facilement l'identification des mots clés des sections et donc l'extraction du contenu de chaque section.

La croissance du nombre d'utilisateurs des sites d'emplois tels que LinkedIn, Indeed ou Monster, qui sont devenus des plate-formes très utilisées dans le cadre des recherches d'emplois, a favorisé la mise en place d'un espace commun à tous les candidats qui leur permet de mettre en ligne leurs CVs tout en respectant une structure prédéfinie (voir figure 2.2). De même, les recruteurs disposent d'un espace dédié à la publication des offres d'emplois avec une structure d'offre à respecter. Ces sites d'emplois proposent d'autres fonctionnalité, par exemple, les recruteurs peuvent également renseigner des critères de sélection afin de faire un premier filtre de candidats, etc...

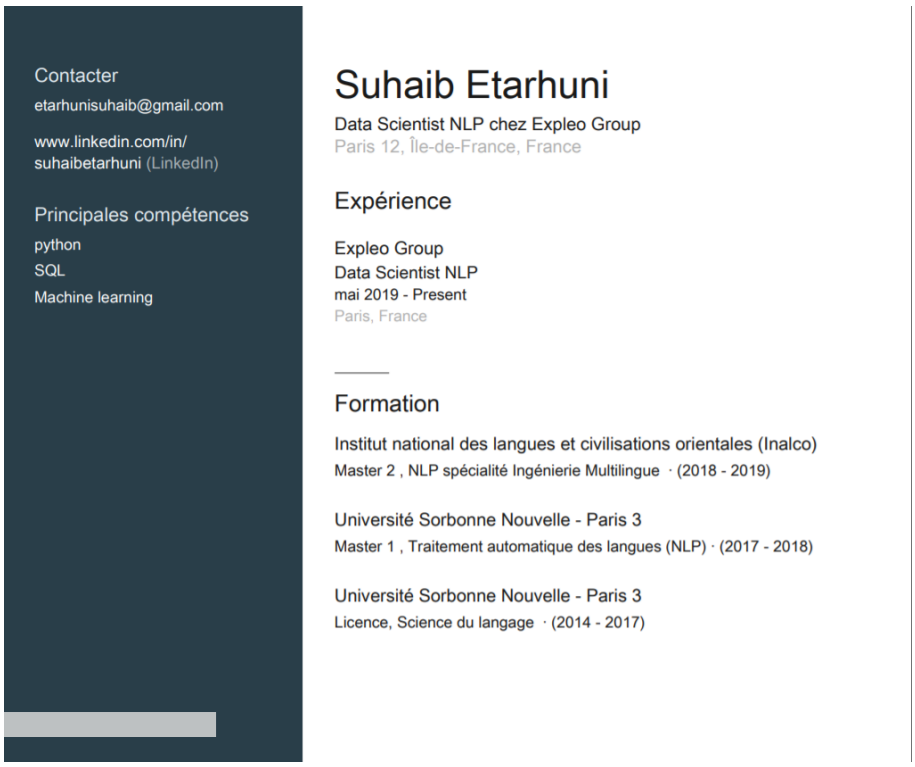



FIGURE 2.2 – Exemple d'un CV Linkedin

Cependant, il est difficile d'appliquer cette méthode symbolique sur des CVs avec des formats quelconques (HTML, Images, etc.) pour deux raisons principales :

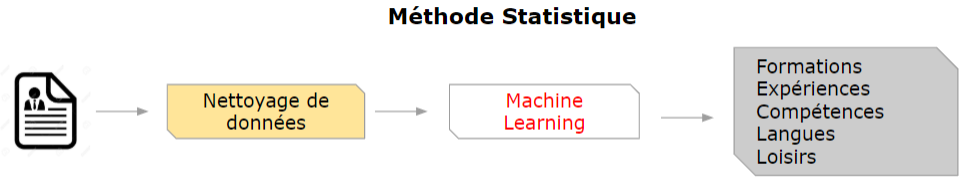
1. La liste des mots clés que nous avons mise en place n'est pas exhaustive.
2. Certains CVs peuvent contenir des tableaux, des colonnes, des images etc, ce qui risque de désorganiser/mélanger les différentes parties du CV après l'extraction du texte brut.

Dans la sous-section suivante, nous allons présenter une méthode plus robuste basée sur des modèles de *Machine Learning*.

### Méthodes statistiques

L'approche statistique, contrairement à l'approche symbolique, se base sur de l'apprentissage automatique (en anglais *Machine Learning*), littéralement « apprentissage machine ». Il s'agit d'une technologie qui s'appuie sur des modèles mathématiques et statistiques traduits informatiquement à travers des algorithmes capables d'analyser les données, (le texte extrait des CVs dans notre cas) de façon à pouvoir réaliser un certain nombre d'opérations complexes. Contrairement à la méthode symbolique, l'objectif de cette méthode est de traiter tous les types de CVs en se basent sur leur texte et non pas sur la structure. Nous présentons les algorithmes utiles pour notre application par la suite.

FIGURE 2.3 – Schéma de la méthode statistique




Pour identifier et séparer les différentes parties d'un CV, quelle que soit sa structure, nous allons pour chaque phrase extraite du CV, lui associer une catégorie, à l'aide d'un modèle de *Machine Learning* comme présenté dans la figure 2.3. Une classe est représentée par la section à laquelle la phrase appartient. Ceci nous amène à définir une problématique de classification. La classification est une des tâches centrales dans la fouille de données. Elle peut se définir comme étant l'association d'une catégorie factuel unique à une donnée, à choisir parmi un ensemble fini de réponses possibles (Tellier).

Il existe deux approches possibles : supervisée et non supervisée. La première consiste à entraîner un modèle mathématique sur des phrases annotées avec des classes, pour ensuite prédire la classe d'une nouvelle phrase. Pour la seconde approche, il n'est pas question de s'appuyer sur des éléments prédéfinis. La tâche revient à la machine de procéder toute seule à la catégorisation des phrases. Notre démarche s'inscrit dans la classification supervisée des phrases extraites des CVs et nécessite donc une construction des données d'apprentissage qui sera détaillée dans le prochain paragraphe.

### Préparation des données

Comme nous l'avons expliqué précédemment, la première phase de notre projet est la segmentation du CV. Cette dernière vise à découper le CV en cinq sections principales (Formation, Expérience, Compétences, Langues et Loisirs) pour non seulement avoir un format structuré facile à traiter mais aussi ne conserver que les informations les plus pertinentes. Pour cela, notre démarche vise à extraire les phrases du CV et réussir à leur attribuer une classe, à savoir, une section parmi les cinq sections définies. Pour ce faire, nous allons développer un modèle mathématique capable de prendre en entrée les phrases extraites du CV et ensuite prédire leurs classes. Ceci nous amène donc à une problématique de classification supervisée (introduite dans la section précédente) qui nécessite de construire les données d'apprentissage qui vont servir à l'entraînement de nos modèles de *Machine Learning*.

Notre corpus d'apprentissage détaillé dans le chapitre 3, contient pour chaque classe (section) à repérer un certains nombres d'exemples afin que

le modèle soit le plus robuste possible. Ces exemples sont des phrases de CVs. Autrement dit, notre modèle apprend sur des phrases de CVs. Dans un premier temps nous extrayons le texte du CV. Dans un second temps nous segmentons le texte récupéré en phrases, qui constitueront les entrées (Input) de notre modèle.

### Pré-traitements

Les phrases récupérées sont au format brut. Nous avons décidé de les pré traiter, c'est-à-dire les nettoyer du plusieurs manières :

— Élimination des *stop-words* : ce sont les mots vides tels que les mots grammaticaux, les auxiliaires, les verbes supports, les ponctuations et les caractères spéciaux.
Exemples : **le**, **à**, **pour**, **comment**, **avoir**, **sont**, **sans**, **trop**...

— Lemmatisation : associer à chaque mot la forme canonique qui le représente dans un dictionnaire. Il s'agit de l'infinitif pour les verbes, la forme masculin singulier pour les noms et les adjectifs. Exemple : **mange, mangera, mangeraient** -> **manger**.

— Suppression des accents, icônes et transformation en minuscule.

Voici un exemple pour illustrer les étapes pré-citées sur phrase extraite d'un CV :

*Participation aux différents projets d'analyse de données utilisant ➤ Python, ➤ R*

Après nettoyage, nous obtenons :

*participation different projet analyse donnee utiliser python r*

Après le nettoyage des phrases extraites des CVs, il est nécessaire de les représenter sous forme de vecteurs de nombres réels car il n'est pas possible de donner du texte directement en entrée d'un modèle mathématique. Cette représentation a pour but d'encoder le sens sémantique des mots et d'appliquer les algorithmes d'apprentissage automatique. Ceci est l'objet de la section suivante.

### Représentation Vectorielle

Les données textuelles nécessitent quelques étapes supplémentaires afin qu'elles puissent être exploitables par l'apprentissage automatique. En effet, il est nécessaire de trouver une représentation mathématique de ces données pour pouvoir les utiliser.

Cette représentation mathématique du texte est connue sous le nom de « Plongement de mots » ou « Words Embeddings ». Cette technique est basée sur la théorie linguistique « Sémantique distributionnelle » de Zellig Harris en 1954 [1]. Cette théorie a pour particularité que les mots apparaissant dans des contextes similaires possèdent des sens proches.

La représentation vectorielle a pour objectif d'analyser le sens des mots car elle permet de représenter chaque mot par un vecteur de nombres réels correspondant. Par conséquent, les mots apparaissant dans des contextes similaires ont des vecteurs relativement proches.

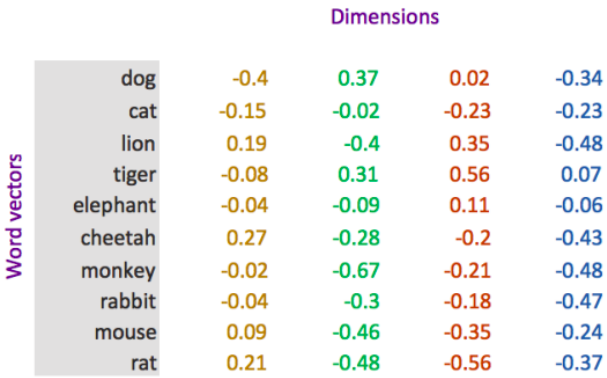



FIGURE 2.4 – Représentation vectorielle des mots

Par exemple, on pourrait s'attendre à ce que les mots « chien » et « chat » soient représentés par des vecteurs relativement proches dans l'espace vectoriel car ils représentent une relation sémantique commune (Animaux). En revanche les mots « chien » et « voiture » seront relativement distants dans l'espace vectoriel où ces vecteurs sont projetés.

L'information textuelle (les phrases des CVs dans notre cas) est donc représentée sous forme d'une matrice, DM (Document Matrix), de vecteurs de nombres réels. Chaque ligne de la matrice représente une phrase, et chaque colonne représente soit un mot, soit un sens porté par plusieurs mots :

$$DM = \begin{array}{c} \\ s_1 \\ \vdots \\ s_t \end{array} \begin{array}{c} \begin{array}{ccc} t_1 & \cdots & t_n \end{array} \\ \begin{pmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{t1} & \cdots & w_{tn} \end{pmatrix} \end{array}$$

Avec :

- $s_i$ le document i. Dans notre cas c'est une phrase.
- $t_j$ le vecteur j. Il peut être un mot, une combinaison de mots.

1. Zellig Sabbetai Harris est un linguiste américain. Il est connu pour ses travaux sur la linguistique structuraliste et l'analyse du discours.

- $w_{ij}$ le poids de $t_j$ dans la phrase $s_i$.

Les poids $w_{ij}$ peuvent être calculés en utilisant différentes représentations décrites dans la suite :

### La représentation de comptage creuse

Elle repose sur trois quantités :

- $TF(i,j) = n_{ij}$, calcule la fréquence d'apparition du mot dans un document.
- $IDF(i) = \log\left(\frac{|S|}{|\{s_j : t_i \in s_j\}|}\right)$, mesure l'inverse de la fréquence d'un mot dans l'ensemble des documents. En d'autres termes, elle mesure la rareté de l'attribut $t_j$ dans la collection. Notons que $|S|$ est le nombre total de documents (phrases dans notre cas) dans le corpus et $|\{s_j : t_i \in s_j\}|$ est le nombre de documents où le terme $t_i$ apparaît (c'est-à-dire $n_{i,j} \neq 0$).
- $w_{ij} = TF(i,j) \times IDF(i)$ appelée produit TF-IDF, quantité souvent utilisée dans les projets de Text Mining.

Ici, les $w_{ij}$ ne visent à mesurer que l'importance d'un mot $t_j$ dans un document (phrase dans notre cas) $s_i$. Or, la représentation d'un texte dépend de l'ensemble de la collection dont il fait partie et de son contexte, pas uniquement de son seul contenu. Pour cela, nous allons présenter les modèles de Words Embeddings avec deux approches : contextuelle et non contextuelle.

### Modèles de Words Embeddings

Le domaine de Traitement automatique des langues ne cesse d'évoluer, notamment sur l'aspect de la transformation vectorielle du texte. Les modèles de *Words Embeddings* sont des modèles mathématiques basés généralement sur des algorithmes de Deep Learning [2]. Chaque modèle a sa propre architecture qui lui permet de transformer le texte en vecteurs. Nous pouvons distinguer deux modèles d'embeddings :

2. Le Deep Learning (Apprentissage profond) est un ensemble de méthodes d'apprentissage automatique s'appuyant sur un réseau de neurones artificiels s'inspirant du cerveau humain.

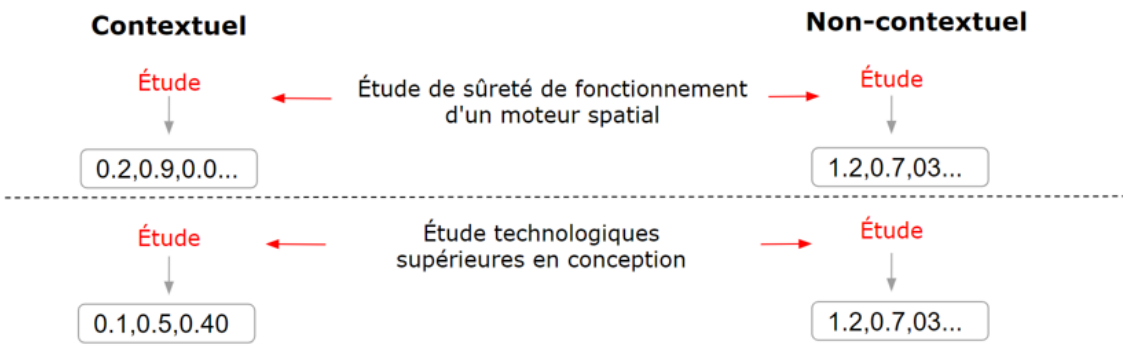



FIGURE 2.5 – Les modèles contextuels et non-contextuels

Deux exemples sont présentés sur la figure 2.5 ci-dessus. Nous pouvons remarquer que le mot *Étude* possède deux sens différents : dans la première phrase, il fait référence à l'analyse d'un sujet particulier, tandis que dans la deuxième, il fait référence à une formation supérieure. Dans ce sens, l'approche contextuelle va prendre ceci en considération et va donc générer deux vecteurs différents pour le mot *Étude*, contrairement à l'approche non contextuelle qui va générer un seul et même vecteur de ce mot quel que soit son contexte.

Il existe plusieurs modèles de « Words Embeddings ». Nous pouvons citer les modèles suivants :

| Version contextuelle | Version non contextuelle |
|---|---|
| Flair (Zalando Research 2018) | Word2vec (Google 2013-2014) |
| Bert (Google 2018) | Glove (Stanford 2015) |
| Elmo (Google 2018) | Fasttext (Facebook 2017) |

Nous décrivons deux modèles d'embeddings : Word2Vec et Flair.

— **Word2Vec**

Word2Vec est un réseau de neurones permettant de trouver à partir d'un mot, ceux qui lui sont le plus similaires en terme de sens. Il prend en argument un grand corpus de documents (phrases) et produit un espace vectoriel d'une centaine de dimensions. La représentation Word2Vec prend en compte le contexte dans lequel le mot a été trouvé lors de l'entraînement, c'est-à-dire les mots avec lesquels il est souvent utilisé et lui génère un unique vecteur. Ce qui est intéressant, c'est que ce contexte permet de créer un espace qui rapproche plusieurs mots qui ne sont pas forcément à côté les uns des autres dans un corpus.

— **Flair**

La représentation Flair prend en compte le contexte dans lequel le mot a été trouvé et lui génère autant de vecteurs que de contextes différents. Cette méthode est implémentée sous la forme d'une librairie Python qui a été développée en 2018. Elle propose de nombreuses fonctionnalités de Traitement automatique des langues comme la reconnaissance d'entités nommées, l'annotation morpho-syntaxique et le plongement lexical. Pour notre projet, nous nous sommes intéressés principalement aux modèles de plongement de mots présentés dans l'article « *Contextual String Embeddings for Sequence Labeling* » proposée par (1) (voir figure 2.6). Ces modèles ont l'avantage d'être des modèles permettant de prendre en compte le contexte du terme en proposant deux modèles appelés « Forward » et « Backward » (voir la figure 2.6). Autrement dit, pour un terme donné, ces modèles vont analyser les mots qui le précèdent ainsi que les mots qui le succèdent et proposeront deux vecteurs de la même taille qui seront combinés à l'aide d'une moyenne.

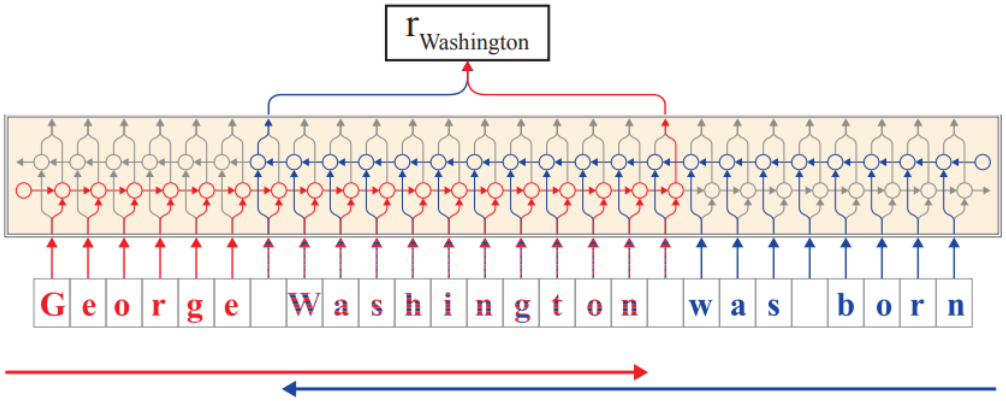



FIGURE 2.6 – Contextual Embeddings of "Washington" (Akbik et al., 2018)

— **Stacked Embeddings**

Nous pouvons également combiner plusieurs modèles d'embeddings. Ceci est connu sous l'appellation *Stacked Embeddings* ou encore *les embeddings empilés*. La libraire Flair de Python permet également cette combinaison.

Une fois les phrases transformées en vecteurs, nous pouvons appliquer des modèles de *Machine Learning* afin de répondre à notre premier objectif à savoir la segmentation des CVs.

### Modèles de classification

Nous souhaitons segmenter les CVs. Nous allons, pour chaque phrase extraite d'un CV, lui attribuer une classe à l'aide d'un modèle de *Machine Learning*. La classe représentera la section auquelle la phrase appartient : Expériences, Formation, Compétences, Langues, Loisirs. Nous traitons donc un problème dit *Multi-Class Classification*. L'objectif est de déduire une règle de

décision qui attribue correctement à toute nouvelle observation, i.e. à toute nouvelle phrase provenant d'un nouveau CV, sa classe d'appartenance.

Tout au long du projet, nous avons testé plusieurs algorithmes d'apprentissage automatique qui s'adaptent à notre problématique.

Dans ce qui suit, nous présentons le fonctionnement des trois modèles que nous avons expérimenté à savoir le modèle SVM, la régression logistique et le modèle Xgboost.

### SVM (Support Vector Machines)

Les Support Vector Machines sont des modèles permettant de réaliser des tâches de classification ou de régression. Ils sont basés sur la recherche de l'hyperplan de marge optimale qui, lorsque c'est possible, classe ou sépare correctement les données tout en étant le plus éloigné possible de toutes les observations. Le principe est donc de trouver un classificateur dont la qualité de prévision est la plus grande possible.

Dans le cadre d'un problème de classification binaire, supposons que les deux classes à prédire sont séparables à l'aide d'un classificateur linéaire dit hyperplan :

- Il existe en général une infinité de classificateurs
- Le meilleur classificateur est celui qui maximise la distance **marge** aux plus proches exemples des deux classes. Il est appelé **classificateur de marge maximale**.

L'hyperplan séparateur optimal est celui qui maximise la marge. Comme nous cherchons à maximiser cette marge, nous parlerons donc de séparateurs à vaste marge (voir figure 2.7).

Dans la présentation des principes de fonctionnements nous schématiserons les données par des « points » dans un plan.

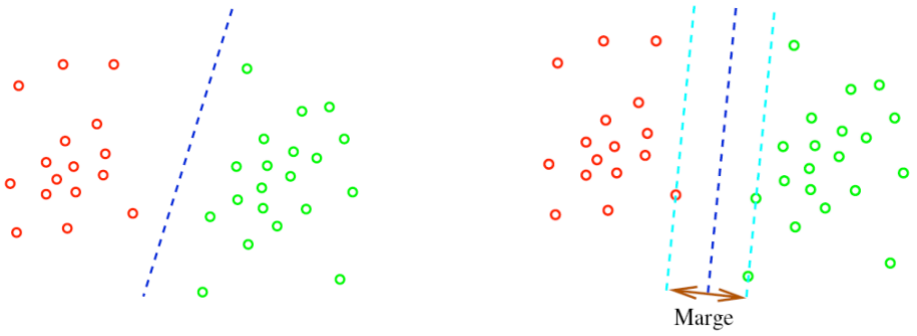



FIGURE 2.7 – Exemple de séparation des deux classes à l'aide d'une fonction linéaire : droite

Les points les plus proches, qui sont utilisés pour la détermination de l'hyperplan, sont appelés vecteurs de support (voir figure 2.8).

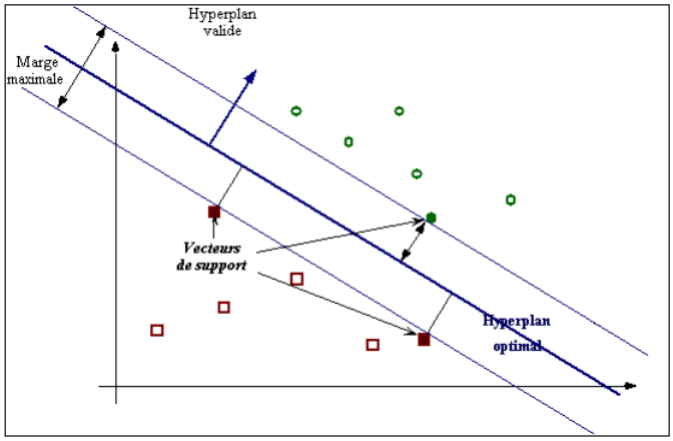



FIGURE 2.8 – Vecteurs de support et marge

L'hyperplan optimal est celui pour lequel la distance aux points les plus proches (marge) est maximale.

Dans un modèle linéaire (cf. figure 2.8), nous cherchons une fonction f telle que $f(x) = w.x + b$. L'hyperplan séparateur (frontière de décision) a donc pour équation $w.x + b = 0$.

La distance d'un point au plan est donnée par $d(x) = \frac{|w.x+b|}{||w||}$. L'hyperplan optimal est celui pour lequel la distance aux points les plus proches (marge) est maximale. Soient $x_1$ et $x_2$ deux points de classes différentes ($f(x_1) = i$ et $f(x_2) = j$), si nous transformons *i* et *j* en deux facteurs $+1$ et $-1$ alors :
$(w.x_1) + b = +1$ et $(w.x_2) + b = -1$ donc $(w.(x_1 - x_2)) = 2$

D'où : $\frac{w(x_1-x_2)}{||w||} = \frac{2}{||w||}$.

Nous pouvons donc en déduire que maximiser la marge revient à minimiser $||w||$. Ceci nous amène à des problèmes d'optimisation que nous résumons en deux paramètres que l'utilisateur doit fixer : la « pénalité » ou la « fonction du noyau ».

La pénalité est une constante qui permet de contrôler le compromis entre nombre d'erreurs de classement et la largeur de la marge. Elle doit être choisie par l'utilisateur, en général par une recherche exhaustive dans l'espace des paramètres. En ce qui concerne la fonction noyau (ou kernel), c'est une fonction à définir également par l'utilisateur, en fonction des points si sont séparables linéairement ou non linéairement. L'exemple le plus simple de fonction noyau est le noyau linéaire.

### La régression logistique

La régression logistique est une technique prédictive qui permet de modéliser l'effet de variables $(X_1, ..., X_n)$ sur une variable réponse *Y*. C'est un modèle supervisé de classification. La variable *Y* peut être binaire (i.e à deux modalités ou classes) comme elle peut être à plusieurs modalités. Nous pouvons

donc distinguer deux types de régression logistique : la régression logistique binaire et la régression logistique pour la classification multi-classes.

Le modèle de régression logistique fait l'hypothèse que (modulo la transformation logit) la probabilité $p(x) = P(Y = i|X = x)$ est linéaire (avec $i$ la classe d'intérêt) :

$$\text{logit}(p(x)) = \log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 x_1 + ... + \beta_p x_n = x_i'\beta$$

Ici $x_i'$ est la transposée du vecteur $x_i$ et $\forall x \in ]0;1[$ la fonction logit se définit comme ceci : $\text{logit}(x) = \frac{x}{1-x}$ .

Les paramètres inconnus $\beta_1, ..., \beta_n$ sont estimés en maximisant la *log* vraisemblance :

$$\ln(\beta) = \sum_i^n \left(y_i x_i'\beta - \log(1 + \exp(x_i'\beta))\right)$$

La régression logistique transforme sa sortie à l'aide de la fonction sigmoïde logistique pour retourner une valeur de probabilité qui peut ensuite être affectée à deux ou plusieurs classes discrètes.

Dans le cas où nous avons une classification binaire, nous allons parler de la fonction Sigmoïde. Cette fonction est une courbe en forme de S qui peut prendre n'importe quel nombre réel et la transformer en une valeur comprise entre 0 et 1 (voir figure 2.9). En règle générale, si la valeur obtenue après la transformation est supérieure à une valeur seuil, nous lui attribuons l'étiquette 1, sinon nous lui attribuons l'étiquette 0.

$$sigmoïde(x) = \frac{1}{1 + exp(-x)}$$

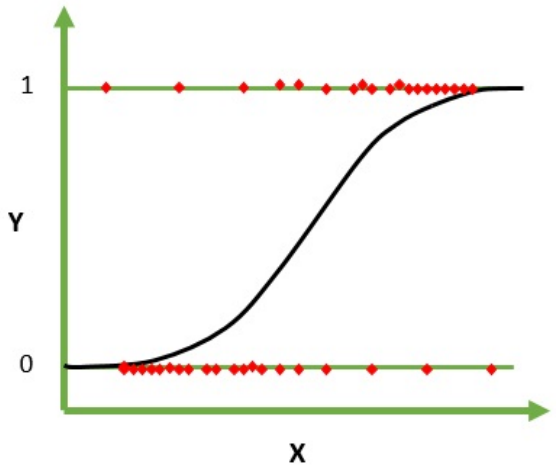



FIGURE 2.9 – Exemple de la régression logistique binaire

Si la régression logistique binaire utilise la fonction Sigmoid, la régression multinomiale (multi-classes) utilise la fonction Softmax. La différence entre ces deux fonctions réside dans le fait que la fonction Softmax se base sur un calcul de probabilité plus complexe d'appartenance dans chaque classe sur toutes les classes cibles possibles. Autrement dit, Softmax permet d'avoir une distribution de probabilités pour toutes les classes. classe retenue sera la classe avec la probabilité la plus haute. La somme des probabilités de la fonction Softmax est égale à 1.

$$softmax(x)_i = \frac{exp(x_i)}{\sum_j exp(x_j))}$$

**XGBOOST**

eXtreme Gradient Boosting (Le Boosting de Gradient) a été développé dans le cadre d'un projet de recherche à l'Université de Washington. C'est un algorithme d'apprentissage supervisé dont le principe et de combiner les résultats d'un ensemble de modèles plus simples et plus faibles afin de fournir de meilleurs résultats en termes de prédictions. Cette combinaison présentée dans la figure 2.10 est appelée méthode d'agrégation de modèles. Le Boosting de gradient peut se définir à l'aide de deux notions :

— Le Boosting : technique qui consiste à agréger des modèles élaborés séquentiellement sur un échantillon d'apprentissage dont les poids des données d'entrées (les individus) sont corrigés au fur et à mesure pour obtenir les meilleures prédictions possibles.

— La descente de gradient : technique itérative qui construit les modèles en recherchant leurs paramètres à l'aide d'une optimisation d'une fonction objectif qui est, dans le cas de l'apprentissage automatique supervisé, la minimisation de l'erreur de prédiction (différence entre la valeur réelle et sa valeur prédite par le modèle).

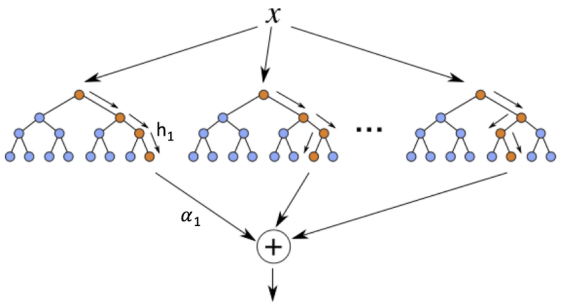



FIGURE 2.10 – Exemple d'agrégation des modèles

Nous avons exploré principalement ces trois modèles afin d'associer chaque phrase avec la section qui lui appartient. Les résultats sont présentés dans le chapitre 4.

## 2.2 Classement des offres d'emplois

Dans la section précédente, nous avons introduit la phase de structuration des CVs des candidats. En effet la segmentation des CVs permet de récupérer leurs sections afin de pouvoir les traiter séparément et les utiliser comme données d'entrées de modèles d'apprentissage automatique. Ceci constitue la deuxième phase de notre travail. Elle consiste en une recherche d'offres d'emplois qui pourraient intéresser les candidats. Dans ce sens, nous allons proposer un classement des offres d'emplois d'Expleo qui peuvent correspondre aux profils de ses candidats.
A l'aide du modèle de classification qui sert à segmenter le CV, nous allons récupérer la section décrivant les compétences acquises des candidats et la comparer aux compétences requises renseignées dans l'offre d'emploi.

Il existe plusieurs approches pour classer les offres d'emplois. Dans les travaux de (9), les auteurs considèrent qu'il s'agit d'une tâche de régression et utilisent des modèles d'apprentissage automatique supervisée comme la régression linéaire, SVM etc. Or, comme mentionné dans la section précédente, pour pouvoir appliquer des modèles de classification supervisée, il est nécessaire que les données d'entrée soient labellisées. Ces dernières vont servir à la construction et l'entraînement du modèle afin qu'il puisse, pour une donnée nouvelle, prédire sa classe d'appartenance parmi les classes contenues dans les données d'apprentissage.

Dans notre cas, nous ne disposons pas de données labellisées. Par conséquent, nous avons adopté une approche basée sur un calcul de similarité entre les informations extraites des sections de CV et les informations extraites de l'offre d'emploi.

Le calcul de similarité entre les textes est utilisé dans plusieurs domaines, comme la recherche d'information pour rapprocher la requête (l'information recherchée) avec les documents les plus similaires. La similarité est une mesure permettant de comparer deux objets de même type. En TAL (Traitement Automatique du Language), c'est une métrique qui mesure la similarité ou la dissimilarité entre les textes (15). La similarité peut être calculée sur le plan syntaxique ou le plan sémantique.

La similarité syntaxique consiste à comparer des textes en comparant leurs chaînes de caractères. Prenons par exemple les mots suivants « appartement » et « apparemment », ils peuvent être considérés comme étant proches syntaxiquement. Parmi de telles mesures de similarité, citons par exemple : l'indice de Jaccard, la distance euclidienne et la similarité cosinus.

Selon (15), deux concepts sont considérés comme sémantiquement similaires s'il y a une synonymie, hyponymie [3], antonymie ou troponymie [4] entre eux. La similarité sémantique se base donc sur la ressemblance de leur signification (contenu sémantique).

Pour ce travail, nous avons appliqué à la fois la similarité syntaxique et sémantique afin de proposer au candidat les offres d'emplois qui correspondent à son profil.

Nous avons d'une part toutes les phrases du CV qui sont labellisées comme étant des compétences, et d'autre part nous avons les compétences requises renseignées dans l'offre d'emploi. Notre première approche consiste à utiliser la librairie Spacy [5] qui fournit plusieurs fonctionnalités en traitement du texte. Parmi ces fonctionnalités, Spacy propose un calcul de similarité sémantique basé sur une représentation vectorielle de deux textes. Autrement dit, nous allons transformer nos deux parties de textes en vecteurs (comme dans l'exemple de la figure 2.4). Ensuite nous appliquons la fonction « similarity » qui va générer un score de similarité compris entre 0 et 1 en comparant les vecteurs générés.

En ce qui concerne le calcul de similarité syntaxique, nous avons considéré que la section Compétences du CV ainsi que la section compétences requises de l'offre d'emploi sont deux ensembles au sens mathématique. Ensuite, nous avons calculé l'indice de Jaccard sur ces deux ensembles. L'indice de Jaccard est le rapport entre le cardinal (la taille) de l'intersection des ensembles considérés et le cardinal de l'union des ensembles. Il permet d'évaluer la similarité entre les ensembles. Soit deux ensembles A et B :

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Avec

$$0 \leq J(A, B) \leq 1$$

Plus l'offre d'emploi et le CV ont des compétences en commun, plus la valeur de l'indice de Jaccard sera proche de 1.

Dans ce chapitre, nous avons détaillé les différentes techniques utilisées dans ce projet. Nous avons expliqué les deux principales méthodes utilisées pour le traitement du CV à savoir la méthode symbolique et la méthode statistique. Nous avons également présenté les modèles de plongement de mots

3. Relation entre un mot et un autre, le premier ayant un sens inclus dans celui du second
4. Relation sémantique entre deux verbes, l'un décrivant de manière plus précise l'action de l'autre
5. https://spacy.io

(Words Embedding) les plus répandus dans le domaine de TAL. Enfin, nous avons décrit les techniques permettant la proposition d'un classement d'offres d'emplois pour les candidats d'Expleo.

Nous présenterons, dans la partie Expérimentations, les données utilisées ainsi que les résultats obtenus pour les différentes méthodes décrites auparavant.

# Deuxième partie

# Expérimentations

CHAPITRE 3

# CORPUS ET IMPLÉMENTATION DES MODÈLES

A la différence des corpus de référence utilisés dans certains travaux mentionnés dans le premier chapitre, notre corpus est constitué de 200 CVs et 13 offres d'emplois. Les CVs ont des structures différentes et appartiennent aux candidats comme aux employés d'Expleo.

Nous cherchons à identifier les offres d'emplois qui pourraient intéresser les candidats d'Expleo. Une offre d'emploi est représentée par une « fiche projet ». Cette fiche projet correspond soit à un sujet interne à l'entreprise soit à un sujet client. Expleo a mis en place des fiches pour décrire et capitaliser chacune de ses offres d'emplois. Une fiche projet comporte différents champs d'informations : une description du projet, le ou les secteurs (automobile, ferroviaire...), les compétences requises (Python, Machine Learning...), etc.

Notre travail consiste donc à :

— Construire, à partir des 200 CVs, nos données d'apprentissage pour développer un modèle mathématique capable de segmenter le contenu d'un CV donné en plusieurs sections.
— Récupérer la section compétences après la segmentation des CVs pour calculer la similarité entre les compétences requises des offres d'emplois et les compétences acquises des candidats. Ceci va conduire à une proposition d'un classement des offres d'emplois des plus adaptées aux moins adaptées aux profils des candidats d'Expleo.

Dans ce chapitre, nous allons présenter, dans un premier temps, notre corpus d'apprentissage et comment nous l'avons construit. Ensuite, nous allons détailler les étapes d'implémentation de nos modèles allant du choix du modèle jusqu'à l'optimisation de ses hyper-paramètres. Enfin, nous allons présenter le calcul de similarité retenu pour l'identification des projets/offres d'emploi les plus adaptés aux profils des candidats.

## 3.1 Données d'apprentissage

Nous avons construit notre corpus d'apprentissage manuellement à partir des 200 CVs à notre disposition. Dans un premier temps, nous avons divisé chaque CV en une liste de phrases. Ensuite, nous avons attribué une classe (label) à chaque phrase. Les classes sont les suivantes : Formation, Expérience, Compétences, Langues et Loisirs comme présenté dans la figure 3.1.

| Phrases | Classes |
|---|---|
| 2012-2014 Ingénieur en informatique | Expérience |
| 2010-2012 Master management des organisations | Formation |
| Python, R, Java, Excel | Compétences |
| Voyages, football, musique | Intérêt |
| Anglais, Français | Langues |

TABLE 3.1 – Exemples des phrases labellisées

Notre base de données constituée de 200 CVs nous a permis d'obtenir environ 8 000 phrases. Dans la figure 3.1, nous avons projeté notre corpus d'apprentissage en deux dimensions avec l'outil T-SNE [1] de Scikit-learn. T-SNE (t-distributed stochastic neighbor embedding) est une technique de réduction de dimension pour la visualisation de données. Il s'agit d'une méthode non-linéaire permettant de représenter un ensemble de points d'un espace à grande dimension dans un espace de deux ou trois dimensions. Elle a pour propriété de respecter la proximité des points, c'est-à-dire que les points qui sont proches l'espace d'origine, seront proches dans l'espace réduit. Cette projection permet d'avoir une idée générale sur les données du corpus.

1. https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

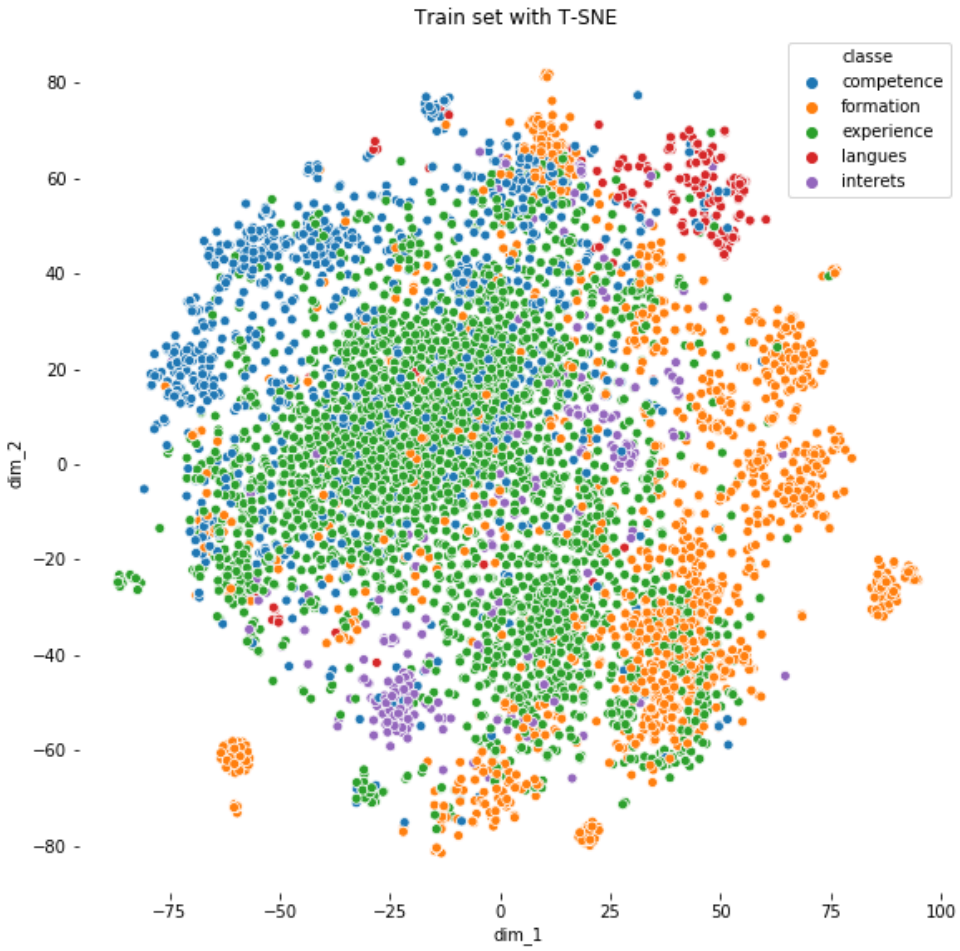



FIGURE 3.1 – Représentation des phrases du corpus d'apprentissage après normalisation en 2D

En s'appuyant les données présentes sur la figure 3.1, on peut constater que les données de la classe loisirs (ici intérêts) sont assez dispersées. Ceci peut être justifié par le fait que les loisirs varient d'un CV à l'autre. Cependant, les données de la classe langues sont assez homogènes car nous avons généralement des langues communes dans les CVs. Ainsi, on remarque que la classe expérience est centrale mais se chevauche avec la classe formation. Ce chevauchement est dû au fait que les phrases dans ces deux classes sont très similaires surtout lorsqu'il s'agit des descriptions des projets professionnels ou académiques.

Nous avons labellisé l'ensemble de notre corpus manuellement, l'avantage de l'annotation manuelle est sa fiabilité sur des petits corpus. Cependant, ce processus est très coûteux en terme de temps. Pour ce travail, nous avons

adopté une approche simple et basique, nous avons labellisé les phrases selon leur section dans le CV. Autrement dit, nous n'introduisons aucune subjectivité lors de l'annotation, nous considérons que le candidat a lui même annoté les informations dans son CV.

| Classe | Phrases |
|---|---|
| Expérience | 4 227 |
| Formation | 2 516 |
| Compétences | 1 626 |
| Loisirs | 491 |
| Langues | 403 |

TABLE 3.2 – La répartition de phrases dans chaque classe

D'après la table 3.2, nous pouvons remarquer que les classes sont déséquilibrées. Il s'agit d'un problème fréquent en apprentissage automatique. Les données déséquilibrées peuvent biaiser les résultats du modèle voire les fausser. Le rééquilibrage des classes est un choix qui dépend de la problématique traitée. En effet, dans certains cas, nous pouvons considérer qu'il n'est pas nécessaire de procéder à l'équilibrage des classes. Néanmoins, dans le domaine médical par exemple, si le modèle a pour but de diagnostiquer le cancer chez un patient, il est n'est pas possible de se permettre des erreurs de classifications. En effet classifier un patient comme négatif (n'est pas atteint de cancer) alors qu'il est positif (est atteint de cancer) est beaucoup plus grave car le patient peut perdre la vie suite à une erreur de diagnostic (prédiction).

Il existe plusieurs techniques pour résoudre ce problème. Ce sont des techniques appliquées directement aux données connues sous le nom de réechantillonnage : sur-échantillonnage et sous-échantillonnage 3.2. Les approches de sur-échantillonage ont pour but de rajouter des exemples dans les classes minoritaires. Parmi les méthodes les plus connues dans le cas de sur-échantillonage on cite le SMOTE « *synthetic minority over-sampling technique* » de (5) qui consiste à générer des exemples synthétiques des classes majoritaires en se basant sur les exemples réels. Au contraire, les techniques de sous-échantillonage ont pour but d'éliminer aléatoirement certaines données de la classe majoritaire. Elles sont nommées RUS *Random Under Sampling*.

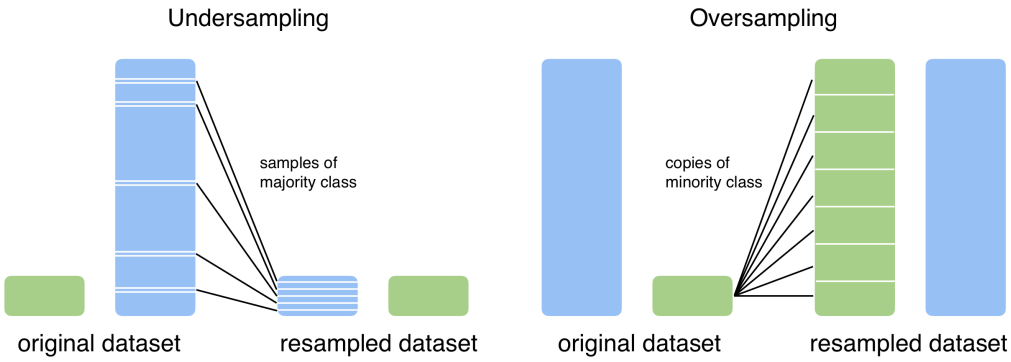



FIGURE 3.2 – Les deux principales méthodes de ré-échantillonnage

Pour notre travail, nous avons fait le choix de ne pas prendre en compte le déséquilibre des classes car nous voulons conserver notre corpus original. Après observation de plusieurs CVs, nous avons constaté que la section expériences professionnelles est toujours la classe dominante en terme de quantité d'informations (de phrases) comme présenté dans la table 3.2.

D'autres techniques existent afin d'ajuster « l'importance » des classes minoritaires. Elles ne sont pas basées sur les données mais plutôt sur des algorithmes d'apprentissage automatique. Une des méthodes consiste à attribuer un poids à chaque classe en fonction du nombre d'exemples observés. Ceci peut être spécifié avec le paramètre «class weight» présent dans certains classifieurs comme le SVM ou la régression logistique.

## 3.2 Modèles

Nous avons introduit dans le chapitre 2 les modèles utilisés dans ce projet. Dans cette section, nous expliquons les différentes étapes d'implémentation de ces modèles.

Avant de choisir le modèle de classification d'apprentissage automatique, il a fallu choisir d'abord une représentation vectorielle. Nous avons testé six représentations vectorielles pour chaque modèle. Dans ce sens, nous avons exploré les deux approches décrites au chapitre 2, à savoir l'approche non-contextuelle en utilisant les modèles Glove et Word2vec (custom), ensuite l'approche contextuelle en utilisant Flair et Bert. Enfin, nous avons exploré la combinaison des deux approches qui se traduit en une combinaison de Flair et Word2Vec et en une autre de Flair et Glove.

Nous avons entraîné notre propre modèle de Word2Vec sur nos phrases de CVs afin d'avoir un modèle adapté au domaine du recrutement. L'implémenta-

tion du modèle de Word2Vec est fait avec la bibliothèque Gensim [2] de Python. En ce qui concerne Glove nous avons pris le modèle proposé par l'université de Stanford [3].

Pour cela et pour des raisons d'optimisation de temps de calcul, nous avons dès le début segmenté notre corpus de phrases en deux échantillons : un échantillon d'apprentissage (*Train*) pour entraîner le modèle de classification, et un deuxième de test pour tester la performance du modèle entraîné et ce de manière définitive. Cette segmentation est réalisée à l'aide de la fonctionnalité « *train_test_split* » fournie par Scikit-Learn dans Python. La taille des deux échantillons est contrôlée par les paramètres « *train_size* » et « *test_size* » que nous avons fixé à 75% pour l'apprentissage et 25% pour le test des données dont nous disposons. Enfin, pour pouvoir comparer la performance des modèles, il est très important de les entraîner sur les mêmes données. Ceci est fait à l'aide du paramètre « random_state » qui nous permet d'avoir les mêmes données dans le corpus d'apprentissage et de test à chaque lancement de la fonction de partitionnement.

### Optimisation des paramètres

Les modèles d'apprentissage automatique ont souvent plusieurs paramètres, appelés hyper-paramètres, qu'il faut définir avant leur entraînement sur les données. La performance des modèles dépend de ces paramètres, il est donc important de bien les choisir ou de définir une méthode afin de les optimiser. Ceci est assez complexe et coûteux en terme de temps de calcul dans le sens où les modèles possèdent souvent plusieurs paramètres à optimiser et il faut donc tester plusieurs combinaisons afin d'atteindre la performance maximale. Pour cela, nous avons utilisé une méthode classique d'optimisation nommée GridSearchCV proposé par Scikit-learn qui permet d'optimiser les paramètres en comparant de manière exhaustive toutes les combinaisons possibles. Dans ce qui suit, nous présentons quelques hyper-paramètres des modèles régression logistique et Xgboost.

Les hyperparamètres les plus importants du modèle logistique sont « C », « solver » et « class_weight ». « C » est un paramètre de régularisation, c'est-à-dire qu'il sert à contrôler le poids des autres paramètres afin d'éviter le sur-apprentissage ou le sous-apprentissage. « Solver » est la méthode d'optimisation. Voici les valeurs que le paramètre « Solver » peut prendre :

- Pour les petits jeux de données, « liblinear » est un bon choix.
- Pour les problèmes multi-classes, seuls « newton-cg », « sag », « saga », et « lbfgs » sont acceptables.

« class_weight », comme son nom l'indique, permet de prendre en compte le poids de chaque classe représenté par le nombre d'observations. Cet hyper-paramètre peut prendre un dictionnaire de valeur où chaque valeur représen-

2. https://pypi.org/project/gensim/
3. https://nlp.stanford.edu/projects/glove/

tera le poids d'une classe, ou « Balanced » (équilibrées) qui permet au modèle de prendre en compte le déséquilibre des classes en attribuant un poids pour chaque selon les données observées.

Xgboost quant à lui, dispose de beaucoup de paramètres car c'est un modèle complexe qui englobe plusieurs petits modèles sous forme d'arbres. Nous en citerons quelques-uns qui concernent l'apprentissage dans chaque arbre comme par exemple « learning_rate, n_estimators, max_depth, subsample ». Le « learning_rate » est un hyperparamètre commun dans la plupart des modèles d'apprentissage qui contrôle le pas d'apprentissage. Une grande valeur exprime un apprentissage rapide. Le « n_estimators » est aussi commun dans la famille des modèles à base d'arbres et représente le nombre d'arbres crés. « max_depth renvoie à la profondeur dans chaque arbre (sous-arbres). Le subsample est le ratio d'échantillons d'apprentissage (*training set*).

Nous nous sommes donc servi de l'outil GridSearchCV [4] afin de trouver les valeurs optimales pour chaque hyper-paramètre.

Afin d'évaluer les différentes valeurs des hyper-paramètres et donc choisir la combinaison optimale pour chaque modèle, nous avons utilisé une autre méthode nommée la validation croisée. Il s'agit d'une technique d'évaluation très courante en apprentissage automatique qui consiste à diviser aléatoirement notre ensemble de données en *N* parties. Pour notre cas, *N=5*. A chaque itération, *N-1* parties de données vont être utilisées comme échantillons d'apprentissage tandis que l'autre partie sera l'échantillon de test. On répète cette opération jusqu'à ce que chacune des N parties ait servi à la fois à l'entraînement et au test (voir figure 3.3).

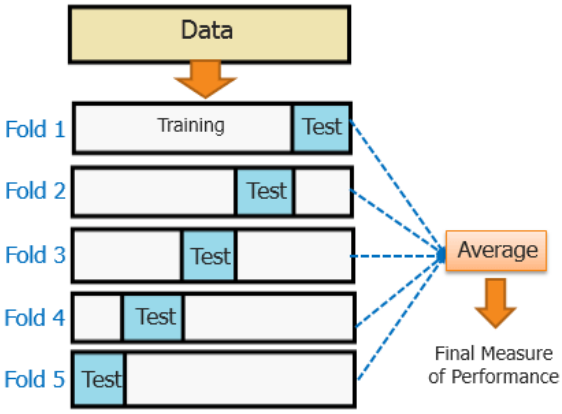



FIGURE 3.3 – Schéma de Validation Croisée

4. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Il est important de signaler que les modèles ensemblistes comme Xgboost sont coûteux en temps de calcul. C'est un modèle séquentiel (se base sur les erreurs du modèle précédent pour ajuster les poids des paramètres) à l'inverse des modèles ensemblistes parallèles qui sont plus rapides.

Dans ce chapitre, nous avons décrit nos données ainsi que les modèles d'apprentissage automatique et la manière d'optimiser les différents paramètres en utilisant GridSearchCV. Le chapitre suivant présentera les résultats obtenus pour chaque modèle avec les différentes transformations vectorielles utilisées.

CHAPITRE 4

# RÉSULTATS

## 4.1 Mesures d'évaluation

En apprentissage automatique, il existe plusieurs méthodes d'évaluation des modèles de classification dont la matrice de confusion. Cette dernière permet d'indiquer si le modèle parvient à classer correctement, c'est-à-dire à quel point les prédictions données par le modèle sont proches des données réelles. La matrice de confusion indique, pour chaque classe, le nombre de prédictions correctes ainsi que le nombre de prédictions incorrectes. Chaque ligne de la table 4.1 correspond à une classe prédite et chaque colonne correspond à une classe réelle.

Voici un exemple d'une matrice de confusion d'un modèle de classification qui a pour but de classer les patients en deux classes (Malade et non malade) en fonction du résultat d'un test médical.

| | Malade | Non malade |
|---|---|---|
| Test Positif | VP (Vrai Positif) | FP (Faux Positif) |
| Test Négatif | FN (Faux Négatif) | VN (Vrai Négatif) |

TABLE 4.1 – Matrice de confusion

Afin de mieux comprendre la matrice de confusion, nous citons la définition de chacun de ces termes :

— VP (Vrai Positif) : la prédiction est positive et la valeur réelle est effectivement positive.
— VN (Vrai Négatif) : la prédiction est négative et la valeur réelle est effectivement négative
— FP (Faux Positif) : la prédiction est positive, mais la valeur réelle est négative
— FN (Faux Négatif) : la prédiction est négative, mais la valeur réelle est positive

Néanmoins, il existe d'autres critères à prendre en compte lors de l'évaluation de modèles de classification. Nous citons :

$$\textit{Exactitude} = \frac{VP + VN}{VP + VN + FP + FN}$$

L'exactitude correspond à la proportion de prédictions correctes effectuées par le modèle.

$$\textit{Précision} = \frac{VP}{VP + FP}$$

La précision indique la proportion du nombre d'exemples correctement classés dans une classe par rapport au nombre total d'exemples classés dans cette classe.

$$\textit{Rappel} = \frac{VP}{VP + FN}$$

Le rappel mesure la proportion du nombre d'exemples correctement classés dans une classe par rapport au nombre total de données appartenant à cette classe.

$$\textit{F-mesure} = 2 \times \frac{\textit{Précision} \times \textit{Rappel}}{\textit{Précision} + \textit{Rappel}}$$

La F-mesure correspond à une moyenne harmonique de la précision et du rappel et est considérée comme un résultat global.

## 4.2 Évaluation de la méthode symbolique

Afin d'évaluer notre méthode symbolique qui se base sur les mots clés pour segmenter le CV, nous avons comparé la sortie de cette méthode (les éléments dans chaque section extraite de chaque CV) et les éléments présents réellement dans les sections du CV. Autrement dit, pour chaque CV, nous observons en amont les informations dans chacune des sections, ensuite nous les comparons avec les informations des sections sorties par la méthode symbolique que nous avons implémentée.

Les CVs donnés en entrée de cette méthode ont différentes structures, c'est-à-dire que nous avons fourni des CVs linéaires comme les CVs LinkedIn, mais aussi des CVs personnalisés avec des structures totalement différentes. Nous avons construit un corpus de 50 CVs dont 20 issus de LinkedIn. Les résultats ont montré que tous les CVs LinkedIn ont été bien structurés. Cependant, seulement 70% des CVs suivant une des structures différentes ont été bien traités. Il est difficile d'évaluer cette méthode car elle dépend de la structure

du CV, dans le cas où les informations des sections se mélangent entre elles lors de l'extraction, il serait difficile d'avoir les bonnes information dans les bonnes sections.

## 4.3 Évaluation des modèles de classification

Nous avons considéré tous les critères d'évaluations que nous avons évoqué au début de ce chapitre afin d'évaluer nos modèles de classification. Nous allons présenter les résultats qui correspondent aux modèles Xgboost et régression logistique. En ce qui concerne le modèle SVM, (voir Annexe 5). En effet, nous avons décidé de l'écarter pour ses faibles performances.

Voici les résultats de ces deux modèles avec les différentes représentations vectorielles. Les résultats sont présentés dans les tableaux 4.2 et 4.3

| Xgboost | | | | |
|---|---|---|---|---|
| Représentation vectorielle | Accuracy | Précision | Rappel | F-mesure |
| TFIDF | 0.78 | 0.79 | 0.79 | 0.79 |
| Word2Vec (custom) | 0.84 | 0.85 | 0.85 | 0.85 |
| Glove | 0.64 | 0.73 | 0.64 | 0.63 |
| Bert | 0.81 | 0.81 | 0.81 | 0.81 |
| Flair | 0.82 | 0.83 | 0.83 | 0.83 |
| Flair + Glove | 0.85 | 0.84 | 0.84 | 0.84 |
| Flair+Word2Vec (custom) | 0.86 | 0.86 | 0.86 | 0.86 |

TABLE 4.2 – Le modèle Xgboost avec les différentes représentations vectorielles

| Régression logistique | | | | |
|---|---|---|---|---|
| Représentation vectorielle | Accuracy | Précision | Rappel | F-mesure |
| TFIDF | 0.79 | 0.80 | 0.80 | 0.80 |
| Word2Vec (custom) | 0.82 | 0.82 | 0.82 | 0.82 |
| Glove | 0.73 | 0.74 | 0.74 | 0.74 |
| Bert | 0.80 | 0.80 | 0.80 | 0.80 |
| Flair | 0.80 | 0.80 | 0.80 | 0.80 |
| Flair + Glove | 0.82 | 0.82 | 0.82 | 0.82 |
| Flair+Word2Vec (custom) | 0.84 | 0.85 | 0.85 | 0.85 |

TABLE 4.3 – La Régression logistique avec les différentes représentations vectorielles

Les différentes expérimentations montrent que la représentation vectorielle et la façon dont le sens des phrases est représenté change la performance du modèle du *Machine Learning*. TF-IDF présentée en détails dans le chapitre 2, est une méthode de pondération répandue principalement dans le domaine de la recherche d'information. Néanmoins, dans l'apprentissage automatique et notamment dans la classification du texte, cette méthode a atteint de très bon résultats. L'avantage est qu'elle génère des caractéristiques (*Features*) permettant au modèle de classer le texte. Pour cela, nous avons utilisé le « TfidfVectorizer » de la librairie Scikit-learn [1] dans Python.

L'inconvénient de cette méthode est que le nombre de caractéristiques générées (égale à 25000 dans notre cas) augmente considérablement en fonction de la taille des textes. Ce qui n'est pas optimale comparé aux autres représentation vectorielles.

1. scikit-learn.org

Certains des autres modèles figurants dans la table 4.3 s'appuient sur des calculs statistiques comme Glove et Word2Vec qui sont en effet des modèles non-contextuels. Autrement dit, une fois qu'un modèle Glove ou Word2Vec est entraîné sur un texte, les vecteurs des termes sont fixés. Si le modèle rencontre un mot qu'il a déjà vu, et peu importe où le mot apparaît dans le nouveau corpus, ce mot aura toujours la même représentation vectorielle défini lors de l'entraînement. Il est important de préciser que notre modèle Word2Vec (custom) est un modèle entraîné sur nos phrases extraites de CVs. Le Word2Vec pré-entraîné proposé par Google est entraîné sur un grand corpus Wikipédia. Néanmoins, il se peut que ce corpus ne contienne pas le vocabulaire des CVs.

D'autre part, les modèles comme Bert et Flair, sont des modèles contextuels, c'est-à-dire qu'ils prennent en compte le contexte du terme même dans un nouveau corpus, le vecteur du terme ne va pas être fixé en amont, il dépendra du contexte. On peut remarquer que les modèles contextuels utilisés sont assez performants, cependant ils n'ont pas dépassé les performances données par le modèle Word2Vec (custom).

La dernière approche utilisée pour la transformation vectorielle de nos phrases est la combinaison de plusieurs modèles de Word Embeddings. Le principal avantage de la librairie Flair de Python est qu'elle offre les modèles de Word Embeddings pré-cités dans le chapitre 2 avec une possibilité de les combiner en se basant sur la moyenne, sur le minimum ou encore sur le maximum des vecteurs générés par les modèles. Nous pouvons constater, au niveau de la table 4.3, que les meilleures performances sont atteintes avec la représentation vectorielle Flair combinée à Word2vec. Cela peut être justifié par le fait nous avons à la fois des phrases qui ne dépendent pas du contexte (les noms d'établissements scolaires par exemple), et des phrases dépendantes du contexte comme les descriptions des projets professionnels ou académiques.

Ainsi, nous avons conservé la représentation Flair combinée avec Word2Vec pour les modèles de classification. Ces modèles ont été comparés par la suite à l'aide d'un rapport de classification, qui permet d'avoir une vision plus détaillée de la performance des modèles et plus précisément la performance à l'intérieur de chaque classe.

Rapport de classification de Xgboost

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| competence | 0.70 | 0.79 | 0.74 | 363 |
| experience | 0.91 | 0.83 | 0.87 | 1126 |
| formation | 0.89 | 0.92 | 0.91 | 610 |
| interets | 0.77 | 0.92 | 0.83 | 107 |
| langues | 0.87 | 0.94 | 0.90 | 110 |
| micro avg | 0.86 | 0.86 | 0.86 | 2316 |
| macro avg | 0.83 | 0.88 | 0.85 | 2316 |
| weighted avg | 0.86 | 0.86 | 0.86 | 2316 |

FIGURE 4.1 – Rapport de classification du modèle Xgboost

Rapport de classification de la Régression logistique

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| competence | 0.73 | 0.79 | 0.76 | 375 |
| experience | 0.89 | 0.83 | 0.86 | 1112 |
| formation | 0.86 | 0.91 | 0.88 | 596 |
| interets | 0.80 | 0.87 | 0.83 | 119 |
| langues | 0.88 | 0.92 | 0.90 | 114 |
| micro avg | 0.85 | 0.85 | 0.85 | 2316 |
| macro avg | 0.83 | 0.86 | 0.85 | 2316 |
| weighted avg | 0.85 | 0.85 | 0.85 | 2316 |

FIGURE 4.2 – Rapport de classification du modèle Régression logistique

Nous constatons que nos deux modèles sont très proches en termes de performances. Or, nous nous sommes rendus compte que, dans notre échantillon test, nous avions un certain nombre de phrases sur lesquelles les deux modèles se trompent. Ces phrases sont des phrases qui perturbent les modèles. Nous les avons analysé manuellement et nous avons constaté qu'il s'agit des phrases difficiles à classer car elle ne portent pas d'informations précises sur leur section. Ces phrases peuvent appartenir à la fois à la classe Formation et à la classe Expérience. Par exemple, les deux phrases suivantes sont extraites de deux CVs différents :

1. *Études de fiabilité et plan de fiabilisation des équipements.*
2. *Études en maintenance mécanique aéronautique.*

Il n'est pas évident de déterminer la classe de ces deux phrases manuellement. En effet, les deux phrases peuvent appartenir aussi bien à la classe Formation qu'à la classe Expérience.

Nous avons considéré les phrases ambiguës que nous avons filtré manuellement comme étant un « bruit ». Vu le nombre de ces phrases (environ 200) et le nombre de phrases dans notre corpus (8 000), nous avons fait le choix

de les éliminer et de recommencer la procédure de segmentation du corpus, représentation vectorielle et entraînement des modèles avec notre nouveau corpus.

L'élimination des phrases perturbantes, a amélioré la performance des deux modèles (voir Figures 4.3 et 4.4). Nos deux modèles passent de 85%, 86% à 89% d'exactitude. Ainsi, on gagne en rappel et en précision dans chaque certaines classes. Pour la classe compétences notre score passe de 79% à 85% du rappel pour le modèle xgboost, et à 83% pour la régression logistique. Pour la classe expériences, le modèle de Xgboost passe de 83% du rappel à 86%. Et 89% de précision à 94%. Le modèle de la régression logistique s'est amélioré en passant de 83% à 87% rappel. Et de 89% à 94% de précision dans la classe expérience.

Rapport de classification de xgboost

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| competence | 0.72 | 0.85 | 0.78 | 296 |
| experience | 0.95 | 0.86 | 0.90 | 1121 |
| formation | 0.91 | 0.95 | 0.93 | 589 |
| interets | 0.74 | 0.88 | 0.81 | 104 |
| langues | 0.89 | 0.93 | 0.91 | 100 |
| micro avg | 0.89 | 0.89 | 0.89 | 2210 |
| macro avg | 0.84 | 0.90 | 0.87 | 2210 |
| weighted avg | 0.89 | 0.89 | 0.89 | 2210 |

FIGURE 4.3 – Rapport de classification du modèle Xgboost sans bruit

Rapport de classification de la Régression logistique

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| competence | 0.75 | 0.83 | 0.79 | 317 |
| experience | 0.94 | 0.87 | 0.90 | 1097 |
| formation | 0.89 | 0.95 | 0.92 | 582 |
| interets | 0.80 | 0.86 | 0.83 | 115 |
| langues | 0.88 | 0.93 | 0.90 | 99 |
| micro avg | 0.89 | 0.89 | 0.89 | 2210 |
| macro avg | 0.85 | 0.89 | 0.87 | 2210 |
| weighted avg | 0.89 | 0.89 | 0.89 | 2210 |

FIGURE 4.4 – Rapport de classification du modèle Régression logistique sans bruit

Nous constatons que les deux modèles ont le même score global de « micro avg ». Ce score décrit le score moyen de la performance du modèle sur l'ensemble des classes. Ce score est très important lorsqu'il s'agit d'une problématique multi-classes.

En revanche, nos modèles diffèrent à l'intérieur de certaines classes. Par exemple, d'après le rapport du modèle Xgboost présenté dans la figure 4.3, ce dernier a une meilleure précision lorsqu'il s'agit de classer les phrases appartenant à Formation. En ce qui concerne le modèle de la régression logistique (figure 4.4), il a une meilleure précision au niveau de la classe Compétences.

Pour atteindre notre objectif, nous nous intéressons principalement aux compétences extraites du CV. Nous avons fait le choix donc d'utiliser le modèle de la régression logistique. Sur l'échantillon test, ce modèle atteint 95% de bonne classification des compétences comme nous pouvons le voir dans la figure 4.5. Autrement dit, parmi les compétences dans l'échantillon test, ce modèle a réussi à identifier 95% de ces compétences. Ce choix est dû aussi au fait que le modèle de la régression logistique est plus simple et plus rapide dans le sens où il a moins d'hyper-paramètres à optimiser que le modèle Xgboost. Ce dernier est beaucoup plus long en temps de calculs et a beaucoup d'hyper-paramètres à optimiser comme nous l'avons expliqué au niveau du chapitre précédent.

Cependant, nous pouvons remarquer que ce modèle (la régression logistique) confond les classes Expérience et Formation. 11% des phrases qui sont issues de la classe Expérience, ont été classées comme étant des phrases de Formation. Ceci est dû au fait que la frontière entre ces deux classes n'est pas toujours facile à identifier car et comme nous l'avons mentionné précédemment ces deux classes utilisent le même vocabulaire surtout lorsqu'il s'agit des projets académiques ou professionnels.

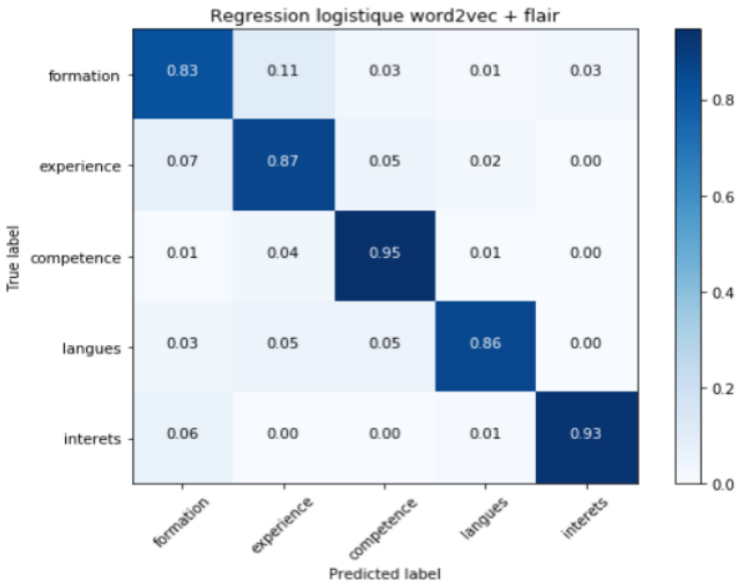



FIGURE 4.5 – Matrice de confusion du modèle régression logistique choisi

## 4.4 Évaluations du classement des offres d'emploi

Le deuxième résultat de notre projet après l'extraction des informations du CV est de réaliser un calcul de similarité (un « Matching ») entre les compétences du candidat extraites du CV et les compétences requises des offres d'emploi d'Expleo. Or, nous ne disposons pas de données labellisées manuellement (CVs et Offres d'emploi correspondants) pour nous permettre de construire des modèles d'apprentissage automatique supervisés. Ainsi, nous avons évalué notre calcul grâce aux membres du pôle Data Science d'Expleo Group. Autrement dit, nous avons collecté un certain nombre des anciennes offres d'emplois du pôle Data Science ainsi que les CVs des personnes qui occupent ces postes actuellement. L'idée est de voir si notre calcul de similarité est performant, c'est à dire que nous sommes capables de proposer les bons postes aux bonnes personnes.

Pour des besoins métier, le calcul de similarité adopté se base sur le calcul de la distance Jaccard entre la section compétences des CVs et la section compétences des offres d'emploi. Pour chaque personne représentée par son CV, notre outil lui a proposé parmi les trois postes les plus adéquats au moins un poste (projet ou mission) sur lequel cette personne travaille réellement. Par exemple, nous avons expérimenté mon propre CV, et le résultat du matching était le projet de mon stage « Smart HR » présenté dans ce mémoire.

Dans ce chapitre nous avons présenté les différents résultats que nous avons obtenu à l'aide de la méthode symbolique et de la méthode statistique. Nous avons détaillé les différents modèles utilisés dans la méthode statistique, notamment les modèles de Words Embeddings. Ces derniers nous ont permis de prouver que la combinaison d'un modèle contextuel comme Flair avec un modèle non-contextuel comme Word2Vec (Custom) améliore la performance des modèles de classification. Parmi ces derniers nous avons comparé le modèle ensembliste Xgboost et le modèle de la régression logistique. Ainsi nous avons expliqué notre choix final en conservant la régression logistique comme modèle d'analyse de CV.

#### Points d'amélioration

Les résultats que nous avons obtenus sont très satisfaisants en termes d'analyse de CV surtout avec le peu de données dont nous disposons. Notre modèle a atteint 89% d'exactitude. Notre méthode pour classer les offres d'emplois (similarité syntaxique et sémantique) en fonction du profil montre également de bons résultats, nous avons au moins un poste dans le top trois qui est réellement en lien avec le profil du candidat.

Toutefois, un des moyens pour améliorer notre système est d'abord de récolter plus de CVs, idéalement des CVs issus de sources variées. Deuxièmement, alimenter les offres d'emploi avec des champs techniques ciblés permettrait d'avoir une meilleure identification des candidats sur l'aspect technique. Nous recommandons aussi d'expérimenter plusieurs combinaisons de modèles de Words Embeddings notamment d'autres modèles contextuels et non-contextuels comme Elmo, Fasttext etc. Nous pouvons aussi essayer de combiner plusieurs modèles de classification via des méthodes comme le Voting Classifier ou le Stacked Classifier.

Enfin, nous recommandons l'implémentation de modèles de classification basés sur le Deep Learning (apprentissage profond) pour la l'analyse de CVs. Les modèles de Deep Learning sont plus complexes mais généralement plus performants.

# Conclusion générale

Le traitement manuel des CVs est un travail fastidieux surtout lorsqu'on dispose d'un flux de CVs important. L'objectif de ce mémoire était de proposer un outil permettant l'optimisation du processus de recrutement en automatisant l'analyse des CVs et la présélection des profils.

Notre outil propose deux fonctionnalités. La première consiste à analyser le CV et extraire les informations pertinentes. La deuxième consiste à proposer un classement des offres d'emplois en fonction du CV.

Pour la première fonctionnalité, nous avons mis en place deux méthodes fonctionnelles pour traiter automatiquement les CVs, à savoir la méthode symbolique et statistique. Nous avons montré les limites de la première qui nécessite une structure figée comme celle de LinkedIn (2.2). En revanche, la méthode statistique a pour but de restructurer intelligemment le CV en cinq sections (Formation, Expérience professionnelle, Compétence, Langues, Loisirs) en se basant sur son texte et non pas sur sa structure. Cette structuration artificielle permet d'extraire les éléments ciblés par les recruteurs. La performance de notre modèle (régression logistique) s'élève à 89% d'exactitude.

Le classement des offres d'emplois se fait en trois étapes. La première étudie la sortie du modèle afin de récupérer les compétences techniques du candidat. Tandis que la deuxième étape consiste à récupérer les compétences requises renseignées dans l'offre d'emploi. L'étape finale se base sur un calcul de similarité syntaxique et sémantique. Le résultat est un classement décroissant des offres d'emplois en fonction du profil du candidat.

Enfin, les modèles adoptés ont été livrés sous forme d'API (Application Programming Interface) en collaboration avec un Data Engineer. L'avantage de cette API est qu'elle peut être utilisée et intégrée facilement par les autres services d'Expleo dont le recrutement.

# BIBLIOGRAPHIE


[1] Akbik, A., Blythe, D., and Vollgraf, R. (2018). Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649. – Cité page 29.

[2] Amdouni, S. and abdessalem Karaa, W. B. (2010). Web-based recruiting. In *ACS/IEEE International Conference on Computer Systems and Applications-AICCSA 2010*, pages 1–7. IEEE. – Cité page 17.

[3] Armstrong, M. (2006). *A handbook of human resource management practice*. Kogan Page Publishers. – Cité pages 13 et 14.

[4] Cabrera Diego, L. A. (2015). *Automatic methods for assisted recruitment*. PhD thesis, Université d'Avignon. – Cité page 14.

[5] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote : synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16 :321–357. – Cité page 42.

[6] Clech, J. and Zighed, D. A. (2003). Data mining et analyse des cv : une expérience et des perspectives. *Revue d'intelligence artificielle*, 17(1-3) :189–200. – Cité page 16.

[7] Diego, L. A. C., Torres-Moreno, J.-M., and El Bèze, M. (2013). Segcv : traitement efficace de cv avec analyse et correction d'erreurs. In *TALN récital*. – Cité page 18.

[8] Dipboye, R. L. and Jackson, S. L. (1999). Interviewer experience and expertise effects. *The employment interview handbook*, pages 259–278. – Cité page 15.

[9] Faliagka, E., Ramantas, K., Tsakalidis, A., and Tzimas, G. (2012). Application of machine learning algorithms to an online recruitment system. In *Proc. International Conference on Internet and Web Applications and Services*. Citeseer. – Cité page 34.

[10] Grouin, C. (2013). *Anonymisation de documents cliniques : performances et limites des méthodes symboliques et par apprentissage statistique*. PhD thesis. – Cité page 21.

[11] Kessler, R., Béchet, N., Torres-Moreno, J.-M., Roche, M., and El Bèze, M. (2009). Profilage de candidatures assisté par relevance feedback. In *TALN'09 : Traitement Automatique des Langues Naturelles*, pages N–A. – Cité page 16.

[12] Kumaran, V. S. and Sankar, A. (2013). Towards an automated system for intelligent screening of candidates for recruitment using ontology mapping (expert). *International Journal of Metadata, Semantics and Ontologies*, 8(1) :56–64. – Cité page 15.

[13] Malherbe, E. (2016). *Standardization of textual data for comprehensive job market analysis*. PhD thesis. – Cité page 14.

[14] Montuschi, P., Gatteschi, V., Lamberti, F., Sanna, A., and Demartini, C. (2013). Job recruitment and job seeking processes : how technology can help. *It professional*, 16(5) :41–49. – Cité page 15.

[15] Negre, E. (2013). Comparaison de textes : quelques approches... – Cité pages 34 et 35.

[16] Pande, S. (2011). E-recruitment creates order out of chaos at sat telecom : system cuts costs and improves efficiency. *Human Resource Management International Digest*, 19(3) :21–23. – Cité page 14.

[17] Peretti, J.-M. (2018). *Gestion des ressources humaines*. Vuibert. – Cité page 13.

[18] Roche, M. and Kodratoff, Y. (2006). Pruning terminology extracted from a specialized corpus for cv ontology acquisition. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 1107–1116. Springer. – Cité page 19.

[19] Singh, A., Rose, C., Visweswariah, K., Chenthamarakshan, V., and Kambhatla, N. (2010). Prospect : a system for screening candidates for recruitment. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 659–668. ACM. – Cité page 15.

[Tellier] Tellier, I. *Introduction 'a la fouille de textes*. université de Paris 3 - Sorbonne Nouvelle. – Cité page 24.

[21] Yahiaoui, L., Boufaida, Z., and Prié, Y. (2006). Automatisation du e-recrutement dans le cadre du web sémantique. – Cité page 16.

[22] Yu, K., Guan, G., and Zhou, M. (2005). Resume information extraction with cascaded hybrid model. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 499–506. Association for Computational Linguistics. – Cité page 17.

[23] Zaroor, A., Maree, M., and Sabha, M. (2017). A hybrid approach to conceptual classification and ranking of resumes and their corresponding job posts. In *International Conference on Intelligent Decision Technologies*, pages 107–119. Springer. – Cité pages 16 et 19.

CHAPITRE 5

# ANNEXE

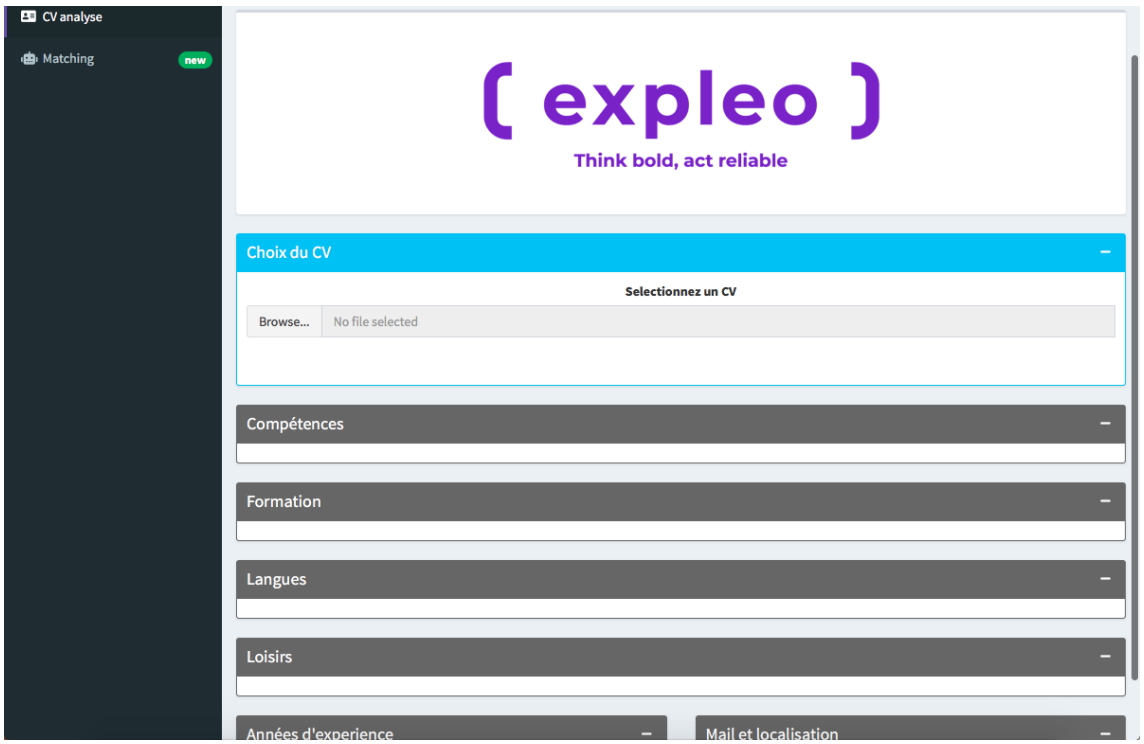



FIGURE 5.1 – Notre application destiné au service recrutement

```
def Model(X,Y,model):

    print("splitter la data en train, test")
    x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.25, shuffle = True)

    print("splitter de Train en train/dev")

    x_train_val, x_val, y_train_val, y_val = train_test_split(x_train, y_train, test_size = 0.25, shuffle = True)


    if model == "SVC_linear":
        pipeline = Pipeline([
        ('tfidf', TfidfVectorizer(analyzer="word")),
        ("linear_svc",LinearSVC())
        ])

        parameters = {
        'tfidf__max_df': (0.25, 0.5, 0.75),
        'tfidf__min_df':range(5,10),
        'tfidf__ngram_range': [(1, 1), (1, 2), (1, 3)],
        'tfidf__max_features': (None, 5000, 10000, 50000),
         "linear_svc__C":[0.01, 10, 0.1],
         'linear_svc__class_weight':["balanced",None]}


    if model == "xgboost":

        pipeline = Pipeline([
            ('tfidf', TfidfVectorizer(analyzer="word")),
            ('xgboost', XGBClassifier())])
        parameters = {
              'xgboost__objective':['binary:logistic'],
              'xgboost__learning_rate': [0.001,10,0.01], #so called `eta` value
              'xgboost__max_depth': [10,500,10],
              'xgboost__n_estimators': [100,1000,100] #number of trees, change it to 1000 for better results
             }

    if model == "reg":
        pipeline = Pipeline([
            ('tfidf', TfidfVectorizer(analyzer="word")),
            ("clf_log", LogisticRegression(multi_class='multinomial',solver='newton-cg'))])
        parameters = {
            'tfidf__ngram_range': [(1, 2), (1, 3)],
            'clf_log__C':np.logspace(-5, 8, 15)}
```

FIGURE 5.2 – optimisation des paramètres des modèles

```
from gensim.models import word2vec

model = word2vec.Word2Vec(data.phrases_lemme.str.split(), size=300, window=5,
                          min_count=1, workers=1, iter=100)


model.corpus_count


model.wv.save_word2vec_format('w2vec_gensim.bin')
```

FIGURE 5.3 – Implémentation du modèle Word2vec custom

## 9.2 FLAIR + Word2Vec

```
]: def transform_wv_mean(word2vec,phrase):
            liste_vect,liste_phrase=[],[]

            phrase=phrase.split()
            for mot in phrase:
                if mot in word2vec.wv.vocab:
                    vecteur=word2vec[mot]
                else:
                    vecteur=np.zeros((300))

                liste_vect.append(vecteur)
            array=np.mean(liste_vect,axis=0)

            return array
    def flair_wv(model_flair,model_wv,X):
        liste_vect,liste_phrases=[],[]
        for phrase in X :
            if phrase!="":
                sentence = Sentence(phrase)
                model_flair.embed(sentence)

                falir_embedding=sentence.get_embedding().detach().numpy()
                embedding_wv=transform_wv_mean(model_wv,phrase)

                flair_wv=np.concatenate((embedding_wv, falir_embedding), axis=0)

                liste_phrases.append(flair_wv)

        return np.vstack(liste_phrases)
```

FIGURE 5.4 – Combinaison des Words Embeddings Flair et Word2vec custom

```
def jaccard_similarity2(list1, list2):
    intersection = len(list(set(list1).intersection(list2)))
    union = len(list(set(list1).union(set(list2))))-intersection
    return float(intersection / union)
```

FIGURE 5.5 – La similarité de Jaccard

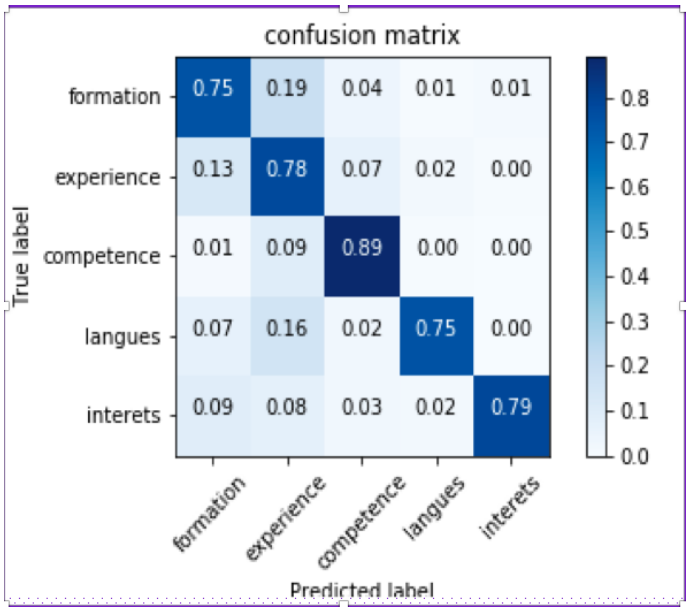



FIGURE 5.6 – Le modèle SVM avec TFIDF

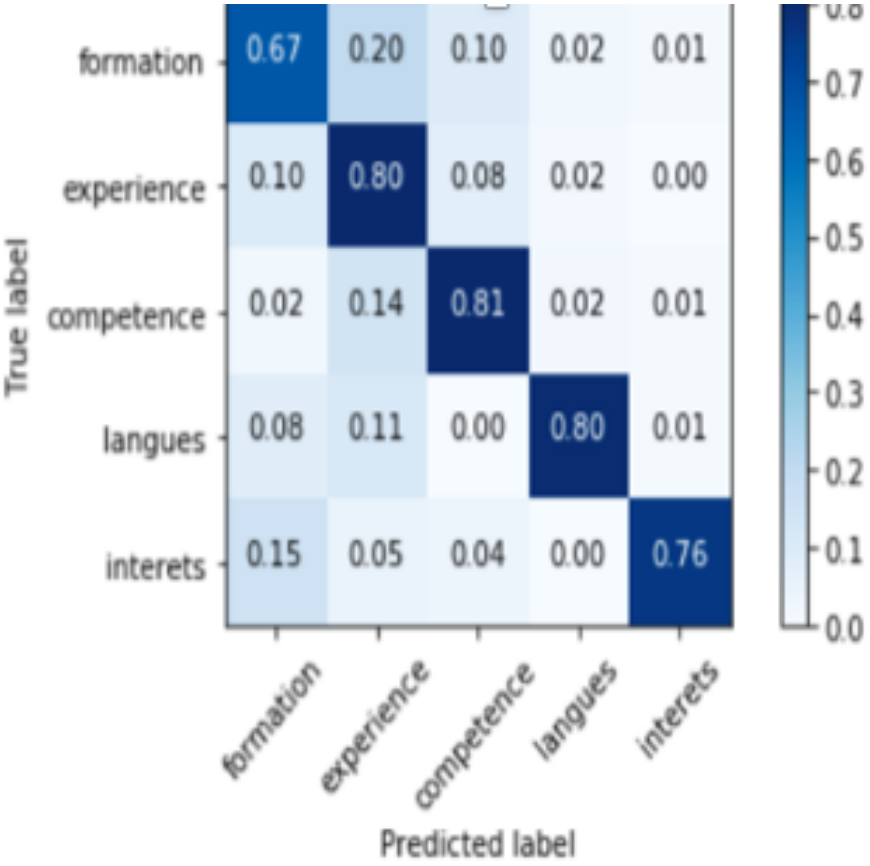



FIGURE 5.7 – Le modèle SVM avec la combinaison Word2Vec et Glove

```python
def pdf_minder_to_txt(fp):
    """

    parser  le pdf avec pdfminder

    """
    rsrcmgr = PDFResourceManager()
    retstr = io.StringIO()
    codec = 'utf-8'
    laparams = LAParams()
    device = TextConverter(rsrcmgr, retstr, codec=codec, laparams=laparams)
    interpreter = PDFPageInterpreter(rsrcmgr, device)

    for page in PDFPage.get_pages(fp):
        interpreter.process_page(page)

    text = retstr.getvalue()

    device.close()
    retstr.close()

    return text
```

FIGURE 5.8 – La fonction qui extrait le texte du CV

```python
@route('/start_matching', method='POST')
def start_matching():
    """
    Starts the matching process for a resume

    :return: HTTP/1.1 200 OK, body: ProjectMatching object
    """
    file = request.files.get('resume').file
    n_matches = request.forms.get('n_matches')

    # TODO: run some checks to sanitize the input

    # Creating the job's unique id
    job_id = str(uuid.uuid4())

    # Adding the new job to the queue
    jobs_queue.put((job_id, file, int(n_matches)))
    builds_pending.append(job_id)

    # Creating the ProjectMatching object
    pm = json.dumps({
        'id': job_id,
        'status': 'DATAMINING_BUILD_PENDING',
        'projectIds': []
    })

    return HTTPResponse(
        status=200,
        body=pm
    )


@route('/get_result/<job_id>', method='GET')
def get_result(job_id='job_id'):
    """
    Returns the result of the specified job id

    :param job_id: The requested job id (str)
    :return: HTTP/1.1 200 OK, body: ProjectMatching object
```

FIGURE 5.9 – La fonction de matching dans l'API