Appariements collaboratifs des offres et demandes d'emploi

Thomas Schmitt

université PARIS-SACLAY

NNT : 2018SACLS210

Thèse de doctorat

UNIVERSITÉ PARIS SUD
Comprendre le monde, construire l'avenir

Appariements collaboratifs des offres et demandes d'emploi

Thèse de doctorat de l'Université Paris-Saclay
préparée à l'Université Paris-Sud

École doctorale n°580 : sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : informatique

Thèse présentée et soutenue à Orsay, le 29/06/2018, par

Thomas Schmitt

Composition du jury :

Yann Chevaleyre Professeur, Paris Dauphine (– LAMSADE)	Rapporteur
Jérémie Mary Maître de Conférences, INRIA Lille (– Sequel)	Rapporteur
Philippe Caillou Maître de conférence, Paris-Sud (– TAU)	Examinateur
Olivier Schwander Maître de conférence, Sorbonne Universités (– LIP6)	Examinateur
Anne Vilnat Professeur, Paris-Sud (– LIMSI)	Présidente
Michèle Sebag Directeur de recherche, Paris-Sud, LRI (– TAU)	Directeur de thèse
Jean-Pierre Nadal Directeur de recherche, CNRS (– LPS)	Invité

~

Remerciements

Je tiens ici à remercier chaleureusement ma directrice de thèse Michèle Sebag, pour ses nombreux conseils pertinents et enrichissants ainsi que pour m'avoir communiqué son goût et son enthousiasme pour l'apprentissage automatique. Je remercie également mon encadrant Philippe Caillou, qui m'a soutenu tout au long de ma thèse.

Merci aux rapporteurs Yann Chevaleyre et Jérémie Mary pour le temps passé sur mon manuscrit et pour les nombreuses questions et suggestions. Je remercie également Jean-Pierre Nadal, Olivier Schwander et Anne Vilnat, qui m'ont fait l'honneur de faire parti de mon jury.

Merci à François Weber, pour ses encouragements précieux durant les derniers mois de rédaction.

Je remercie vivement mes collègues et amis de l'équipe TAU, sans qui cette thèse n'aurait pas été aussi profitable. Nos discussions au déjeuner, au café et dans nos bureaux furent pour moi une ressource inestimable. À ceux qui ont partagé mon bureau, Guillaume pour nos discussions littéraires, Victor pour nos moments de qualités partagés, merci et bon courage à vous. Merci également à Mehdi, pour les challenges en machine learning et tous les conseils en Python, ainsi qu'à Aris pour les challenges et les entraînement de natation. À mon co-auteur, François Gonard, avec qui j'ai pu comprendre en profondeur les systèmes de recommandation. Merci à Diviyan, Olivier et Corentin, à votre équipe qui cartonne et qui me stimule. Merci à Etienne pour le foot, Théo, Victor B. pour le go.

Merci aux anciens également, Nicolas et sa bonne humeur inoubliable, Gaëtan et ses conseils et ses discussions toujours pertinentes. À Marie-Liesse pour ses conseils, à Basile également, merci pour la gentillesse. Merci à Sourava.

Merci aux permanents, Guillaume Charpiat pour sa sympathie, Aurélien Decelle pour son amitié, Cyril Furtlehner, Isabelle Guyon pour sa bonne humeur inaltérable, Flora Jay, Paola Tubaro, Marc Schoenauer, Olivier Teytaud pour sa gentillesse. Je remercie avec une attention particulière Odalric Maillard, pour l'enthousiasme partagé après chaque discussion.

Je remercie également Stéphanie Druetta, secrétaire de l'école doctorale, pour son aide dans mes démarches administratives.

À mes anciens colocataires, Thibault et Charlène, pour ces super années à Orsay, à vivre au soleil et à danser le rock ; merci et bon vent à tout les deux.

Je salue ici Paul, pour la littérature, Alexis aussi, pour les 42 et le rock and roll.

À mes amis de prépa, Adrien, Billel, Julien, Pierre, pour l'élan que nous avons pris ensemble et que nous avons gardé, merci pour votre soutien.

À toute ma famille, pour m'avoir toujours soutenu et encouragé, je vous remercie infiniment.

Enfin et surtout merci à Claire, pour son écoute bienveillante, sa gentillesse et son soutien indéfectible.

Résumé

Notre recherche porte sur la recommandation aux demandeurs d'emploi de nouvelles offres venant d'être postées (recommandation à froid). L'approche proposée étend les systèmes de recommandations développés pour le commerce électronique, en exploitant d'une part les traces d'usage de l'ensemble des demandeurs d'emploi sur les offres antérieures (données collaboratives) et d'autre part les CVs et les documents des offres d'emploi.

Une des spécificités du travail présenté est d'avoir considéré des données du monde réel, pour lesquelles nous sommes reconnaissant à l'agence et plate-forme d'emploi Qapa, et l'Association Bernard Gregory. Ces données nous ont mis en face des défis dus à l'hétérogénéité des secteurs d'emplois (respectivement le secteur de l'intérim pour Qapa et le secteur des PhD en sciences pour l'ABG), et au bruit des documents textuels.

Notre contribution est la conception et l'implémentation d'un modèle de langage, reflétant l'information des données collaboratives et permettant la recommandation à froid d'une offre nouvelle. Ce modèle de langage, dont la propriété essentielle est de définir une bonne métrique sur l'espace des offres d'emploi, est implémenté par un réseau neuronal et optimisé en définissant deux fonctions de perte. La première vise à préserver la structure locale des informations collaboratives, en s'inspirant des approches de réduction de dimension non linéaires. La seconde s'inspire des réseaux siamois pour reproduire la métrique définie par la matrice collaborative. Le passage à l'échelle de l'approche et ses performances reposent un échantillonnage spécifique des paires d'offres (considérées comme similaires et dissimilaires).

Le mérite de l'approche proposée est validé empiriquement sur les données réelles, ainsi que sur les données publiques du problème CiteULike (visant à recommander des articles scientifiques) permettant la comparaison du système proposé avec les systèmes de l'état de l'art.

Nous avons aussi démontré l'intérêt de notre approche en participant au challenge international RecSys 2017, dont les données comprennent plus d'un million de demandeurs et d'offres d'emploi. Notre rang est 15 sur 100.

Abstract

The presented work focuses on the recommendation to job seekers of new job positions (cold start recommendation). The proposed approach builds upon the state of the art in recommendation systems for e-commerce, and adapt the existing systems by exploiting i) the collaborative data reporting the interactions of job seekers with previous job openings ; ii) the job opening and CV documents.

One of the specificities of the presented work is to consider real-world data, kindly provided by the Web Hiring Agency Qapa on the one hand, and the Association Bernard Gregory on the other hand. We have thus faced the challenges of heterogeneous job sectors (blue and white collars), and the noise of textual documents.

Our contribution is the design and implementation of a language model, driven by the collaborative data and supporting the cold start recommendation. This language model, essentially aimed to define a relevant metric on the space of job openings, is implemented as a neural network and trained according to two loss functions. The former one aims to preserve the local structure of collaborative information, drawing on non-linear dimension reduction approaches. The latter one, inspired by Siamese networks, aims to reproduce the similarities induced from the collaborative data. The scaling up of the approach and its performance are based on specific samplings of the so-called positive and negative pairs of offers (deemed similar or dissimilar).

The merit of the proposed approach is demonstrated empirically on the real-world data ; for the sake of comparison with the state of the art, we also considered the public domain benchmark CiteULike, aimed at recommending scientific articles to scientists.

We also demonstrated the potential of our approach by participating into the international challenge RecSys 2017, involving over one million of job seekers and job openings. Our final rank was 15th out of 100 participants.

À mes amis de licence

Table des matières

Notations

— $\boldsymbol{x} \in \mathbb{R}^d$ désigne le vecteur $\boldsymbol{x} = (x^1, \ldots, x^d)$, avec $\forall j, x^j \in \mathbb{R}$;
— $X \in \mathbb{R}^{n \times d}$ désigne un ensemble de n points dans l'ensemble $\mathbb{R}^d$;

$$X = (\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}) \ \forall i, \boldsymbol{x_i} \in \mathbb{R}^d$$

— M^T la transposée de la matrice M ;
— $\mathbb{1}_A$ l'indicatrice de l'évènement A ;
— $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ désigne le produit scalaire entre les vecteurs $\boldsymbol{u} \in \mathbb{R}^d$ et $\boldsymbol{v} \in \mathbb{R}^d$. $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \sum_{i=1}^{d} u^i \cdot v^i$;
— $D_{KL}(\boldsymbol{p}||\boldsymbol{q})$ la divergence de Kullback-Leibler entre les distributions $\boldsymbol{p}$ et $\boldsymbol{q}$. Dans le cas où $\boldsymbol{p}$ et $\boldsymbol{q}$ sont des distributions sur un espace fini,

$$D_{KL}(\boldsymbol{p}||\boldsymbol{q}) = \sum_i p_i \log(p_i) - p_i \log(q_i)$$

Pour les systèmes de recommandation :

— u et v désignent des utilisateurs ; l'ensemble des utilisateurs est noté $\mathcal{U}$;
— $\mathcal{U}(i)$ représente l'ensemble des utilisateurs ayant sélectionné l'objet i ;
— i et j désignent des objets ; l'ensemble des objets est noté $\mathcal{O}$;
— $\mathcal{O}(u)$ représente l'ensemble des objets sélectionné par l'utilisateur u ;
— $\mathcal{R}$ est la matrice collaborative, de taille $|\mathcal{U}| \times |\mathcal{O}|$;
— $\mathcal{R}_{u,i}$ est la note d'un utilisateur u sur un objet i ;
— $\widehat{\mathcal{R}}_{u,i}$ est l'estimation de la note $\mathcal{R}_{u,i}$;
— κ-nn(j) représente les κ plus proches voisins de l'objet j ;

Pour les documents :

— $\mathcal{V}$ est l'ensemble du vocabulaire ;
— $n_{\mathcal{V}} = |\mathcal{V}|$ est la taille du vocabulaire ;
— n_d est le nombre de documents ;
— $\mathbf{D}$ est la matrice "documents $\times$ termes" $\in \mathbb{R}^{n_d \times n_{\mathcal{V}}}$;

Introduction

Contexte économique et social

La motivation applicative de cette thèse concerne le chômage. En France, le chômage touche plus de 10% de la population active en 2015. Plus spécifiquement, les travaux présentés s'intéressent au chômage des non-diplômés, particulièrement aigu en France.

Chômage En effet, d'après l'enquête *Génération 2004* menée sur l'insertion des jeunes par le Centre d'études et de recherches sur les qualifications (Mazari and Recotillet (2013)), deux tiers des jeunes non diplômés sont entrés très jeunes dans la vie active (18 ans en moyenne) et durant les 3 premières années de leur vie professionnelle, 53 % d'entre eux ont passé plus de 6 mois au chômage (et 36 % plus d'un an). Après 3 ans, près d'un tiers étaient toujours au chômage. Sur la tranche des 50-64 ans, 33% ont au plus un brevet des collèges. De plus, quand les non diplômés accèdent à un poste, il s'agit souvent d'un poste précaire ; 31% des non diplômés intègrent le monde du travail dans le cadre d'un contrat à durée déterminée (CDD) ; 27% sont intérimaires.

Le taux de chômage des non diplômés est trois fois plus élevé (20%) que celui des personnes qui disposent d'un diplôme niveau bac + 2 (6%) [1]. Chez les non-diplômés de moins de 29 ans, le taux de chômage n'est pas loin de 40% [2]. L'insertion des non-diplômés fait l'objet d'une attention particulière des pouvoirs publics ; les travaux récents consacrés aux problèmes de l'emploi dans le contexte de la science des données (Lu et al. (2013); Siting et al. (2012)) ciblent par contre les emplois à forte valeur ajoutée pour des raisons évidentes.

Chômage frictionnel Nous n'avons pas la prétention de nous attaquer au phénomène global du chômage, dont les causes et les modalités sont multiples. Notre étude se focalise sur l'un des modes du chômage, appelé chômage frictionnel. Selon Pissarides (2011), le chômage frictionnel est lié aux imperfections informationnelles, dûes aux coûts de collecte, de traitement et de diffusion de l'information, ainsi qu'à l'asymétrie d'information entre offreurs et demandeurs, et aux limitations cognitives des individus. Ces imperfections sont une des raisons pour laquelle certains emplois restent inoccupés alors même qu'un chômage important est observé dans les mêmes secteurs d'emploi.

Nos travaux partent de l'idée qu'un meilleur accès à l'information améliorerait la situation en termes de recherche d'emploi et d'employé, en diminuerait le coût en temps et en opportunité, et permettrait de remédier dans une certaine mesure au phénomène de chômage frictionnel. La

1. `https://www.inegalites.fr/Le-taux-de-chomage-selon-la-categorie-sociale?id_theme=16`
2. `https://www.inegalites.fr/Le-taux-de-chomage-selon-le-diplome?id_theme=16`

fluidification de l'information est d'autant plus importante dans les secteurs à bas salaire, que la description des offres y est plus succincte et que la durée de vie d'une offre peut être courte.

Quoique les problèmes de recherche d'emploi, d'une part, et de recherche d'employé i.e. de recrutement, d'autre part, soient identiques d'un point de vue abstrait, ils diffèrent toutefois fortement en pratique. Seul le problème de la recherche d'emploi, où l'utilisateur considéré est le chercheur d'emploi, sera considéré dans la suite de ce document.

Appariement direct des offres et des CVs La solution envisagée pour pallier ces problèmes d'information est la recommandation individualisée d'offres d'emploi adaptées au profil du chercheur d'emploi. Dans la littérature des recommandations fondées sur les descriptions des offres et des *Curriculum Vitae* (CVs), utilisant des mots-clefs ou des techniques de recherche d'information, sont proposées (Faliagka et al., 2012; Singh et al., 2010). Les problèmes rencontrés sont liés entre autres à la nature différente des langages utilisés pour décrire les utilisateurs (chercheurs d'emploi) et les objets (offres d'emploi), comme nous le verrons en chapitre 3.2.2. Une solution passe par la définition de compétences standards [3], constituant idéalement un langage commun entre offres et CVs. Cependant, cette liste de compétences ne répond que partiellement au problème posé. Tout d'abord, il s'agit d'une liste qui évolue plus lentement que les métiers et les compétences demandées dans l'ensemble des secteurs (depuis les métiers sociaux à faible qualification jusqu'aux métiers high-tech). En second lieu, l'identification par les utilisateurs des termes correspondant à leurs compétences réelles entraine un certain bruit ; symétriquement, les recruteurs peuvent aussi n'identifier que partiellement les compétences pertinentes pour les postes à pourvoir.

Par ailleurs, d'autres facteurs rentrent en jeu lors d'un appariement direct entre CVs et offres. En effet, les offres et les CVs font intervenir des informations semi-structurées, contenant outre les compétences et l'expérience des notions géographiques, un intervalle de salaire et des perspectives d'évolution. L'importance de ces informations dépend du contexte, du secteur d'emploi et du poste à pourvoir : comme nous le verrons au chapitre 6.4.3, la mobilité géographique a relativement peu d'impact dans le contexte des offres d'emploi scientifiques, comparé aux emplois à très faible salaire ; inversement, des différences de salaire très faible ont beaucoup plus d'impact sur le nombre de candidats dans le contexte des bas salaires que dans celui des hauts salaires (même en considérant les différences *relatives*).

Evolution du marché de l'emploi Classiquement, un demandeur d'emploi rédige un CV qui sera envoyé aux services des ressources humaines (RH) des centaines d'entreprises dans lesquelles il ou elle postule. La formation à la rédaction d'un CV est d'ailleurs l'une des principales offres de service de Pôle Emploi. La qualité d'un CV joue en effet un rôle déterminant sur le recrutement en particulier dans le contexte d'un chômage important, où un directeur RH reçoit des centaines de CVs pour chaque poste ouvert.

La situation a évolué avec l'afflux d'entreprises, notamment sur la Toile [4], proposant des services d'aide d'accès à l'emploi et se fondant sur l'historique de recherche des offres d'emploi ainsi que les traces d'usages lors de la navigation (naturellement disponible sur les plate-formes de recrutement). Cet historique et ces traces enrichissent l'information des CVs voire s'y substituent, fournissant des caractéristiques descriptives spécifiques pour chaque utilisateur. Par analogie avec le contexte de l'e-commerce, nous dirons que le profil d'un utilisateur est décrit ou complété par ses choix, et

3. Pôle-emploi met à disposition des fiches métiers, décrivant les compétences de base spécifiques à chaque métier.
4. Voir Monster, Indeed, Qapa, MeteoJob, Keljob

que l'ensemble des choix d'une communauté d'utilisateurs fournit une riche information appelée information collaborative ou *données collaboratives*.

De manière analogue, certaines plate-formes de musiques en ligne telles Deezer [5] ou Spotify [6] demandent aux utilisateurs nouveaux de noter certaines musiques et de donner leurs préférences sur des thèmes musicaux, par opposition à décrire explicitement leur profil (comme un CV). Cette description initiale de l'utilisateur est graduellement enrichie par ses traces de navigation, et nourrit les recommandations individualisées du système.

Historiquement, l'une des plate-formes emblématiques pour la recommandation est la plate-forme Netflix, qui a organisé un challenge de recommandation en 2006 [7], dont l'objectif était de prédire la note qu'un utilisateur donne à un film, à partir uniquement de ses notes sur les films qu'il a déjà vus.

Problématique

A la suite de cette analyse du besoin et de l'évolution des usages, la mise en place de systèmes automatiques de recommandation personnalisée d'offres d'emploi apparait ainsi comme l'une des clefs pour lutter contre le chômage frictionnel. Le problème de recherche ainsi défini devient le suivant :

Comment utiliser les traces d'usages d'une communauté d'utilisateurs pour recommander de *nouvelles* offres d'emploi, à partir de leur description en langage naturel ?

Les défis posés sont de trois natures. Les méthodes de recommandation collaborative doivent être adaptées au marché du travail, et une difficulté majeure consiste à disposer des données disponibles. Nous avons eu la chance de disposer de ces données dans le cadre des liens du Laboratoire avec la société Qapa [8] ainsi qu'avec l'association Bernard Gregory [9] (ABG).Une seconde difficulté consiste à adapter ces méthodes au traitement automatique du langage naturel, puisque les CVs et les offres d'emploi sont décrits dans ce cadre. Enfin, le troisième défi consiste à traiter les nouvelles offres d'emploi, qui n'ont pas encore de traces d'usage : il s'agit du problème dit de "démarrage à froid" dans le cadre de la recommandation collaborative.

Démarrage à froid Les offres d'emploi sont souvent à durée limitée, et l'objectif est de pouvoir les recommander aussi vite que possible, idéalement dès qu'elles arrivent sur la plate-forme. L'objectif prioritaire consiste donc à recommander des offres d'emploi *nouvelles* sur la plate-forme, i.e. postées récemment, à des utilisateurs inscrits sur le plate-forme [10], définissant ainsi un problème de recommandation "à froid".

Le démarrage à froid pose des difficultés liées au manque d'information. Dans le cas classique du commerce électronique, la recommandation s'applique essentiellement à des utilisateurs connus,

5. deezer.com
6. spotify.com
7. `www.netflixprize.com`
8. Nous tenons à remercier chaleureusement Stéphanie Delestre, PDG de Qapa, pour sa collaboration, la grande qualité des données fournies ainsi que pour son intérêt pour notre étude.
9. Nous tenons à remercier l'équipe d'ABG pour leur confiance et l'intérêt qu'ils ont porté à notre projet.
10. Un autre problème de démarrage à froid concernerait la recommandation d'offres connues à des utilisateurs nouveaux, i.e. venant de s'enregistrer sur la plate-forme. Ce second problème de démarrage à froid ne sera pas considéré dans la suite de ce manuscrit.

qui viennent et reviennent sur la plate-forme ainsi qu'à des produits qui restent sur la plate-forme (au moins un certain temps), et s'enrichissent d'interactions, de notes et de commentaires divers. De plus, dans le commerce électronique, la disponibilité des produits n'est pas limitée à une zone géographique ni à une période donnée, contrairement aux offres d'emploi. En effet, quand une offre trouve preneur, elle n'est plus disponible (contrairement aux films ou aux livres).

Position et impact de la recherche

Les travaux présentés se placent à l'intersection de plusieurs domaines de recherche en apprentissage. Par construction, ainsi que nous l'avons dit, le problème posé relève de la recommandation collaborative (Koren et al., 2009) et du traitement des langues naturelles (Manning et al., 1999).

Nous proposons une approche originale se fondant sur les réseaux neuronaux siamois (Bromley et al., 1994) pour répondre au défi en termes de qualité et de passage à l'échelle. Ces solutions s'inspirent de l'apprentissage de distance (permettant de définir une similarité entre offres, guidée par les données collaboratives) et des modèles continus du langage par réseaux neuronaux, inspirés de Bengio et al. (2003), et permettant de plonger les offres dans l'espace métrique $\mathbb{R}^d$.

Impact vis à vis du filtrage collaboratif Le socle disciplinaire de la recommandation collaborative consiste à dire qu'un utilisateur est "suffisamment" décrit par ses usages et son interaction avec les objets, particulièrement lorsque l'on dispose de la vue globale sur la communauté des utilisateurs, donnée par l'ensemble de leurs interactions avec les mêmes objets.

L'intérêt particulier du domaine de recherche considéré est dû au fait que nous disposons aussi d'une description explicite des utilisateurs et des objets : respectivement les CVs et les fiches de poste. Nous sommes ainsi en mesure de mettre en évidence les différences entre ce que les gens disent (leur CV) et ce qu'ils font (leur choix d'offres est souvent plus large que leur profil). Une seconde leçon concerne la différence des langages entre les utilisateurs (les demandeurs) et les offres ; nous mettrons en évidence le fait que i) ces langages sont différents (notamment en termes de distribution ; voir également Ruotsalo et al. (2015)) ; et ii) la différence de ces langages n'empêche pas les utilisateurs de cliquer sur certaines offres : tout se passe comme si les utilisateurs savaient "lire entre les lignes".

Impact vis à vis du traitement des langues naturelles Dans le cas du traitement des langues naturelles, un contexte classique est celui de l'apprentissage supervisé – chaque document est associé à une étiquette et le but est d'entrainer un classifieur, voir par exemple pour l'analyse de sentiments (Maas et al., 2011). Dans le contexte non supervisé, les objectifs visés concernent souvent la recherche des thèmes du documents (Blei et al., 2003) ou la construction d'un modèle de langue (Jozefowicz et al., 2016).

Notre objectif est intermédiaire entre celui de l'apprentissage supervisé et non-supervisé. Formellement, le but est de définir une représentation des documents qui induise une similarité entre offres d'emploi, pour obtenir *in fine* un système de recommandation. Plus précisément, la représentation cherchée doit être guidée par les informations contenues dans les données collaboratives, telle que la similarité dans l'espace de représentation reflète la similarité pragmatique des offres d'emploi, i.e. le fait qu'elles attirent les mêmes personnes. Calibrer cette représentation, et assurer son passage à l'échelle est un défi.

Analyse des données Ainsi que dit précédemment, nos travaux sont motivés par, et validés sur, des corpus réels, gracieusement mis à notre disposition par la société Qapa et l'association Bernard Gregory (ABG). Chaque corpus contient les textes des CVs et des offres d'emploi, ainsi que les données collaboratives indiquant quel utilisateur a cliqué sur quelle offre d'emploi. Une étape préliminaire et essentielle de recherche, propre à toutes les applications d'apprentissage, consiste à étudier les données et leur qualité. Cette analyse concerne en particulier la diversité des langues utilisées par les demandeurs d'emploi et les recruteurs. Un second axe d'analyse confronte les données textuelles et les données collaboratives et en examine la cohérence. A titre de référence, ces données sont comparées au corpus comparable CiteULike [11], concernant la recommandation d'articles dans le contexte d'une communauté scientifique. Les différences essentielles concernent d'une part le niveau de langage (homogène dans le contexte de CiteULike) et d'autre part le fait que les utilisateurs de CiteULike ne sont pas décrits.

Apprentissage de métriques Enfin, les contributions essentielles de notre travail consistent à apprendre une topologie ou une métrique sur l'espace textuel des offres d'emploi. Cette topologie sera utilisée pour la recommandation en démarrage à chaud et à froid. Dans un souci d'évaluation reproductible, ces méthodes seront évaluées sur le corpus CiteULike et comparées avec les méthodes d'état de l'art (Wang and Blei, 2011).

La première approche proposée, appelée Collaborative Local Embedding (CLE), se fonde sur la structure locale des offres issue des données collaboratives et s'inspire des approches de réduction de la dimensionalité localement linéaires (*Locally Linear Embedding*, (Roweis and Saul, 2000)). Cette représentation latente définit une structure "oracle" et une représentation du texte est apprise pour tenter de conserver cette structure oracle.

La seconde approche, Langage Modeling for Job Applicant Matching (LaJam), s'inspire des réseaux siamois (Bromley et al., 1994) et apprend de même une représentation continue du langage des offres dans le but d'émuler directement les similarités issues de la matrice collaborative.

Plan Le manuscrit est organisé comme suit :

- Le premier chapitre décrit brièvement les concepts de base des réseaux de neurones, qui interviendront comme espaces d'hypothèse pour les deux contributions de la thèse.
- L'état de l'art dans le domaine du traitement automatique des langues naturelles, en se limitant aux approches de types sacs de mots, et les représentations continues du langage sont décrits dans le chapitre 2.
- L'état de l'art dans le domaine des systèmes de recommandation est décrit au chapitre 3, en détaillant les notions de démarrage à chaud (le contexte classique) et le démarrage à froid (le contexte pertinent pour notre étude).
- Les chapitres suivants décrivent les corpus qui ont été utilisés dans le cours de notre étude, et analysent leur qualité du point de vue de la distribution du vocabulaire dans les CVs et les offres, d'une part (chapitre 4), et du point de vue des données collaboratives, d'autre part (chapitre 5).
- Notre premier contribution est l'approche CLE, publiées en (Schmitt et al., 2016), présentée en annexe A. La seconde approche, LaJam, publiées en (Schmitt et al., 2017), améliore CLE et apprend une métrique issue directement de la matrice collaborative (chapitre 6).
- Enfin, le dernier chapitre décrit notre participation au challenge RecSys 2017.

11. CiteUlike.org

Et la thèse s'achève sur une discussion des perspectives de recherches ouvertes par notre travail.

Chapitre 1

Réseaux de neurones

Cette première section introduit les principaux aspects des réseaux de neurones ainsi que les méthodes d'optimisation, qui interviennent dans les approches présentées dans la suite de la thèse :

— ils permettent d'extraire l'information de manière non-supervisée des documents textuels (chapitre 2) ;
— ils sont employés lors de la recommandation directe, permettant de factoriser la matrice collaborative (chapitre 3) ;
— ils extraient une représentation dont la métrique est guidée par les informations collaboratives (Annexe A et chapitre 6).

L'espace des réseaux de neurones est connu pour ses qualités en termes de représentation de fonctions (les réseaux de neurones étant denses dans l'espace des fonctions de carré intégrable sur un compact Hecht-Nielsen (1992)). Un autre élément de la popularité de ces réseaux est l'existence d'algorithmes permettant d'apprendre une approximation d'une fonction donnée à partir d'exemples de cette fonction, la qualité de l'approximation dépendant du nombre d'exemples disponibles et de la complexité de la fonction.

Les premiers réseaux de neurones artificiels, inspirés par les réseaux biologiques (McCulloch and Pitts, 1943), sont popularisés dans le cadre du perceptron par Rosenblatt (1958). Un neurone est une fonction généralement non-linéaire de ses entrées ; la structuration en graphe dirigé d'un ensemble de neurones (les sorties des uns formant les entrées des autres) définit un réseau neuronal. On distingue classiquement les structures en couches (*feedforward*) correspondant à un graphe sans cycle, et les réseaux récurrents (Rumelhart et al., 1985) ; ces derniers ne sont pas utilisés dans la suite de ce document.

Un réseau neuronal est décrit par son architecture (le graphe d'interaction ci-dessus) et les paramètres des fonctions d'activation de chaque neurone, appelés poids. L'apprentissage du réseau consiste à définir les valeurs des poids, obtenues par l'optimisation d'une fonction objectif dépendant du problème, où l'optimisation repose généralement sur une méthode de descente de gradient (Section 1.2).

Les réseaux de neurones ont connu un regain de popularité dans les années 2000 grâce à la conjonction de trois faits : une puissance de calcul sans précédent ; l'accès à des données massives ; de nouvelles architectures et/ou de nouveaux algorithmes d'apprentissage.

L'apprentissage de réseaux de neurones profonds (Rumelhart et al., 1986; LeCun et al., 1998b) (voir (LeCun et al., 2015) pour une introduction plus complète) intègre les phases de construction

de représentation des données (faisant éventuellement intervenir plusieurs niveaux d'abstraction correspondant aux couches) et les phases de modélisation fondées sur ces représentations. Ces approches ont considérablement amélioré l'état de l'art dans les domaines du langage naturel (Collobert et al., 2011b) (notamment pour la classification de documents, l'analyse de sentiments, les réponses automatiques (Bordes et al., 2014), la traduction automatique (Sutskever et al., 2014) et la reconnaissance du langage parlé (Mikolov et al., 2011)), la vision par ordinateur (Krizhevsky et al., 2012; Farabet et al., 2013) et de nombreux autres domaines, allant de la découverte de médicaments (Ma et al., 2015) à la génomique (Xiong et al., 2015).

Dans la recherche présentée, les réseaux de neurones permettent de construire une représentation des données textuelles disponibles (offres d'emploi et CVs) pour permettre la mise en relation des offres d'emplois avec les demandeurs d'emploi.

Nous décrirons ici les architectures neuronales les plus utilisées dans la suite de ce document : le perceptron multi couches et l'Auto-Encodeur (ainsi que sa variante, le denoising Auto-Encodeur). L'architecture plus spécifique des réseaux siamois (Bromley et al., 1994), utilisée de manière centrale dans notre approche, est décrite de manière approfondie au chapitre 6.

La seconde partie de ce chapitre présente les méthodes classiques d'apprentissage des réseaux, et discute les hyper-paramètres de l'apprentissage.

1.1 Modèle de réseaux

1.1.1 Perceptron multi-couches

La forme la plus classique d'un réseau de neurones, appelée perceptron (Rosenblatt, 1958), correspond à une fonction de la forme :

$$\begin{aligned} f_{W,\boldsymbol{b}} : \ & \mathbb{R}^D \to \mathbb{R}^d \\ & \boldsymbol{x} \ \mapsto \sigma(W \cdot \boldsymbol{x} + \boldsymbol{b}) \end{aligned}$$

La matrice des poids $W \in \mathbb{R}^{D \times d}$ ainsi que le vecteur des biais $\boldsymbol{b} \in \mathbb{R}^d$ sont les paramètres du perceptron. D est la dimension de l'espace d'entrée. Dans le cas supervisé (classification ou régression), la dimension de l'espace de sortie d est imposée par le problème (nombre de labels ou dimension de la régression vectorielle). La fonction σ à valeurs dans $\mathbb{R}^d$ se compose de d fonctions non-linéaires (e.g. $tanh, ReLU$ ou *sigmoide*) appliquées à chaque coordonnée du vecteur $W \cdot \boldsymbol{x} + \boldsymbol{b}$. En résumé $f_{W,\boldsymbol{b}}$ est une fonction vectorielle non-linéaire composée avec une fonction affine.

Les réseaux perceptron multi-couches sont une généralisation du perceptron. Ils empilent plusieurs couches de type perceptron, où la sortie de chaque couche définit l'entrée de la couche suivante. Le réseau de neurones définit une fonction de type

$$\phi = f_{W_1,\boldsymbol{b_1}} \circ \ldots \circ f_{W_p,\boldsymbol{b_p}}$$

où p dénote le nombre de couches (ou profondeur) du réseau.

Les paramètres W_i et $\boldsymbol{b_i}$ du modèle sont appris par minimisation de la fonction objectif, dépendant du problème considéré (e.g. entropie croisée dans le contexte de la classification, erreur quadratique dans le cadre de la régression). La manière de mettre à jour ses paramètres est décrite en (Section 1.2).

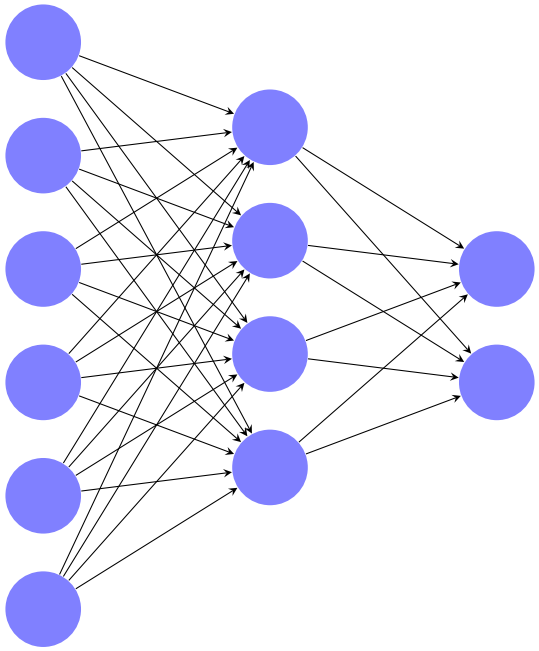


FIGURE 1.1 – Schéma d'un réseau de neurones à une couche cachée

1.1.2 Auto-Encoder

Une des approches neuronales dans le cas non-supervisé est l'Auto-Encodeur. Cette approche permet d'opérer une réduction de dimension, induisant une nouvelle métrique sur les données. Les Auto-Encodeurs sont également utilisés pour dé-bruiter les données ou pour les visualiser, soit directement en fixant $k = 2$ ou 3 (Hinton and Salakhutdinov, 2006), soit en phase préalable d'une visualisation t-SNE (Maaten and Hinton, 2008).

Un auto-encodeur suit le principe suivant. Soit $\boldsymbol{x} \in \mathbb{R}^D$ la représentation initiale d'un objet (e.g. le vecteur tf-idf d'un document ou l'ensemble des pixels d'une image). Une fonction f_θ appelée "encodeur" envoie $\boldsymbol{x}$ sur $f_\theta(\boldsymbol{x}) = \boldsymbol{y}$ dans $\mathbb{R}^k$, avec k un hyperparamètre du modèle (la dimension de la couche intermédiaire), habituellement $k << D$. Une autre fonction g_θ appelée "décodeur" envoie $\boldsymbol{y}$ sur une représentation de même dimension que la dimension initiale

$$g_\theta(\boldsymbol{y}) = \boldsymbol{z}$$

où f et g sont optimisées pour minimiser l'erreur de reconstruction au sens des moindres carrés $L_\theta(\boldsymbol{x}, \boldsymbol{z}) = \|\boldsymbol{x} - \boldsymbol{z}\|_2^2$ par descente de gradient.

Dans le cas où f et g comprennent une seule couche cachée, les fonctions sont données par :

$$f_\theta(\boldsymbol{x}) = \sigma(W\boldsymbol{x} + \boldsymbol{b})$$
$$g_\theta(\boldsymbol{y}) = \sigma(W'\boldsymbol{y} + b')$$

où $W \in \mathbb{R}^{D\times k}$, $W' \in \mathbb{R}^{k\times D}$ et $\boldsymbol{b} \in \mathbb{R}^d$ un vecteur de biais. σ est une fonction d'activation non-linéaire. Dans le cas dit des poids liés, $W' = W^T$ est imposé (Vincent et al., 2008), réduisant ainsi le nombre de paramètres et contraignant le modèle, minimisant ainsi le risque de sur-apprentissage.

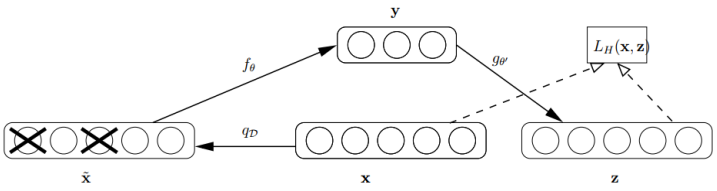


FIGURE 1.2 – Denoising Auto-Encoder. La fonction de perte $L(\boldsymbol{x}, \boldsymbol{z})$ est calculée sur l'entrée *non-bruitée* et la sortie du décodeur

Dans le cas d'une fonction d'activation linéaire ($\sigma(\boldsymbol{x}) = \boldsymbol{x}$) avec une seule couche, la compression apprise par un Auto-Encodeur avec k neurones sur la couche cachée approxime la projection sur les k premiers axes (vecteurs propres) d'une analyse par composantes principales (Bourlard and Kamp, 1988).

1.1.3 Denoising Auto-Encoder

Une représentation plus robuste est obtenue en bruitant l'entrée de l'auto-encodeur (Vincent et al., 2008). Soit $\tilde{\boldsymbol{x}}$ une représentation obtenue à partir de $\boldsymbol{x}$ en ajoutant un bruit Gaussien ou en annulant des coordonnées aléatoires (bruit de type "dropout"). L'auto-encodeur est optimisé pour minimiser la perte par rapport à l'entrée originelle x.

$$L(\boldsymbol{x}, \boldsymbol{z}) = \|\boldsymbol{x} - \boldsymbol{z}\|_2^2$$

avec $z = g_\theta(f_\theta(\tilde{\boldsymbol{x}}))$.

L'approche Denoising Auto-Encoder impose que le gradient de l'encodeur soit (quasi) nul au voisinage des points considérés (Vincent et al., 2008), ce qui agit comme une régularisation et prévient le phénomène de sur-apprentissage.

Dans le cas d'un auto-encodeur profond (stacked auto-encoder), l'encodeur comme le décodeur font intervenir plusieurs couches neuronales (Vincent et al., 2010). L'objectif est d'obtenir une représentation plus abstraite et plus riche au niveau des couches intermédiaires, souvent de plus faible dimension ; la représentation est toujours donnée par $\boldsymbol{y} = f_\theta(\boldsymbol{x})$. Cette approche a notamment obtenu de très bons résultats dans le contexte de l'analyse de sentiment (Glorot et al., 2011).

1.2 Optimisation d'un réseau de neurones et hyperparamètres

Les méthodes d'optimisation des réseaux de neurones sont un aspect crucial dans notre étude. Nous décrivons ici les principales techniques utilisées lors de l'entraînement d'un réseau de neurones.

1.2.1 Descente de gradient

La descente de gradient, formulée pour les réseaux de neurones (Rumelhart et al., 1986), vient de l'optimisation fonctionnelle. En effet, tout réseau de neurones représente une fonction différentiable en tout point : la descente de gradient peut alors s'appliquer. Il s'agit de minimiser la fonction objectif ℓ (dépendant du problème) sur l'ensemble des données d'entraînement, en modifiant les paramètres internes du modèle. Cela peut être obtenu en calculant le gradient sur l'ensemble des données. Cependant, cette opération étant coûteuse, nous séparons habituellement l'ensemble d'entraînement en une partition aléatoire, où chaque partie est appelé batch. Les paramètres du modèle sont enfin mis à jour suivant le gradient calculé sur les éléments du batch X, nous avons ainsi

$$\boldsymbol{\theta_{i+1}} = \boldsymbol{\theta_i} - \alpha \nabla \ell_{\boldsymbol{\theta_i}}(X)$$

Ici X correspond aux données du batch, α au pas d'apprentissage (learning rate) et $\nabla \ell_{\boldsymbol{\theta_i}}(X)$ est la moyenne des gradients de f paramétré par $\boldsymbol{\theta_i}$ aux points de X. Cette mise à jour est facilement calculée par back-propagation de l'erreur, dont une présentation peut-être trouvée dans (LeCun et al., 1998b).

La stochasticité de cette descente de gradient est apportée par les batch. De plus, une taille de batch faible (entre 2 et 32 éléments par batch) confère une plus grande stabilité pour l'intervalle du pas d'apprentissage, c'est à dire que l'intervalle de valeurs du pas d'apprentissage garantissant une convergence du réseau est plus large (Keskar et al., 2016; Masters and Luschi, 2018).

1.2.1.1 Momentum

De nombreuses améliorations sont apportées à cette descente de gradient stochastique, dont la première est l'ajout d'un momentum qui vise à lisser le gradient. Cela accélère non seulement la convergence mais stabilise également l'apprentissage, augmentant la robustesse de la convergence par rapport au pas d'apprentissage (Sutskever et al., 2013). En posant $\boldsymbol{z_i}$ la variation des paramètres à l'étape i, la mise à jour des poids s'écrit :

$$\begin{aligned}\boldsymbol{z_{i+1}} &= \beta \boldsymbol{z_i} - (1-\beta)\nabla \ell_{\theta_i}(X) \\ \boldsymbol{\theta_{i+1}} &= \boldsymbol{\theta_i} + \alpha \boldsymbol{z_i}\end{aligned}$$

Ici $\beta \in [0, 1]$ est un paramètre contrôlant la force du momentum. Ainsi, plus β est proche de 1, plus le momentum est élevé, si $\beta = 0$ nous retrouvons la descente de gradient classique.

1.2.1.2 Adam

Une méthode récente appréciée et utilisée par la communauté est la méthode Adam (Kingma and Ba, 2014) dont la robustesse est discutée par Ruder (2016).

Adam repose sur le lissage du gradient, ajusté par le facteur β_1 ainsi que de son carré (composante par composante), ajusté par le facteur β_2. Ces lissages sont effectués par la formule :

$$\boldsymbol{m_{i+1}} = \beta_1 \boldsymbol{m_i} + (1-\beta_1)\nabla \ell_{\theta_i}(X)$$
$$\boldsymbol{v_{i+1}} = \beta_2 \boldsymbol{v_i} + (1-\beta_2)(\nabla \ell_{\theta_i}(X))^2$$

Les valeurs de m_0 et v_0 sont initialisées à zéro, causant un biais, particulièrement durant les premières étapes. Pour corriger ce biais, la mise à jour fait intervenir :

$$\boldsymbol{m_i} = \frac{\boldsymbol{m_i}}{(1-\beta_1^i)}$$
$$\boldsymbol{v_i} = \frac{\boldsymbol{v_i}}{(1-\beta_2^i)}$$

enfin, les paramètres sont mis à jour selon la formule :

$$\boldsymbol{\theta_{i+1}} = \boldsymbol{\theta_i} + \alpha \frac{\hat{\boldsymbol{m}}_{\boldsymbol{i}}}{(\sqrt{\hat{\boldsymbol{v}}_{\boldsymbol{i}}} + \varepsilon)}$$

La constante ε, souvent fixée à 10^{-8}, permet d'éviter des divisions par zéro. Les valeurs par défaut sont $\beta_1 = 0.9$ et $\beta_2 = 0.999$.

1.2.2 Hyperparamètres

1.2.2.1 Fonctions d'activations

Les fonctions d'activations les plus usitées sont, dans l'ordre chronologique :
— sigmoïde, $x \mapsto \frac{1}{1+e^{-x}}$
— tanh, $x \mapsto \frac{e^x - e^{-x}}{e^x + e^{-x}}$
— ReLU (Rectified linear units), $x \mapsto max(0, x)$ (He et al., 2015).

Les fonctions *sigmoide* et *tanh* soufrent de saturation dans les régions extrêmes, où leur gradient est proche de zéro, ce qui rend inactifs ces neurones qui restent collés à 1 ou à 0 durant tout l'apprentissage.

La ReLU est au coeur des récents succès des réseaux profonds. En effet, il y a moins de problèmes de disparition du gradient car son gradient est soit de zéro pour les valeurs négatives, soit de 1 pour les valeurs positives.

1.2.2.2 Initialisation

L'initialisation des poids du réseau peut également jouer un rôle critique durant l'apprentissage. En effet, certains choix d'initialisation aboutissent à des performances médiocres (Sutskever et al., 2013). Deux méthodes d'initialisation sont populaires. La première est appelée Glorot (Glorot and Bengio, 2010a), où les poids initiaux sont tirés par une loi normale centrée en 0 de déviation standard $\sqrt{\frac{2}{fan_{in}+fan_{out}}}$, dont la valeur absolue est tronquée à 1, où fan_{in} est le nombre de neurones entrant du neurone et fan_{out} le nombre de neurones sortant du neurone. Cette initialisation permet de contrôler la distribution des valeurs de la sortie, remédiant ainsi au problème de saturation des fonctions d'activation. La seconde méthode d'initialisation appelée He (He et al., 2015) est très proche de l'initialisation Glorot. Elle utilise une déviation standard de $\sqrt{\frac{2}{fan_{in}}}$, jugée plus robuste pour les fonctions d'activation de type ReLU.

1.2.2.3 Dropout

Le dropout (Hinton et al., 2012; Srivastava et al., 2014) est une méthode pour prévenir le sur-apprentissage. Pour chaque exemple d'entraînement présenté au réseau, chaque unité cachée est aléatoirement fixée à 0 avec une certaine probabilité p. Le dropout permet de prévenir la co-adaptation des neurones, la contribution d'un neurones s'ajoutant a celle d'un ensemble aléatoire de neurones de la même couche, chaque neurone est encouragé à apporter une contribution utile et indépendante des autres.

Lors de la prédiction, tous les neurones sont actifs et leur valeur est multipliée par $1 - p$. En contre partie, le dropout augmente généralement le temps d'apprentissage par un facteur deux ou trois.

1.2.2.4 Normalisation par batch

Il est recommandé de normer et centrer les données en entrée du réseau de neurones pour éviter l'évanescence et/ou l'explosion du gradient. Cependant, les couches cachées ne sont pas normalisées, ce qui peut amener ces inconvénients.

La normalisation par batch (Ioffe and Szegedy, 2015) permet d'étendre les bénéfices de la normalisation au niveau de chaque couche cachée.

Dans ce cas, l'entrée de chaque neurone est indépendamment normalisée, pour avoir une moyenne de zéro sur l'ensemble des données d'apprentissage et une variance de un. Ici, la moyenne et la variance sont estimées sur l'ensemble d'entraînement. Expérimentalement, la normalisation par batch permet, sur des données comme ImageNet, de réduire d'un ordre de grandeur le temps d'entraînement ainsi que d'améliorer le score de prédiction [1]. Cette technique est maintenant une des composantes standard des méthodes d'optimisation des réseaux de neurones.

1.2.2.5 Recherche d'hyperparamètres

La performance d'un algorithme d'apprentissage en général dépend de ses hyperparamètres. La recherche d'hyperparamètres manuelle, par essai-erreur pose des problèmes de reproductibilité et de passage à l'echelle. La méthode historique fondée sur la recherche des hyperparamètres selon une grille de valeurs croît de manière exponentielle avec le nombre d'hyperparamètres et chaque expérience peut être coûteuse en temps de calcul. Cette recherche par grille peut mener à de médiocres performances comparativement à une recherche aléatoire (random search) (Bergstra and Bengio, 2012), où chaque valeur est tiré indépendamment suivant une distribution. En effet, prenons en exemple un cas d'optimisation lorsque seulement certains hyperparamètres sont cruciaux pour le modèle (comme le pas d'apprentissage) et que d'autres n'ont pas d'influence (comme parfois l'ajout de Dropout)

- Dans le cas de la recherche en grille, chaque hyperparamètre est traité avec la même intensité. En particulier, à un pas d'apprentissage fixé, toutes les valeurs de Dropout sont testées (inutilement) ;
- contrairement à la recherche aléatoire, lors de chaque nouvel essai, une valeur différente pour le pas d'apprentissage et pour la valeur de Dropout sont essayées. Ainsi, un hyperparamètre non influent ne pénalise pas la recherche aléatoire.

1. Ce fut l'une des plus grandes avancées sur le score de classification de la base de données d'ImageNet

De plus, une recherche aléatoire explore une plus grande diversité de valeurs dans chaque dimension définie par les hyperparamètres, et dispose ainsi de plus de liberté pour trouver de meilleures régions de l'espace.

Cependant, la recherche purement aléatoire peut être "malchanceuse" et ne tirer qu'un ensemble de valeurs laissant certaines régions inexplorées. Ainsi, des méthodes quasi-aléatoires telles que des séquences de discordance faibles (Low Discrepancy Sequences) sont proposées pour remédier à ce problème (Bousquet et al., 2017). Notons également qu'il existe une recherche active autour de la recherche automatique d'hyperparamètres (Kotthoff et al., 2017; Guyon et al., 2015).

1.2.2.6 Surapprentissage

Le surapprentissage (overfitting) intervient lorsque la complexité du modèle (typiquement son nombre de couches ; ou le nombre de neurones dans la couche intermédiaire) est trop élevée par rapport aux données disponibles (Vapnik, 1998). Le signe du surapprentissage est que les performances sur les données d'apprentissage sont bien meilleures que les performances en test. Plusieurs méthodes sont mises en place pour éviter ce phénomène ; les plus populaires sont les suivantes :

— réduire le nombre de paramètres du modèle ;
— ajouter un terme de régularisation à la fonction objectif, e.g. par l'ajout d'une pénalité de type L1 ou L2 sur les paramètres du modèle (nommé régularisation "ridge" (Hoerl and Kennard, 1970) ou "lasso" (Tibshirani, 1996) dans le cas de la régression linéaire) ;
— la validation croisée (Kohavi et al., 1995) ;
— arrêter l'apprentissage avant la convergence (early stopping) si l'erreur de généralisation augmente.

Et spécifiquement dans le cas des réseaux de neurones :

— masquer aléatoirement l'activation de certains neurones pendant l'entraînement (dropout) ou masquer certaines connections (dropconnect) (Wan et al., 2013) ;
— la normalisation par batch.

1.2.2.7 Implémentation

Le gain récent de popularité des réseaux de neurones a mené lors des dernières années aux développement de nombreuses bibliothèques logicielles. La communauté d'utilisateurs enrichit continuellement cette base. Parmi les bibliothèques les plus populaires, figurent TensorFlow (Abadi et al., 2016), Keras (Chollet et al., 2015), Torch (Collobert et al., 2011a) (ainsi que sa version python PyTorch (Paszke et al., 2017)), CNTK (Seide and Agarwal, 2016) et Theano (Theano Development Team, 2016). Développées par les acteurs académique et industrielles, ces bibliothèques sont utilisées par tous.

Permettant de prototyper des modèles rapidement, les opérations bas niveaux, telles que les produits matriciels ou les calculs de gradients sont aisément accessibles et manipulables. Par ailleurs, ces bibliothèque implémentent rapidement les avancées de l'état de l'art.

Chapitre 2

Traitement des langues naturelles & Représentations

Ce document concerne le filtrage collaboratif dans le domaine de l'emploi, où les objets (offres d'emploi) et les utilisateurs sont décrits par des documents. Ce chapitre se focalise ainsi sur la représentation de textes en langage naturel pour le filtrage collaboratif, à l'exclusion des nombreuses autres applications du traitement des langues naturelles [1]. Ce chapitre est en particulier consacré à la représentation continue des documents en langage naturel, plongeant un texte dans $\mathbb{R}^d$ où d est un hyper-paramètre spécifiant la taille de la représentation (usuellement entre 50 et 300). Un tel plongement dans un espace métrique définit de fait une distance entre documents. Cela permet l'utilisation de système de recommandation à base de mémoire, décrit au chapitre suivant (Section 3).

Après avoir décrit les prétraitements génériques des documents (Section 2.1), ce chapitre détaille plusieurs méthodes de représentations non-supervisées de documents en distinguant deux familles de méthodes : les approches fondées sur les sacs de mots (Section 2.2) et celles fondées sur les modèles de langue (Section 2.3). Ce chapitre s'achève sur les méthodes de réduction de dimensionalité et de visualisation (Section 2.4). Dans la suite, n_d désigne le nombre de documents du corpus considéré.

1. Citons en particulier les approches :
   - Se fondant sur des modèles de langue :
     - La reconnaissance de la parole (Mikolov and Zweig, 2012) ;
     - L'annotation de mots (part of speech tagging) (Ling et al., 2015) ;
     - La traduction automatique (Schwenk et al., 2012) ;
     - Le résumé de texte (Rush et al., 2015) ;
     - La classification de textes (Zhang et al., 2015; Kim et al., 2016; Kim, 2014).
   - Se fondant sur une représentation non supervisée :
     - La détection de sujets (topic detection) (Blei et al., 2003) ;
     - La recherche d'information et de documents (Manning, 1995) ;
     - L'analyse de sentiments (Le and Mikolov, 2014) ;
   - Se fondant sur une représentation apprise pour la tâche :
     - La classification de texte (Joulin et al., 2016) ;
     - L'implication textuelle (Recognizing Textual Entailment) (Parikh et al., 2016) ;
     - La traduction automatique (AP et al., 2014)

	Sac de mots	Modèle de langue
Prétraitements	Nettoyage du texte. Retirer mots vides et peu fréquents	
Sélection des termes	Mot ou n-gram	
Représentation vectorielle	tf-idf	Représentation des termes
Projection dans un espace latent	LSA, LDA, SDAE	Word2Vec, RNN
Similarité	Cosinus	Cosinus

TABLE 2.1 – Chaîne de traitement pour la représentation d'un document texte

2.1 Prétraitements usuels

Après l'acquisition des données, la première phase de traitement consiste à nettoyer le texte. Cette phase est illustrée sur un extrait d'une offre d'emploi de Qapa :

```
Nous recrutons pour <strong>20 postes de serveurs en restaurant (H/F)</strong> situés dans la région de Blois.
```

— Nettoyer les balises : Enlever les caractères spéciaux, non reconnus et les balises html.

```
Nous recrutons pour 20 postes de serveurs en restaurant (H/F) situés dans la région de Blois.
```

— Tokenisation : segmentation des données en unités signifiantes : les *termes*, parfois appelé «tokens». Les mots sont les chaînes de caractères séparées par les espaces et les signes de ponctuations. Un terme est défini comme un mot ou une suite de mots consécutifs. Plus généralement, n-gram désigne un terme formé de n mots consécutifs. Dans la suite de ce manuscrit seuls les mots (1-grams ou unigram) sont considérés.

```
['Nous', 'recrutons 'pour', '20', 'postes', 'de', 'serveurs', 'en', 'restaurant', '(H/F)', 'situés', 'dans', 'la', 'région', 'de', 'Blois', '.']
```

— Lemmatisation : Normalisation des termes afin de faire abstraction des variations de flexion (essentiellement liées à la conjugaison, aux majuscules et à la déclinaison). Nous conservons le masculin singulier pour un adjectif et l'infinitif pour un verbe conjugué. La lemmatisation permet d'unifier les différentes flexions d'un même mot (au risque d'unifier des mots indépendants, par exemple le verbe conjugué "joué" et le nom "joue"). La lemmatisation occasionne également une perte d'information ; par exemple des mots peuvent avoir des sens différents selon qu'ils sont au singulier ou au pluriel [2].

```
nous recruter pour 20 poste de serveur en restaurant (H/F) situer dans la region de blois
```

— Mots vides : (stop words) filtrage des termes ne contenant pas d'information sémantique (e.g. "de", "et", "ou", "mais", "on", "pour") ; la liste de ces mots-outils est déterminée manuellement pour chaque langue. L'élagage des termes les plus fréquents (termes présents sur l'ensemble des textes et donc sans impact pour leur caractérisation) est également effectué ; le seul de fréquence est un hyper-paramètre de l'approche.

2. Par exemple dans une phrase du type "vous devez bien connaître la région" et "vous devez bien connaître les régions"

```
recruter 20 poste serveur restaurant (H/F) situer region blois
```

Le pré-traitement des documents en langue naturelle utilise souvent des méthodes telles que i) l'étiquetage morpho-syntaxique (part-of-speech tagging) associant à chaque terme sa catégorie grammaticale (nom, verbe, déterminant) dans l'énoncé (Valli and Véronis, 1999) ou ii) l'identification des relations ou dépendances syntaxiques des termes dans la phrase (sujet, objet, complément). Ces deux méthodes ne sont pas considérées dans la suite du manuscrit, car les phrases des offres et des CVs sont souvent courtes, avec une grammaire pauvre (Table C.1).

2.2 Représentation vectorielle - Sac de mots

La phase suivant la phase de nettoyage dans l'approche présentée associe une représentation vectorielle à une suite de termes. Deux approches sont distinguées. La première considère des sacs de mots (ou sacs de termes), où l'ordre des termes n'est pas pris en compte. La deuxième approche, que nous décrivons avec les modèles de langue (Section 2.3), garde l'information séquentielle des phrases. Dans l'approche "sac de mots" un ensemble de n_d documents, appelé *corpus*, est représenté par une matrice "documents $\times$ termes" notée $\mathbf{D}$, où une ligne correspond à un document et une colonne à un terme [3]. La cellule i, j de cette matrice représente le nombre d'occurrences du terme j dans le document i (ou bien une valeur binaire de présence du terme dans le document). Cette matrice est très clairsemée. Ainsi dans le contexte de Qapa [4] la fraction des cellules non nulles est de .8 % .

2.2.1 Sac de mots

Dans la suite, $\mathbf{D_i} \in \mathbb{R}^{n_\mathcal{V}}$ dénote la i-ème ligne de la matrice $\mathbf{D}$, représentant le i-ème document du corpus.

2.2.2 tf-idf

Les termes apparaissant dans la matrice $\mathbf{D}$ sont pondérés pour faire ressortir les termes rares et diminuer l'importance des termes trop fréquents. La pondération la plus utilisée est appelée tf-idf pour "Term Frequency, Inverse Document Frequency" (la fréquence des termes fois l'inverse de la fréquence dans les documents).

$$\text{tf-idf}(t) = tf(t) \cdot idf(t)$$

$\text{tf}(t)$ représente la fréquence du terme t dans le document courant. L'approche présentée considère une version lissée [5] de la pondération idf (Manning et al., 2008).

$$idf(t) = log\frac{1 + n_d}{1 + df(t)} + 1$$

Où $df(t)$ est le nombre de documents contenant le terme t.

3. Notons que dans la communauté de traitement automatique des langues naturelles (TALN) (Levy et al., 2015; Pennington et al., 2014; Manning et al., 2008) la convention inverse est utilisée.

4. Composé de 56 000 documents et 10 000 mots

5. Correspondant à la valeur par défaut de scikit-learn (Pedregosa et al., 2011) définie pour des raisons de qualité empirique.

La matrice ainsi pondérée notée $\mathbf{D}$ constitue une première représentation du corpus, sur laquelle la similarité la plus utilisée est la similarité *cosinus*. Un des intérêts de la similarité cosinus, variant dans $[-1, 1]$, est d'être indépendante de la taille des vecteurs (par opposition à leur produit scalaire).

La similarité cosinus permet de définir la *distance* cosinus :

$$d_{cos}(\boldsymbol{u}, \boldsymbol{v}) = 2 \cdot (1 - sim_{cos}(\boldsymbol{u}, \boldsymbol{v}))$$

Notons que dans le cas où les vecteurs $\boldsymbol{u}, \boldsymbol{v}$ sont de norme 1, la distance cosinus correspond à la distance euclidienne au carré (i.e. $\|\boldsymbol{u}\|_2 = \|\boldsymbol{v}\|_2 = 1 \Rightarrow d_{cos}(\boldsymbol{u}, \boldsymbol{v}) = \|\boldsymbol{u} - \boldsymbol{v}\|_2^2$).

La limite de la similarité cosinus dans l'espace tf-idf est de ne pas tenir compte des phénomènes de synonymie : si deux concepts voisins sont exprimés avec des termes différents, leur similarité cosinus est nulle.

2.2.3 Projection LSA

Une réponse à cette limitation consiste à effectuer une réduction de dimension de l'espace du vocabulaire. L'une des approches les plus étudiées consiste à factoriser la matrice $\mathbf{D}$ selon une décomposition en vecteurs singuliers (SVD). Cette approche proposée par Deerwester et al. (1990) [6], est appelée analyse sémantique latente (LSA, *Latent Semantic Analysis* ou LSI, *Latent Semantic Indexing*) ; elle détermine une représention dense de faible dimension pour les documents. L'hypothèse sous-jacente de LSA est que la dimensionalité "intrinsèque" des documents est faible (quelques centaines) comparée à la taille du vocabulaire (en dizaines de milliers). D'autre part, ainsi que mentionné ci-dessus, le langage naturel souffre des deux phénomènes de synonymie (différents termes peuvent avoir le même sens) et de polysémie (un même terme peut avoir deux sens différents). La LSA, compressant les documents dans un espace de taille réduite, permet de remédier au premier phénomène : si deux termes peuvent être utilisés de manière similaire, et donc avec les mêmes contextes, la réduction de dimensionalité les projettera tous deux sur des vecteurs proches. Cette réduction de dimensionalité a également pour effet de définir une représentation dense (non clair-semée) des termes.

SVD Formellement, la matrice $\mathbf{D} \in \mathbb{R}^{n \times n_\mathcal{V}}$ est factorisée par SVD en produit de trois matrices $U \in \mathbb{R}^{n_d \times n_d}, \Sigma \in \mathbb{R}^{n_d \times n_\mathcal{V}}, V^T \in \mathbb{R}^{n_\mathcal{V} \times n_\mathcal{V}}$.

$$\mathbf{D} = U \cdot \Sigma \cdot V^T \tag{2.1}$$

Σ est une matrice diagonale ; les valeurs singulières sont positives et ordonnées par valeur décroissante sur la première diagonale. Les colonnes de U et de V sont orthogonales. La représentation des documents donnée par la matrice U correspond à un changement de base des documents (lignes de $\mathbf{D}$), projetant chaque document sur les vecteurs singuliers de la décomposition.

La SVD est en général suivie d'une phase de troncature, annulant les valeurs singulières les moins élevées. Cette troncature définit une approximation de $\mathbf{D}$ données par

$$\hat{\mathbf{D}} = U_{|k} \Sigma_{|k} V_{|k}^T$$

où $U_{|k}$ (respectivement $V_{|k}$) est la matrice ne gardant que les k premières colonnes de U (resp., de V), de taille $n \times k$ (resp., $m \times k$) et $\Sigma_{|k}$ est la matrice diagonale de taille $k \times k$ définie par les k plus

6. voir (Manning et al., 2008) pour une présentation détaillée

grandes valeurs singulières de Σ. $\hat{\mathbf{D}}$ correspond à l'approximation optimale de rang k de la matrice $\mathbf{D}$ pour la norme de Frobenius $\|,\|_F$ avec

$$\|M\|_F = \sqrt{\sum_{i,j} M_{i,j}^2}$$

Il s'agit donc de l'erreur au sens des moindres carrés sur l'ensemble des coefficients de la matrice. En résumé,

$$\min_{Z|rang(Z)=k} \|M - Z\|_F = \left\|M - U_k \cdot \Sigma_k \cdot (V_k^T)\right\|_F$$

avec U_k et V_k données par les k premières colonnes de U et V.

La représentation continue des documents (respectivement celle des termes) est obtenue en intégrant les valeurs singulières à la matrice U (resp., V). Notons $\tilde{U}$ la représentation des documents dans l'espace $\mathbb{R}^k$: $\tilde{U} = U_{|k} \cdot \Sigma_{|k}^p$ et $\tilde{V}$ la représentation des termes $\tilde{V} = V_{|k} \cdot \Sigma_k^p$. Le paramètre p sera fixé à .5 : ainsi que l'ont montré (Levy et al., 2015) dans le contexte de l'identification sémantique, ce choix $p = 0.5$ donne les meilleurs résultats. Dans la suite, la représentation latente de dimension k d'un document i est définie par $\tilde{\boldsymbol{U}}_{\boldsymbol{i}} \in \mathbb{R}^k$. Cette représentation est une combinaison linéaire sur l'espace du vocabulaire, avec

$$\tilde{U} = \mathbf{D} \cdot V_{|k} \cdot \Sigma_{|k}^{1/2} \tag{2.2}$$

De même, la représentation latente de dimension k du terme j est donnée par le vecteur $\tilde{\boldsymbol{V}}_{\boldsymbol{j}}$. La similarité utilisée dans l'espace latent LSA est la similarité cosinus.

Auto-Encodeur Un Auto-Encodeur (Section 1.1.2) peut également être utilisé pour compresser la représentation sac de mots des documents dans un espace de faible dimension. Dans ce cas, l'entrée du réseau est le vecteur tf-idf et la couche caché donne la représentation du document. Dans le cas où l'Auto-Encodeur dispose d'une seule couche caché et sans non-linéarité, la représentation minimisant l'erreur sera la représentation LSA. Dans le cas plus profond avec des non-linéarités, l'Auto-Encodeur permet une plus grande capacité de compression que la SVD.

2.2.4 LDA

Une autre représentation continue des documents se fonde sur l'Allocation de Dirichlet Latente (LDA, *Latent Dirichlet Allocation*). Cette méthode introduite par Blei et al. en 2003 (Blei et al., 2003) est certainement l'une des plus populaires et les plus citées par la communauté de l'apprentissage automatique. LDA est un modèle génératif probabiliste, utilisé pour découvrir les thèmes latents (*topics*, aussi appelés «sujets» ou «concepts» abstraits) d'une collection de documents. Le modèle sous-jacent considère qu'un document est une mixture de thèmes latents, chaque thème correspondant à une distribution sur les mots du vocabulaire. Nous illustrons cette méthode sur les données Qapa (Section C). La représentation $\boldsymbol{\theta_d}$ d'un document d est définie par les probabilités de présence des k thèmes (où k est un hyper-paramètre) du modèle : $\boldsymbol{\theta_d} = (\theta_d^1, \ldots, \theta_d^k) \in \mathbb{R}^k$, avec $\sum_i \theta_d^i = 1$.

Un thème z est défini par une distribution $\boldsymbol{\beta_z}$ sur les mots du vocabulaire $\mathcal{V}$: $\boldsymbol{\beta_z} = (\beta_z^1, \ldots, \beta_z^{n_\mathcal{V}}) \in \mathbb{R}^{n_\mathcal{V}}$. Le modèle génératif associé à une représentation $\boldsymbol{\theta_d}$ et aux distributions $\boldsymbol{\beta_z}$ est défini de la manière suivante. Le nombre de mots N est tiré selon une loi de Poisson, puis pour chaque mot :

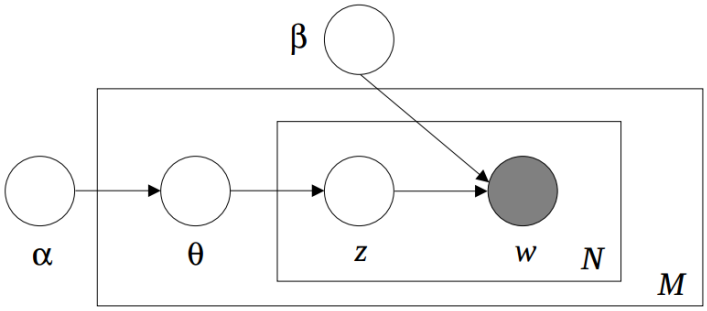


FIGURE 2.1 – Modèle graphique de l'Allocation de Dirichlet Latente

— Un thème z est tiré selon la loi multinomiale définie par $\boldsymbol{\theta_d}$:

$$z \sim Multinomial(\boldsymbol{\theta_d})$$

— Conditionnellement à z, le mot courant w est tiré selon la distribution $\boldsymbol{\beta_z}$.

$$w \sim Multinomial(\boldsymbol{\beta_z})$$

Conditionnellement au modèle $\boldsymbol{\theta_d}$ et aux distributions $\boldsymbol{\beta} = (\boldsymbol{\beta_z}, z \in \{1, \ldots, k\})$, la vraisemblance d'un document ainsi généré est donnée par :

$$\mathbb{P}(w|\boldsymbol{\theta_d}, \boldsymbol{\beta}) = \prod_w \mathbb{P}(z|\boldsymbol{\theta}) \cdot \mathbb{P}(w|z, \boldsymbol{\beta})$$

Les paramètres $(\boldsymbol{\theta}, \boldsymbol{\beta})$ du modèle LDA sont optimisés par un algorithme de type Expectation-Maximization (Dempster et al., 1977), pour maximiser la (log) vraisemblance du corpus considéré. Le prior de $\boldsymbol{\beta}$ est une loi de Dirichlet de paramètre $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k)$, pris par défaut à $(\frac{1}{k}, \ldots, \frac{1}{k})$, permettant un petit nombre de sujets par document.

La similarité entre deux document d_1 et d_2 est donnée par la similarité cosinus entre leur représentation $\boldsymbol{\theta_1}$ et $\boldsymbol{\theta_2}$. Notons que cette similarité ne prend pas en compte la similarité entre les thèmes (topics).

2.3 Modèle de langue

Un modèle de langue est une distribution de probabilité sur une séquence de symboles. C'est une approche alternative aux sacs de mots, prenant en compte l'ordre des termes. Un modèle de langue permet d'évaluer la vraisemblance d'une phrase. Il peut également être utilisé pour générer une séquence de mots, ou le mot suivant dans une phrase [7]. Les premiers modèles de langues sont basés sur les statistiques des n-gram (Good, 1953; Witten and Bell, 1991; Kneser and Ney, 1995). Cependant ces modèles ne génèrent pas de représentations vectorielles ni pour les mots ni pour les documents. Un premier modèle continu de langue proposé en 2003 se fonde sur les réseaux de

7. Certains modèles de langue considèrent les caractères individuellement, le vocabulaire est alors l'ensemble des caractères plutôt que l'ensemble des termes.

neurones (Bengio et al., 2003). Cette approche génère une représentation vectorielle pour chaque mot. Les modèles de langue récents mettent l'accent sur la représentation des mots (Mikolov et al., 2013a). Ces représentations des mots sont ensuite évaluées sur la tâche visée, ou utilisées pour une représentation continue des documents (Section 2.3.1.1). Notons que certaines approches définissent directement une représentation continue des documents (Section 2.3.2).

Une évaluation classique de la qualité d'un modèle de langue se fonde sur le corpus «One Billion Words» (Chelba et al., 2013). Dans ce cas, les modèles doivent retrouver les mots retirés aléatoirement dans le corpus. L'approche par statistique sur les n-gram (Kneser and Ney, 1995) reste une bonne mesure de référence, mais est dépassée par les approches neuronales (Bengio et al., 2003).

2.3.1 Représentation des mots (words embedding)

La première représentation des mots, dite «one hot encoding», code un mot par un vecteur binaire, possédant une seule valeur, donnant l'index du mot dans le vocabulaire. Notons que cette représentation n'a aucune propriété sémantique (tous les mots sont à la même distance les uns des autres). La matrice $\mathbf{D}$ induit une représentation des mots où chaque mot est un point dans l'espace des documents. Le vecteur de représentation code ainsi le sous-ensemble des documents contenant le mot (*i.e.* le vecteur $\mathbf{D}_{.,i}$ représente le mot i).

PPMI Une version plus fine de cette approche considère la matrice termes $\times$ *contexte*, le contexte étant la liste des mots apparaissant dans une fenêtre autour du mot cible. Cette méthode, nommée PMI (*pointwise mutual information*) est introduite en 1990 (Church and Hanks, 1990) et améliorée sous le nom de PPMI (*positive point wise mutual information*) en affinant la valeur pour les termes n'apparaissant pas dans l'ensemble d'entraînement. Cette matrice donne ainsi une représentation d'un mot dans l'espace des contextes. La factorisation de cette matrice donne une représentation de plus faible dimension, plus dense et généralisant mieux, étant ainsi de meilleure qualité pour les tâches sémantiques [8] (Levy and Goldberg, 2014; Levy et al., 2015).

Word2Vec La représentation vectorielle des mots est également devenue très populaire depuis les travaux de Mikolov et al. en 2013, proposant un modèle de langue neuronal nommé Word2Vec (Mikolov et al., 2013a,0), obtenant de bons scores dans les tâches de similarités sémantiques. Cette approche est étendue à la représentation des documents, nommé Doc2Vec (Le and Mikolov, 2014; Dai et al., 2015), décrite (Section 2.3.2).

Citons également **GloVe** (Pennington et al., 2014) (pour global vector representation) qui génère une représentation vectorielle des termes de façon non-supervisée (sans toutefois entraîner de modèle de langue).

Ces trois approches, PPMI, Word2Vec et GloVe projetant les mots dans $\mathbb{R}^k$ sont comparées sur la tâche d'identification de synonymes (Levy et al., 2015), en considérant toujours la similarité cosinus. Les différences de performance entre ces trois méthodes sont souvent moins importantes que les différences de performance dues au choix des hyper-paramètres d'apprentissage de ces méthodes.

8. Ces tâches sont arbitraires et construites pour cette occasion (Mikolov et al., 2013a) ; elles sont du type : "Roi - homme + femme = ?"

2.3.1.1 Des mots aux documents

Plusieurs méthodes sont proposées pour construire une représentation des documents (ou directement une similarité entre documents) à partir d'une représentation des mots. Notons que l'approche naïve (en prenant la moyenne des représentations, le max sur chaque dimension ou le centre par une approche de $K-$moyenne du sac de mots) n'apporte pas d'information sur le document ; elle est moins informative que tf-idf pour la classification de documents (Zhang et al., 2015).

Word Mover Distance Une autre approche appelée Word Mover Distance (WMD) (Kusner et al., 2015) est inspirée de l'approche Earth Mover Distance (Rubner et al., 1998). La distance entre deux phrases est définie comme le coût de transport pour passer d'une phrase à l'autre, pondéré par la distance euclidienne dans l'espace de représentation des mots (par ex. Word2Vec). Cependant, cette méthode est inopérante si les phrases sont de tailles trop dissemblables. De plus, évaluée sur de petits jeux de données [9], cette approche n'obtient pas de résultats significativement meilleurs que la similarité LSA.

Réseaux de neurones convolutionels Appelés 1D-CNN (par opposition aux (2D-)CNN, dont la couche de convolution est en dimension 2 pour traiter les images), ces réseaux représentent une phrase par la concaténation des représentations de ses mots, formant un vecteur de taille $k * n_w$, avec k la taille de l'encodage des mots, n_w le nombre de mots. Sur ces phrases est entraîné un réseau convolutionnel (la taille de la fenêtre glissante est un multiple de k) pour classer des phrases (Kim, 2014; Kim et al., 2016). Ces modèles bénéficient d'une initialisation de la représentation des mots suivant une approche non supervisée telle que Word2Vec.

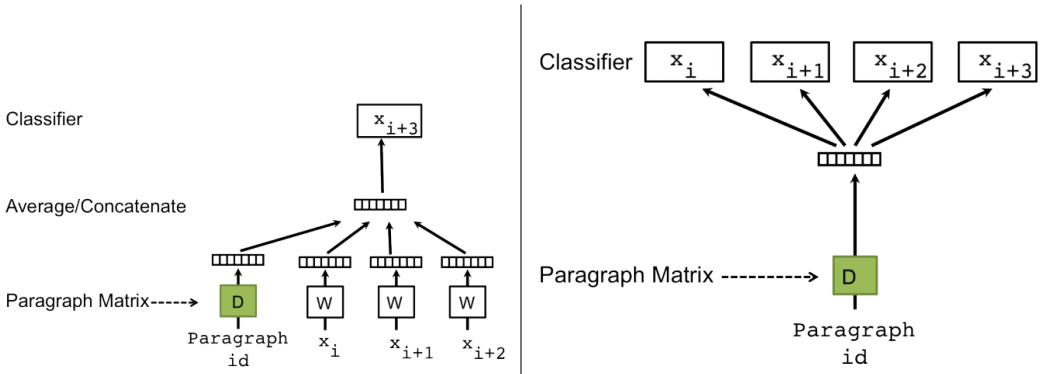
(Zhang et al., 2015; Conneau et al., 2017) considèrent un réseau de convolution travaillant au niveau des caractères pour la classification de texte. De façon intéressante, le meilleur algorithme diffère suivant la taille du jeu de données : tf-idf est meilleur pour de petits jeux de données et les 1D-CNN améliorent les performances pour des données de plus de 500 000 exemples. Ce résultat confirme le "No free lunch" theorem (Wolpert and Macready, 1997) : le choix du meilleur algorithme dépend du contexte. L'approche «sac de représentations» obtient des résultats toujours inférieurs à une approche tf-idf classique. Notons cependant que l'algorithme fastText (Joulin et al., 2016) qui construit un «sac de représentations» supervisé pour classer des textes, obtient de bons résultats avec un temps de calcul de l'ordre de la minute, alors que les entraînements des 1D-CNN sont de l'ordre de l'heure ou de la journée. Cependant, la représentation des mots issus de fastText étant générée de façon spécifique à la tâche, elle n'est pas généralisable.

Les limitations des réseaux profonds sont :

— Le gradient disparaît dans les couches trop profondes, d'où une difficulté à les entraîner ;
— De nombreux hyper-paramètres sont à calibrer et l'entraînement est également sensible à la taille des batchs et au pas d'apprentissage ;
— La durée d'entraînement peut se compter en jours (spécialement pour les réseaux récurrents) ;
— Il est nécessaire de disposer d'un grand corpus d'apprentissage (Zhang et al., 2015).

2.3.2 Au delà des modèles de langues

9. De l'ordre du millier de documents, avec une tâche de plus proches voisins.



(a) Mémoire distribué (b) Sac de mots distribué

FIGURE 2.2 – Deux modèles de Doc2Vec. $\boldsymbol{x_i}$ correspond à l'encodage sous forme «one hot» du mot i. Figures issue de (Dai et al., 2015)

Document2Vec Une autre méthode pour représenter un document, induisant une représentation continue des mots, est une extension de Word2Vec appelé Doc2Vec (Le and Mikolov, 2014; Dai et al., 2015). La différence entre Word2Vec et Doc2Vec concerne la prise en compte en entrée de l'indice du document ou du paragraphe (Le and Mikolov, 2014; Dai et al., 2015).

Ce modèle maintient deux matrices de représentations, appelées tables de correspondances (look-up tables) : $\mathbb{W}$ pour les mots et $\mathbb{D}$ pour les documents. Nous avons donc : $\mathbb{W} \in \mathbb{R}^{n_\mathcal{V} \times k}$ et $\mathbb{D} \in \mathbb{R}^{n_d \times k}$, avec n_d le nombre de documents et k la dimension de l'espace de représentation. Ainsi $\mathbb{D}_{\boldsymbol{i}}$ (respectivement $\mathbb{W}_{\boldsymbol{i}}$) donne la représentation du document (resp. du mot) i.

Deux méthodes sont employées pour entraîner ces représentations :

— **Doc2Vec : Mémoire distribuée** est inspiré de l'approche "Skip-gram" décrite dans (Mikolov et al., 2013a). En entrée figure la représentation du document courant ($\mathbb{D}_{\boldsymbol{paragraph\ id}}$) [10] ainsi que les représentations des mots du contexte ($\mathbb{W}_{\boldsymbol{i}}, \mathbb{W}_{\boldsymbol{i+1}}, \mathbb{W}_{\boldsymbol{i+2}}$). Comme suggéré (Figure 2.2a), le vecteur du paragraphe est concaténé ou moyenné avec le contexte local des vecteurs de mots pour obtenir une représentation latente h [11]. Cette représentation latente sert ensuite d'entrée à un classificateur linéaire pour prédire le mot suivant de la phrase. La prédiction est un softmax sur l'ensemble des mots possibles [12]. Cependant, comme l'ensemble des mots possibles est de trop grande dimension, des techniques sont adoptées pour réduire la complexité, par exemple la sortie du réseau est codée sous forme d'un arbre [13], permettant d'accéder plus rapidement aux mots fréquents. Lors de l'entraînement, les matrices $\mathbb{D}$ et $\mathbb{W}$, ainsi que les poids du softmax sont mises à jour par back-propagation.

— **Doc2Vec : Sac de mots distribués** Dans le cas dit du "sac de mots distribués", l'entrée se limite à la matrice $\mathbb{D}$. Cette représentation des documents est utilisée par un classificteur linéaire entraîné pour prédire l'ensemble de mots du paragraphe (sans ordre) (Figure 2.2b). Ce modèle est plus léger en mémoire car il n'utilise pas la matrice $\mathbb{W}$. Pour les documents

10. Le vecteur associé au document a pour but de fournir l'information complémentaire au contexte.
11. Dans le cas moyenné $h = \frac{1}{4}(\mathbb{D}_j + \mathbb{W}_{\boldsymbol{i}} + \mathbb{W}_{\boldsymbol{i+1}} + \mathbb{W}_{\boldsymbol{i+2}})$ où j est l'indice du document.
12. Notons que dans (Mikolov et al., 2013b), la probabilité d'un mot est le produit scalaire entre sa représentation et la représentation intermédiaire.
13. Codage de (Huffman, 1952).

de l'ensemble de test, la représentation du document est apprise par back-propagation. Ce modèle est dit plus efficace que celui à mémoire distribuée (Dai et al., 2015).

Ces deux modèles mettent donc à jour la matrice $\mathbb{D}$ qui représente les documents.

Réseaux récurrents Prenant en compte l'aspect séquentiel de la phrase, les réseaux de neurones récurrents permettent une représentation de la phrase entière, essayant de prendre en compte les dépendances éloignées (entre le début et la fin de phrase par exemple). Pouvant éventuellement utiliser une représentation des mots pre-entraînée, ces modèles affinent de façon non supervisée la représentation de la phrase. Une couche de neurones récurrents permet de prendre en compte des dépendances à plus long terme, avec des poids dit "de mémoire". L'architecture ainsi définie est appelée LSTM (Hochreiter and Schmidhuber, 1997) pour "long short term memory", mémoire à court et long terme. Les réseaux récurrents obtiennent de bons résultats lorsqu'ils sont combinés avec d'autres modèles sur les N-gram, mais ils sont jugés plus faibles lorsqu'ils sont considérés isolément (Jozefowicz et al., 2016). Les réseaux récurrents sont meilleurs que les 1D-CNN dans le cas où des dépendances de long terme sont importantes. Cependant les 1D-CNN sont meilleurs pour classer les phrases contenant des termes clefs (par exemple dans le cas d'analyse de sentiment) (Yin et al., 2017). Finalement, les réseaux récurrents souffrent des mêmes difficultés que les réseaux 1D-CNN (la dépendance temporelle rend l'apprentissage délicat) ; l'entraînement requiert une très grande expertise pour donner de bonnes performances (Yin et al., 2017).

2.4 Réduction de dimensionalité

Cette section décrit brièvement quelques approches de réduction de la dimensionalité des données, dont une utilisation principale est la visualisation des données. La réduction de dimension consiste à plonger des données de $\mathbb{R}^D$ (avec D de l'ordre de la centaine ou du millier, e.g., taille d'un vocabulaire ou nombre de pixels d'une image) dans un espace $\mathbb{R}^d$ avec $d << D$, et $d = 2$ ou 3 lorsqu'il s'agit de visualiser les données. Ce plongement doit respecter certaines propriétés de l'espace initial, par exemple les distances entre paires de points, ou les voisinages locaux. Le passage de D à d dimensions entraîne en toute généralité une perte d'information. Cependant, si l'on fait l'hypothèse que la dimension intrinsèque des données est de l'ordre de d, la réduction de dimensionalité permet de recouvrer une représentation pertinente, celle du sous-espace vectoriel ou de la variété dans laquelle habitent les données.

Une des premières approches de réduction de dimension est le positionnement multidimensionnel (*Multi-dimensional Scaling*, MDS) (Kruskal, 1964), cherchant une représentation sous contrainte de préserver les distances initiales entre paires de points. Notant $\boldsymbol{x_i}, \boldsymbol{x_j} \in \mathbb{R}^D$ les points de l'espace initial et $\boldsymbol{y_i}, \boldsymbol{y_j} \in \mathbb{R}^d$ les points de la représentation dans l'espace réduit, le MDS cherche $\{\boldsymbol{y_i}\}$ qui minimise un critère tel que

$$\sum_{i \neq j} (\|\boldsymbol{x_i} - \boldsymbol{x_j}\|_2 - \|\boldsymbol{y_i} - \boldsymbol{y_j}\|_2)^2$$

Une autre méthode nommée ISOMAP (Tenenbaum et al., 2000a) peut être vue comme une extension du positionnement multidimensionnel, où la distance $d(\boldsymbol{x}, \boldsymbol{x'})$ entre deux points est définie comme la distance du plus court chemin $\boldsymbol{x} = \boldsymbol{x_0}, \boldsymbol{x_1}, \ldots, \boldsymbol{x_K} = \boldsymbol{x'}$ et la distance euclidienne entre deux points successifs $\boldsymbol{x_i}, \boldsymbol{x_{i+1}}$ étant inférieure à un ϵ prescrit (hyper-paramètre de l'algorithme). L'une des limitations de MDS et de ISOMAP est de chercher à préserver aussi les distances entre points éloignés ; cependant, dans beaucoup de contextes seuls les voisinages locaux sont pertinents.

Deux méthodes levant la limitation ci-dessus, et cherchant essentiellement à conserver la structure locale des données initiales sont la méthode t-SNE (*t-distributed stochastic neighbor embedding*), et la méthode LLE (*Locally Linear Embedding*), dont s'inspire notre première contribution et qui sera décrite en section Section A.

2.4.1 t-SNE

L'algorithme t-SNE (Maaten and Hinton, 2008) (*t-distributed stochastic neighbor embedding*) cherche une représentation fidèle des voisinages initiaux, au sens probabiliste et par rapport à des distributions de t-Student). Cette méthode est non-linéaire, non supervisée et non-paramétrique ; elle est souvent utilisée pour visualiser l'évolution de réseaux de neurones (Olah, 2015). Il existe également une implémentation paramétrique de t-SNE (van der Maaten, 2009) ; cependant celle-ci n'a pas été considérée dans la suite.

Formellement, le but est toujours d'associer aux points $X = (\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}), \boldsymbol{x_i} \in \mathbb{R}^D$ de l'espace initial un ensemble de points $Y = (\boldsymbol{y_1}, \ldots, \boldsymbol{y_n}), \boldsymbol{y_i} \in \mathbb{R}^d$. Soit $P_{i,j}$ la probabilité des points $\boldsymbol{x_i}$ et $\boldsymbol{x_j}$ d'être proches dans $\mathbb{R}^D$, définie par

$$P_{j|i} = \frac{1}{Z_i} e^{-\frac{1}{2\sigma_i^2}\|\boldsymbol{x_i} - \boldsymbol{x_j}\|_2^2}$$

où Z_i est une constante de normalisation assurant que $P_{|i}$ somme à 1. $P_{i,j}$ est alors définie comme la demi-somme de $P_{j|i}$ et $P_{j|i}$; cette symétrisation vise à éviter des observations aberrantes (Maaten and Hinton, 2008). Le paramètre $\boldsymbol{\sigma}$ permet de définir la taille du voisinage, en contrôlant le compromis entre les voisinages locaux et globaux des données. Ce paramètre $\boldsymbol{\sigma}$ est optimisé par une recherche binaire à partir du paramètre de *perplexité* en argument de l'algorithme, correspondant aux nombres de voisins à prendre en compte. Pour une étude approfondie et visuelle de l'influence du paramètre de perplexité, le lecteur intéressé pourra se référer à l'excellent post (Wattenberg et al., 2016).

De manière analogue, soit $Q_{i,j}$ la probabilité des points $\boldsymbol{y_i}$ et $\boldsymbol{y_j}$ d'être proche dans $\mathbb{R}^d$. La difficulté est d'éviter l'agglomération des points (*crowding problem*), due à la différence de volume entre $\mathbb{R}^D$ et $\mathbb{R}^d$. En effet, l'espace disponible dans $\mathbb{R}^d$ pour loger les points de distances modérées est insuffisant comparé à l'espace disponible pour loger les points proches [14]. Une loi du t-Student de degré de liberté 1 (loi à queue lourde) est choisie, menant à :

$$Q_{i,j} = \frac{1}{Z'} \left(1 + \|\boldsymbol{y_i} - \boldsymbol{y_j}\|_2^2\right)^{-1}$$

où $Z' = \sum_{k \neq l} (1 + \|\boldsymbol{y_k} - \boldsymbol{y_l}\|_2^2)^{-1}$ est de même une constante de normalisation garantissant que les $Q_{i,j}$ somment à 1.

Finalement, la fonction de perte à minimiser est donnée par la divergence de Kullback-Leibler entre les deux distributions P et Q : $C_{Loss} = D_{KL}(P||Q)$. Les points $\boldsymbol{y_i}$ sont obtenus en minimisant la fonction de perte (par descente de gradient).

Les *points forts* de t-SNE sont :
— Sensible à la structure locale dans l'espace d'origine ;

14. Par exemple, en dimension 2 nous ne pouvons pas représenter 10 points équidistants entre eux, alors que c'est possible en dimension 10.

— Révèle la structure à plusieurs échelles sur la carte en 2d si la perplexité est correctement ajustée ;
— Préserve les groupements (particulièrement frappant sur les données de Mnist [15]) ;
— Évite de projeter tous les points au centre du plan.

Ses *limitations* sont :
— Sa complexité : la fonction de perte n'est pas convexe et son optimisation requiert plusieurs heures pour des corpus de millions de documents [16] , alors qu'une SVD demande quelques minutes ;
— l'algorithme est stochastique, des exécutions différentes mènent à des représentations (complètement) différentes ;
— enfin, la structure globale n'est pas préservée.

Évaluation L'évaluation d'une nouvelle représentation se fait par rapport à une tâche fixée : dans le cas de l'apprentissage supervisé où les données sont associées à une classe, l'évaluation est le score de classifications correctes pour un algorithme de κ-plus proches voisins dans la représentation obtenue.

Dans le cas *non-supervisé*, une évaluation possible repose sur l'éparpillement des rangs de voisinage. La mesure, appelée fiabilité@κ (trustworthiness) est définie par

$$Fiabilite@\kappa = 1 - \frac{1}{N^2 \kappa} \sum_{i,j} (\mathbf{rg}_i^{initial}(j) - \kappa) \times \mathbb{1}_{\mathbf{rg}_i^{repr}(j) < \kappa} \tag{2.3}$$

Où $\mathbf{rg}_i^{initial}(j)$ est le rang du point j parmi les voisins du point i dans l'espace initial, et $\mathbf{rg}_i^{repr}(j)$ le même rang dans l'espace de représentation nouveau (Maaten and Hinton, 2008).

15. https ://indico.io/blog/wp-content/uploads/2015/08/mnist.jpg
16. Sur Intel Core i7 2.10GHz x 4

Chapitre 3

Système de recommandation

Ce chapitre présente le problème de recommandation, les contextes et les approches de l'état de l'art de recommandation ainsi que les différentes manières de les évaluer.

3.1 Introduction

Un système de recommandation (SR) est un algorithme qui suggère des objets potentiellement pertinents pour un utilisateur. Le terme *d'objets* désigne des produits de toutes sortes : produits à acheter, musiques à écouter, articles à lire ou bien offres d'emplois auxquelles postuler. Un système de recommandation se spécialise sur un type d'objet en particulier. Dans sa forme la plus simple, un SR offre une liste d'objets triés par utilisateur où les objets les plus pertinents sont mis en tête. Dans la plupart des situations, le SR utilise les préférences passées de l'utilisateur pour lui suggérer de nouveaux objets. Les préférences de l'utilisateur sont de toutes sortes et parfois collectées à son insu, incluant des informations explicites (e.g., s'il a noté un film, donné un avis, laissé un *like*) ou implicites (e.g., le temps passé à lire un article de journal). En résumé, son historique de navigation Web regroupe l'ensemble des données implicites. En entrée nous disposons donc :

— des archives de l'utilisateur ;
— des archives d'une communauté d'utilisateurs ;
— d'une description explicite des utilisateurs (parfois ; et parfois aussi d'une description implicite) ;
— d'une description explicite des objets (parfois).

Ce domaine a connu une explosion depuis l'arrivée d'Internet, et plus précisément depuis l'ère des «big data». Pour certains services, la recommandation joue un rôle central. Par exemple Amazon dispose de milliards d'objets à vendre ; un système de recommandation ciblé permet aux utilisateurs de ne pas être noyés dans la diversité de produits. 35% des ventes d'Amazon viennent de la recommandation de produits. Un autre exemple est Netflix, connu pour son système de recommandation de vidéos, dispose d'un vaste catalogue de films, selon qui deux tiers des films visionnés proviennent du système de recommandations (Amatriain, 2014). Nous pouvons aussi citer Google News, qui met en avant les articles jugés les plus pertinents, où le système de recommandations génère 38% de clics supplémentaires. D'autres exemples sont YouTube, Tripadvisor ou Spotify, pour lesquels la recommandation est un élément essentiel de la plateforme.

Il existe des conférences et des séminaires spécialement dédiés aux SRs. La conférence la plus

connue est *ACM Recommender Systems* (RecSys), établie depuis 2007. C'est le premier évènement annuel sur la recherche des techniques de recommandation et ses applications. De plus, chaque année, un challenge portant sur la recommandation est associé à cette conférence ; nous y reviendrons plus en détail (Section 4.2.2).

3.1.1 Définitions

Nous définissons ici les principaux termes qui apparaissent dans les systèmes de recommandation (Ricci et al., 2011).

- Un **utilisateur** est celui à qui nous voulons recommander des objets / produits. Typiquement un utilisateur d'un site Web, un chercheur d'emploi sur Pôle emploi, un internaute sur Youtube ;
- Un **objet** est ce que l'on veut recommander : une offre d'emploi, un article de journal, une vidéo, une annonce publicitaire ;
- La **matrice collaborative** correspond à l'ensemble des *notes* ou des *interactions* entre les *utilisateurs* et les *objets*, rangées dans une matrice notée $\mathcal{R}$: les utilisateurs sont en ligne, les objets en colonnes. La valeur $\mathcal{R}_{u,i}$ correspond à la note, ou à la trace de navigation implicite, laissée par l'utilisateur u sur l'objet i. Dans le cas où les interactions sont binaires, un 1 correspond toujours à une note favorable [1]. Par exemple un *clic*, un *like*, une *candidature*. Dans la suite, nous désignerons par le seul mot «note» tous les types d'interactions.

Avant de présenter les algorithmes état de l'art, précisons deux cas d'applications pour la recommandation, schématisés (Table 3.1) :

- Dans le cas du **démarrage à chaud** (warm-start) nous disposons des notes laissées par les utilisateurs sur les objets, cela correspond aux données collaboratives. Ces données peuvent être ordinales dans le cas de notes explicites ou binaires dans le cas d'action implicites. Ce cas intervient lorsque utilisateurs et objets sont amenés à rester longtemps sur la plate-forme (par exemple sur Netflix ou Amazon) ;
- Dans le cas du **démarrage à froid** objet, nous considérons un nouvel objet qui vient d'apparaître dans la liste des objets disponibles, par exemple une offre d'emploi ou une vidéo qui viennent juste d'être postées. Ce cas est essentiel dans le domaine de l'emploi, où de nouvelles offres sont publiées chaque jour et où de plus les anciennes offres (déjà satisfaites) ne doivent pas être recommandées. Le cas du démarrage à froid objet est donc notre principal cas d'étude. Le cas du démarrage à froid utilisateur concerne un utilisateur qui vient de s'enregistrer sur la plate-forme et qui n'a donc encore laissé ni trace d'usage ni évaluation.

Le prix Netflix Un challenge très populaire de recommandation a été organisé par Netflix de 2006 à 2009. Beaucoup d'équipes ont participé à ce challenge, enrichissant la littérature sur le filtrage collaboratif. Un prix d'un million de dollars était offert à l'équipe améliorant l'algorithme de recommandation de Netflix (CineMatch) de plus de 10%. Le grand prix est remporté en 2009 par la fusion de trois équipes, utilisant chacune un ensemble de méthodes. Cependant, Netflix n'a jamais implémenté la solution (vaste méthode d'ensemble) proposée par les gagnants, pour trois raisons : trop lourde pour passer à l'échelle, trop de travail d'ingénierie pour une amélioration incertaine et enfin la recherche Netflix s'est tournée vers d'autres problématiques avec plus d'impact que la prédiction des notes (Amatriain, 2014). Dans le cas du challenge Netflix, nous sommes en filtrage

1. aussi appelé "retour-positif" ou "filtrage collaboratif à une classe" (Pan et al., 2008)

						1	1		1
			1			1			
1			1		1				
	1		1			1		1	1
1						1			
						1	1		1
						1	1		1
			1			1			
1			1		1				
1	1		1			1		1	1
				1		1			
			1				1		1

(a) Démarrage à chaud. Il s'agit de recommander des objets parmi *tous* les objets non-sélectionnés. Ce cas correspond aux spécifications du filtrage collaboratif.

			1	1		1			
1			1						
1		1							
1			1		1				
			1						
			1	1					
			1	1					
1			1						
1		1							
1			1		1	1			
	1		1						
1				1					

(b) Démarrage à froid *objet*, aussi appelé "out-of-matrix" (Wang and Blei, 2011). Lorsque de nouveaux objets sont ajoutés à la plateforme, aucun utilisateur ne les a encore vus. Ces nouvelles colonnes correspondent par exemple aux nouvelles offres d'emplois postées la veille.

	1		1			1		1	1
1						1			
		1				1	1		1
						1	1		1
			1			1			
1			1		1				
1	1		1			1		1	1
				1		1			
			1				1		1

(c) Démarrage à froid *utilisateur*. Lorsqu'un nouvel utilisateur s'inscrit sur la plateforme, il n'a encore aucune interaction avec les objets.

TABLE 3.1 – Les trois cas de figure pour un système de recommandation. Les données collaboratives sont les 1, traduisant qu'un utilisateur a *sélectionné* un objet, les autres valeurs sont 0. Les cellules roses sont les *objets cibles*. Dans le cas du démarrage à froid objet (3.1b), l'objectif est de recommander *seulement* les nouveaux objets aux utilisateurs. Dans le cas du démarrage à froid utilisateur (3.1c), il s'agit de recommander aux nouveaux utilisateurs des objets déjà présents. Le cas du pur démarrage à froid (nouvel utilisateur et nouvel objet) n'est pas traité dans ce manuscrit.

collaboratif explicite car il n'y a pas de contenu ni pour les objets ni pour les utilisateurs. Pour fixer les idées, les données du challenge contiennent 480 000 utilisateurs, 17 000 vidéos et comportant 100 480 000 notes (de 1 à 5 étoiles) s'étendant sur une période de 7 ans.

3.2 Les méthodes de recommandations

L'objectif est de sélectionner, pour chaque utilisateur, les objets les plus pertinents qu'il n'a pas encore vus. Les méthodes de recommandations s'appuient sur deux approches principales : celles basées sur le filtrage collaboratif et celles basées sur le contenu.

- Le **filtrage collaboratif (FC)** (collaborative filtering) consiste à utiliser *exclusivement* les évaluations engendrées par les différents utilisateurs pour produire des recommandations. Le système de filtrage collaboratif a été popularisé par Amazon avec la fonctionnalité "les clients ayant acheté i ont aussi acheté j"
- L'approche **basée sur le contenu** (content-based) consiste à recommander des objets *similaires* à ceux que l'utilisateur a aimé par le passé. Dans cette approche le calcul de similarité repose exclusivement sur le contenu de l'objet. Par exemple la similarité sémantique entre deux offres d'emplois, ou bien la similarité sur le contenu d'une vidéo (les acteurs, le genre du film, la date de parution).

Dans ces deux approches, la première étape consiste à former la matrice collaborative $\mathcal{R}$. Par convention, les utilisateurs sont en ligne, les objets en colonne et la note de l'utilisateur u sur l'objet i est $\mathcal{R}_{u,i}$. Dans le cas de données implicites (achats, postulation à une offre, consultations), la matrice $\mathcal{R}$ ne contient que des valeurs binaires. Une fois la matrice $\mathcal{R}$ construite, le système estime les «valeurs absentes» en reconstruisant une matrice de score $\widehat{\mathcal{R}}$, cette matrice $\widehat{\mathcal{R}}$ permet enfin de proposer à chaque utilisateur les objets non-vus avec les meilleurs scores estimés. Dans ces deux approches, deux sous-catégories sont considérées : les techniques à base de modèle et celles à base de mémoire :

- Dans le cas **à base de modèle** (model-based), nous avons un modèle qui détermine, en fonction de l'utilisateur et de l'objet en entrée, le score de recommandation. Pour ce faire, utilisateurs et objets sont représentés dans un même espace, appelé espace de représentation latent, et leur score est le produit scalaire entre la représentation latente de l'utilisateur et celle de l'objet. C'est en particulier le cas de "SVD" (Équation 3.1) et de CTR (Section 3.2.2.1), décrites ci-après ;
- Les techniques **à base de mémoire** (memory-based) utilisent le voisinage des objets, ou plus rarement le voisinage des utilisateurs, pour générer les recommandations. Deux cas symétriques apparaissent alors :
  - le cas *orienté objet* : Vous aimez l'objet i, vous allez aimer l'objet i' (car i et i' sont similaires). On se fie au passé de l'utilisateur pour lui recommander des objets similaires à ceux qu'il a déjà aimés ;
  - le cas *orienté utilisateur* : Vous allez aimez l'objet i car les autres utilisateurs u' similaires à vous aiment aussi l'objet i. On se fie au passé de l'objet que l'on recommande aux utilisateurs similaires.

Rappelons que dans le cas où les notes de $\mathcal{R}$ sont des booléens, une note traduit toujours une affinité positive pour l'objet. Cependant, une absence de note ne traduit pas systématiquement un jugement négatif. En effet, les zéros de la matrice collaborative ne sont pas nécessairement de *vrais* zéros : soit l'utilisateur n'a effectivement pas aimé l'objet et n'a pas laissé de note, soit il n'a tout simplement pas vu l'objet.

Méthodes	Approches	Données d'entrée	Exemples
Filtrage collaboratif	Basée sur le modèle	$\mathcal{R}$	Factorisation de matrice "SVD" (Équation 3.1), CDAE (Wu et al., 2016), BPR (Rendle et al., 2009), ItemRank (Gori et al., 2007)
	Basée sur la mémoire (*i.e.* plus proche voisins)	$\mathcal{R}$	Plus proches voisins, $\mathcal{R}$-based (Section 3.2.1.2),
Basé sur le contenu	Basée sur la mémoire (orientée objet)	$\mathcal{R}$, description objets.	LSA (Équation 3.10), CLE (Section A), LAJAM (Section 6)
	Basée sur le modèle	$\mathcal{R}$, Description des utilisateurs et des objets.	CTR (Wang and Blei, 2011), CDL (Wang et al., 2015), (Volkovs et al., 2017)

TABLE 3.2 – Classification des techniques de recommandation

3.2.1 Filtrage collaboratif

Nous décrivons dans cette section les méthodes de filtrage collaboratives, dont la particularité commune est de disposer, comme seule information, de la matrice collaborative $\mathcal{R}$. Ces approches permettent donc de traiter exclusivement les cas de démarrage à chaud.

3.2.1.1 À base de modèle

Une approche à base de modèle représente utilisateurs et objets dans un même espace puis estime le score en fonction de ces deux représentations à l'aide du produit scalaire.

"SVD" L'approche la plus populaire à base de modèle est la factorisation de matrice développée durant le challenge Netflix (Koren et al., 2009). Cette approche consiste à modéliser utilisateurs et objets en décomposant la matrice collaborative $\mathcal{R}$ à l'aide d'une technique de factorisation de matrice adaptée de la décomposition en valeurs singulières (Section 2.2.3). Cette approche permet une approximation de faible rang $\widehat{\mathcal{R}}$ de la matrice $\mathcal{R}$ en minimisant l'erreur au sens des moindres carrés sur les données *observées*, *i.e.* sur les notes [2]. Notons que dans l'hypothèse d'une matrice $\mathcal{R}$ de rang elevé, nous ne pouvons pas généraliser et estimer le nombre de paramètres de $\widehat{\mathcal{R}}$ (de l'ordre de $n \times m$) à partir des informations clairsemées disponibles ; l'hypothèse de rang faible est nécessaire [3]. Une fois les représentations latentes de rang faible obtenues, certaines interprétations sont possibles. Par exemple la représentation latente d'un film peut être interprétée en fonction des types de films (e.g. un film étant 20% de romance et 60% de comédie) ou de publics (à 80% orienté pour les hommes et à 65% sérieux) (Koren et al., 2009).

2. Contrairement à la SVD décrite (Section 2.2.3) qui optimise l'erreur au sens des moindre carrés sur *toutes* les valeurs de la matrice.

3. Dans le cas d'une matrice de rang k, le nombre de paramètres à estimer passe de $n \times m$ à $(n+m) \times k$ (Équation 3.1).

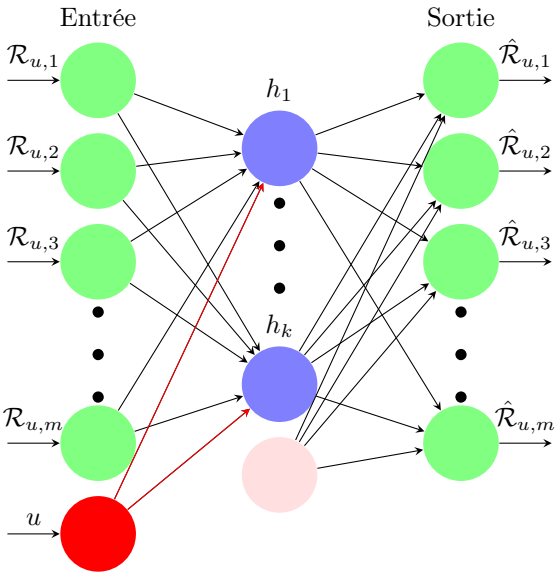


FIGURE 3.1 – Architecture de CDAE. Le noeud rouge ainsi que ses poids sortant sont spécifique à l'utilisateur.

La formalisation est la suivante : trouver des représentations de faible rang $P = (\boldsymbol{p_1}, \ldots, \boldsymbol{p_n})$ des utilisateurs et $Q = (\boldsymbol{q_1}, \ldots, \boldsymbol{q_m})$ des objets. $\boldsymbol{p_u} \in \mathbb{R}^k$ (respectivement $\boldsymbol{q_i} \in \mathbb{R}^k$) est la représentation de l'utilisateur u (de l'objet i) dans l'espace latent et k est un hyper-paramètre du modèle qui représente la dimension de l'espace latent. La fonction à minimiser est la suivante :

$$\sum_{u,i|\mathcal{R}_{u,i}>0} (\mathcal{R}_{u,i} - \boldsymbol{p_u}^T \boldsymbol{q_i})^2 + (\lambda \left\|\boldsymbol{q_i}\right\|_2^2 + \lambda' \left\|\boldsymbol{p_u}\right\|_2^2) \tag{3.1}$$

Le premier terme est le terme d'erreur de reconstruction des données observées, le second est un terme de régularisation sur les représentations latentes.

Pour résoudre ce problème de minimisation, il est commun d'utiliser une descente de gradient stochastique (Koren et al., 2009) ou bien une approche de minimisation alternée [4]. Dans le cas où les notes sont binaires (observations implicites), l'approche est affinée pour prendre en compte l'incertitude concernant les notes absentes [5]. L'erreur de reconstruction de chaque note est pondérée par un poids de confiance $c_{u,i}$, avec $c_{u,i} = 1$ si $\mathcal{R}_{u,i} = 1$ (i.e. confiance maximale pour les valeurs observées) et $c_{ui} = .01$ si $\mathcal{R}_{u,i} = 0$ (*i.e.* confiance relative pour les valeurs non observées). Ainsi la fonction à optimiser devient :

$$\sum_{u,i} c_{ui} \cdot (\mathcal{R}_{u,i} - \boldsymbol{p_u} \boldsymbol{q_i}^T)^2 + \lambda \left\|\boldsymbol{q_i}\right\|_2^2 + \left\|\boldsymbol{p_u}\right\|_2^2) \tag{3.2}$$

4. Le problème global n'est pas convexe ; mais en fixant les valeurs de P ou de Q il devient quadratique et peut être résolu de façon optimale à chaque étape.

5. adapté de (Hu et al., 2008) par (Wang and Blei, 2011) (Section 3.2.2.1).

Collaborative Denoising Auto-Encodeur (CDAE) (Wu et al., 2016) est une approche utilisant les Auto-Encodeurs (Section 1.1.2) représentée (Figure 3.1). Cette méthode repose sur l'encodage-décodage de la représentation de chaque utilisateur dans l'espace des notes des objets (*i.e.* $\mathcal{R}_{u,\cdot}$), en ce sens cela s'apparente à factoriser (de manière non-linéaire) la matrice collaborative (Koren et al., 2009). La sortie du réseau est la reconstruction $\widehat{\mathcal{R}}$, optimisée pour reconstruire les notes de $\mathcal{R}$ au sens des moindres carrés. Les principales contributions de cette méthode sont :

- l'ajout d'un "neurone" pour chaque utilisateur, permettant une représentation latente de l'utilisateur (neurone rouge et ses poids sortants (Figure 3.1)) ;
- La corruption de l'entrée, par l'ajout d'un bruit multiplicatif, dans l'idée de dropout ;
- L'échantillonnage négatif (negative sampling) pour les valeurs non-observées, ainsi qu'une régularisation L2 sur l'ensemble des poids du réseau.

Ce modèle étend AutoRec (Sedhain et al., 2015) et (Strub and Mary, 2015), en ajoutant le terme de représentation latente des utilisateurs et l'optimisation de dé-bruitage par échantillonage négatif. Citons également Deep Collaboratif Filtering (Li et al., 2015) qui représente à la fois les utilisateurs et les objets dans l'espace latent à l'aide d'un Auto-Encodeur dé-bruiteur marginalisé (mDA (Chen et al., 2012)).

Méthodes optimisant le rang Si les précédentes méthodes se concentrent sur la valeur des notes à reconstruire, d'autres méthodes sont proposées pour conserver le même ordre d'objet pour chaque utilisateur. Ainsi, Bayesian Personalized Ranking (Rendle et al., 2009) propose un critère bayesien pour optimiser directement la perte associée à la mesure de rang AUC (Section 3.3.3). Cependant, cette méthode qui requiert 7 hyper-paramètres est jugé difficile à calibrer (Verstrepen and Goethals, 2014). Collaborative Less is More filtering (Shi et al., 2012,0) optimise également le rang des recommandations, par descente de gradient, en dérivant une version lissée du Mean Reciprocal Rank (Section 3.3.3).

ItemRank (Gori et al., 2007) est basé sur l'algorithme de PageRank (Page et al., 1999) (voir également (Fouss et al., 2007; Zhou et al., 2007)). Le score d'un objet i pour un utilisateur u est proportionnel à la probabilité que u visite i par une marche aléatoire dans le graphe collaboratif. Ce graphe non-orienté est construit à partir des données collaboratives : chaque objet est représenté par un noeud et deux objets sont relié par un arc s'ils sont notés par un même utilisateur [6]. Une approche similaire, s'inspirant de la marche aléatoire SALSA [7] (Lempel and Moran, 2000), est utilisée pour recommander des personnes à suivre chez Twitter (Gupta et al., 2013). Les arcs du graphe bi-partie utilisateur-utilisateur où est lancée la marche aléatoire sont définis par le fait qu'un utilisateur *suit* ("follow") un autre utilisateur. Cependant, ces deux visions de graphe (non-orienté et bi-partie) sont peu répandues dans la communauté de système de recommandation (ainsi elles n'apparaissent pas dans la revue sur le filtrage collaboratif (Su and Khoshgoftaar, 2009)), et ne sont pas comparées avec les méthodes état de l'art décrites plus haut.

3.2.1.2 À base de mémoire, orientée objet

La méthode à base de mémoire pour le filtrage collaboratif est certainement la méthode la plus simple et la plus directe pour construire un système de filtrage collaboratif. Elle est aussi

6. Le poids d'un arc est donné par la matrice de transition T, avec $T_{i,j}$ le nombre d'utilisateurs u t.q. $\mathcal{R}_{u,i} > 0$ et $\mathcal{R}_{u,j} > 0$.

7. variante de PageRank.

	i_1	i_2	i_3	i_4	i_5
u_1		1		1	1
u_2	1	?		1	
u_3			1		1
u_4	1				
u_5		1	1	1	

TABLE 3.3 – Exemple de filtrage collaboratif à base de mémoire *orienté objet*

appelée à l'époque du challenge Netflix *basée sur les voisins* (Koren, 2008; Jahrer et al., 2010). Le filtrage collaboratif à base de mémoire peut s'écrire de deux façons symétriques : orientée objet ou bien orientée utilisateur. Nous décrivons ici l'orientation *objet*, qui est la plus répandue. Elle est aussi appelée «Item-based Collaborative Filtering» par (Sarwar et al., 2001) et «Item-based Top-N Recommendations» par (Su and Khoshgoftaar, 2009). Cette méthode se décompose en trois temps :

— La première étape définit les voisins, ainsi que les valeurs de *similarités* entre les objets ;
— la deuxième étape exploite ces similarités pour prédire des notes, à l'aide des anciennes notes des utilisateurs $\mathcal{R}$ - d'où le terme de mémoire ;
— enfin la dernière étape recommande les objets suivant l'ordre relatif de leur notes $\widehat{\mathcal{R}}$.

L'estimation des notes manquantes est ainsi basée sur la proximité avec les anciens objets que l'utilisateur a déjà notés. «Orienté objet» traduit l'hypothèse qu'un utilisateur aime les objets similaires à ceux qu'il a aimés par le passé, induisant une recommandation de "plus du même". Dans le cas du filtrage collaboratif, deux objets sont similaires si plusieurs utilisateurs ont aimé ces deux objets à la fois, la similarité repose donc exclusivement sur les notes.

Le calcul de $\widehat{\mathcal{R}}$ est en deux étapes : la première étape est de construire l'ensemble des objets j «voisins» qui co-apparaissent avec l'objet i, en même temps nous calculons leurs *similarités* avec i (Éq. 3.3 à 3.4). Puis nous pondérons les notes des objets par leurs similarités, cela pour chaque objet j noté par l'utilisateur u (Éq. 3.5 à 3.8), utilisant ainsi la mémoire des notes.

Similarité Il y a plusieurs façons de calculer la *similarité* entre l'objet i et l'objet j, noté $sim(i,j)$.

— *Cosinus :* Nous comparons la valeur des notes entre chaque paire d'objets, normalisée par la norme de leurs notes.

$$Cos(i,j) = \frac{\langle \mathcal{R}_{\cdot,i}, \mathcal{R}_{\cdot,j} \rangle}{\|\mathcal{R}_{\cdot,i}\|_2 \cdot \|\mathcal{R}_{\cdot,j}\|_2} = \frac{\sum_u r_{u,i} \cdot r_{u,j}}{\sqrt{\sum_u r_{u,i}^2} \cdot \sqrt{\sum_u r_{u,j}^2}} \tag{3.3}$$

Dans le cas binaire, cela revient à compter le nombre de cliques communs, normalisé par la racine du nombre de cliques [8]

$$Cos(i,j) = \frac{\sum_u r_{u,i} \cdot r_{u,j}}{\sqrt{\sum_u r_{u,i} \cdot \sum_u r_{u,j}}}$$

8. Cela peut aussi s'écrire $Cos(i,j) = \frac{|\mathcal{U}(i) \cap \mathcal{U}(j)|}{|\mathcal{U}(i)|^{\frac{1}{2}} |\mathcal{U}(j)^{\frac{1}{2}}|}$. Notons que dans (Aiolli, 2012) la normalisation est optimisée à l'aide de deux paramètres α et q, considérant ainsi la similarité $Cos^q_\alpha(i,j) = (\frac{|\mathcal{U}(i) \cap \mathcal{U}(j)|}{|\mathcal{U}(i)|^\alpha |\mathcal{U}(j)|^{1-\alpha}})^q$. Une plus grande valeur de q augmente l'importance des hautes valeurs de cosinus par rapport aux plus faibles, voir (Section 7).

— *Cosinus ajusté :* Dans le cas où les notes ne sont pas binaires, le *cosinus ajusté* est proposé pour prendre en compte les différences d'échelle de notes entre les utilisateurs. La formule devient alors :

$$CosA(i,j) = \frac{\sum_{u \in U_{ij}} (r_{u,i} - \widehat{r_u}) \cdot (r_{u,j} - \widehat{r_u})}{\sqrt{\sum_{u \in U_{ij}} (r_{u,i} - \widehat{r_u})^2} \cdot \sqrt{\sum_{u \in U_{ij}} (r_{u,j} - \widehat{r_u})^2}} \tag{3.4}$$

Avec $\widehat{r_u}$ la moyenne des notes de l'utilisateur u, et U_{ij} l'ensemble des utilisateurs qui ont noté à la fois l'objet i et l'objet j.

Dans notre étude, comme nous n'avons que des valeurs binaires dans la matrice collaborative $\mathcal{R}$, nous privilégions la similarité cosinus.

Mémoire Une fois les valeurs de similarités obtenues, l'estimation s'écrit :

$$\widehat{\mathcal{R}}_{u,i} = \frac{\sum_j sim(i,j) \cdot \mathcal{R}_{u,j}}{\sum_j |sim(i,j)| \cdot \mathbb{1}_{\{\mathcal{R}_{uj}>0\}}} \tag{3.5}$$

La prédiction d'un score $\widehat{\mathcal{R}}_{u,i}$ entre l'utilisateur u et un nouvel objet i correspond donc à la moyenne des notes de l'utilisateur u sur les objets j, pondérée par la similarité des objets j avec l'objet i. La normalisation assure que le score prédit reste dans l'intervalle des notes possibles. Dans le cas où $\mathcal{R}$ est binaire, $\mathcal{R}_{u,i} = \mathbb{1}_{\{R_{uj}>0\}}$ et la normalisation n'est pas pertinente. Nous prenons alors :

$$\widehat{\mathcal{R}}_{u,i} = \sum_j sim(i,j) \cdot r_{u,j} = \sum_{j \in \mathcal{O}(u)} sim(i,j) \tag{3.6}$$

Où $\mathcal{O}(u)$ est l'ensemble des objets noté par l'utilisateur u. Notons que (Deshpande and Karypis, 2004) régularisent l'équation précédente en ne tenant compte que les κ plus proches voisins de l'objet considéré :

$$\widehat{\mathcal{R}}_{u,i} = \sum_j sim(i,j) \cdot r_{u,j} \cdot |\kappa\text{-}nn(j) \cap \{i\}| \tag{3.7}$$

$$= \sum_{j \in \mathcal{O}(u), i \in \kappa\text{-}nn(j)} sim(i,j) \tag{3.8}$$

Cette méthode a l'avantage de réduire l'importance des objets *proches* de tout le monde ; en contrepartie, elle demande de calculer $|\kappa\text{-}nn(j) \cap \{i\}|$ et d'ajuster l'hyper-paramètre κ. Cette contrainte permet de diminuer le nombre de termes intervenant dans le calcul de la similarité.

Orientée utilisateur La méthode orientée utilisateur explore les similarités entre utilisateurs plutôt que les similarités entre objets. Il s'agit ici de recommander des objets que d'autres utilisateurs aux goûts semblables ont aimés. L'équation est de la même forme que celle orientée objet (Équation 3.6) :

$$\widehat{\mathcal{R}}_{u,i} = \frac{\sum_v sim(u,v) \cdot \mathcal{R}_{v,i}}{\sum_v |sim(u,v)| \cdot \mathbb{1}_{\{\mathcal{R}_{vi}>0\}}}$$

	i_1	i_2	i_3	i_4	i_5
u_1		1			1
u_2	1		?	1	1
u_3	1		1	1	1
u_4		1			1
u_5	1		1		

TABLE 3.4 – Exemple de filtrage collaboratif basé sur la mémoire *orienté utilisateur*

La encore, dans le cas où $\mathcal{R}$ est binaire la normalisation ne peut plus s'appliquer. De plus, l'ordre relatif entre $\widehat{\mathcal{R}}_{u,i}$ et $\hat{R}_{uj}$ ne dépend pas de la normalisation. Nous prenons donc :

$$\widehat{\mathcal{R}}_{u,i} = \sum_{v} sim(u,v) \cdot r_{v,i} \tag{3.9}$$

Le calcul de similarités entre utilisateurs s'obtient avec les mêmes équations que dans le cas orienté objet.

Différence d'orientation utilisateurs ou objets L'utilisation de ces deux méthodes dépend du nombre relatif des utilisateurs et des objets. En effet, pour optimiser les temps de calcul, il est préférable de pre-calculer la matrice de similarité. Par exemple pour le cas orienté objet, la matrice de similarité entre objets est de taille $m \times m$. La similarité entre utilisateurs est dite plus dynamique (Amatriain, 2014) et donc plus instable à pre-calculer. La matrice des similarités objet-objet peut être calculée hors ligne. La différence entre approche orientée objet et orientée utilisateur est illustrée sur un cas d'étude (Annexe B).

Bandits manchots Citons également une autre approche fondée sur les bandits manchots (Bubeck et al., 2012) pour les systèmes de recommandation (Li et al., 2010; Mary et al., 2015) où chaque objet est un des bras de bandit manchot. La récompense est la note de l'utilisateur sur l'objet. Ces approches ne sont pas étudiées ici.

3.2.1.3 Limites du filtrage collaboratif

Le filtrage collaboratif nécessite peu d'efforts d'ingénierie et peu de connaissance *a priori* pour produire de bonnes recommandations. Cependant, la principale limite du filtrage collaboratif vient du fait qu'il est inopérant dans le cas de nouveaux objets ou de nouveaux utilisateurs, où des méthodes à base de contenu sont requises. Une autre limite vient de la nécessité d'avoir beaucoup de données pour produire des recommandations pertinentes (Shi et al., 2013).

De plus les objets doivent être standards, c'est à dire qu'ils doivent être *exactement les mêmes* pour pouvoir appliquer les méthodes de filtrage collaboratif. Peu de méthodes prennent en compte l'évolution des utilisateurs et de leurs préférences au cours du temps ; voir par exemple Matchbox **?**.

Le filtrage collaboratif est performant pour les utilisateurs dont les goûts sont suffisamment similaires à ceux d'autres utilisateurs. Les individus aux goûts différents de ceux de la communauté sont mal pris en compte. Ce problème, dit «du mouton gris», est en partie résolu par la recommandation à base de contenu.

Un autre problème associé au phénomène de «longue traîne» (Anderson, 2012) est celui de la popularité : il est en effet plus difficile pour un système de filtrage collaboratif de recommander des objets non populaires (disposant de peu de notes). De plus, ce biais tend à s'accentuer s'il n'est pas pris en compte, recommandant ainsi davantage les objets populaires. Cela est contre-productif car l'ensemble des (nombreux) objets non populaires a souvent un marché plus important que celui des quelques objets les plus populaires.

Enfin, contrairement aux méthodes basées sur le contenu, les recommandations n'ont pas d'interprétation possible a priori, sauf le fameux "ceux qui ont acheté i ont aussi acheté j".

3.2.2 À base de contenu

Lorsque les données collaboratives sont trop parcimonieuses, le filtrage collaboratif seul ne permet plus de générer de recommandations satisfaisantes. Si nous disposons d'informations sur les objets, il est naturel de recourir alors aux méthodes s'appuyant sur le contenu (content-based) où les similarités entre objets font intervenir leurs caractéristiques *i.e.* leur contenu.

À base de mémoire, orientée objets [9] Le cas le plus fréquent est de disposer d'informations sur les objets et de notes pour certains utilisateurs. L'approche à base de mémoire orientée objets est semblable au filtrage collaboratif à base de mémoire orienté objets, à la différence que les similarités entre objets sont maintenant calculées sur le contenu des objets et non plus sur les notes de $\mathcal{R}$. Par exemple deux films sont similaires s'ils ont les mêmes acteurs, sont du même genre (comédie, drame, etc.). Ou bien deux offres d'emplois sont a priori similaires si elles sont du même secteur, de salaire équivalent, dans un domaine semblable et avec les mêmes mots-clé. Cette approche est au centre de notre étude, en reposant sur la similarité entre document textes (Section 2). Le score de recommandation s'obtient de manière semblable au filtrage collaboratif à base de mémoire (Équation 3.6) :

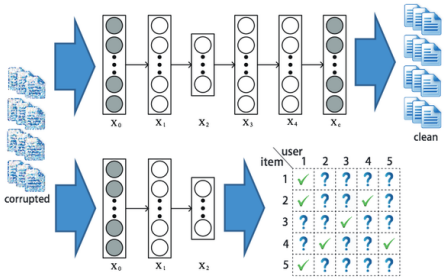
$$\widehat{\mathcal{R}}_{u,i} = \sum_j sim(i,j) \cdot \mathcal{R}_{u,j} \tag{3.10}$$

3.2.2.1 À base de modèle

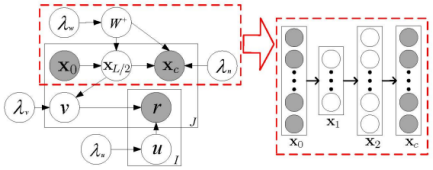
De même que dans le cas du filtrage collaboratif, les approches à base de modèle représentent utilisateurs et objets dans un même espace latent pour calculer leur score. Lorsque des informations auxiliaires sont disponibles pour les objets (et éventuellement les utilisateurs), la représentation latente peut prendre en compte ces informations ainsi que les données collaboratives.

Collaborative Topic Regression (CTR) Wang and Blei (2011) modélisent utilisateurs et objets dans le même espace latent. Les représentations latentes des utilisateurs P et celles des objets Q sont optimisées par la méthode de filtrage collaborative avec les données implicites (Équation 3.2). La contribution principale de CTR est de permettre à la représentation de l'objet j de se décomposer en deux termes $\boldsymbol{Q_j} = \boldsymbol{\theta_j} + \boldsymbol{\epsilon_j}$, où $\boldsymbol{\theta_j}$ est la représentation fixe LDA (Section 2.2.4) du document et $\boldsymbol{\epsilon_j}$ est une partie adaptative, permettant de modéliser l'intérêt des utilisateurs pour cet objet et donc de mieux estimer la matrice $\mathcal{R}$. Ainsi, dans le cas sans information auxiliaire pour les objets, $\boldsymbol{\theta} = \mathbf{0}$ et la méthode est celle du filtrage collaboratif. Dans le cas du démarrage à froid objet, $\boldsymbol{\epsilon} = \mathbf{0}$ (car l'objet n'est pas noté) et la méthode optimise $\langle \boldsymbol{P_u}, \boldsymbol{\theta_j} \rangle$ pour s'adapter à $\mathcal{R}_u i$.

9. Le cas orientée utilisateurs, plus rare, est tout à fait analogue



(a) En haut est représenté un Auto-Encodeur avec de-bruitage optimisant la reconstruction. En bas la représentation intermédiare est utilisé pour reconstruire la matrice $\mathcal{R}$.



(b) Modèle graphique. $\boldsymbol{P_u}$ (noté $\boldsymbol{u}$) et $\boldsymbol{Q_j}$ (noté $\boldsymbol{v}$) sont optimisé pour reconstruire les notes $\mathcal{R}_{uj}$ (noté r).

FIGURE 3.2 – Schéma pour CDL. Figures extraites de (Wang et al., 2015)

Collaborative Deep Learning CTR est étendue par Collaborative Deep Learning (CDL) (Wang et al., 2015) qui optimise également les représentations latentes P des utilisateurs et Q des objets. La représentation de l'objet j $\boldsymbol{Q_j} = \boldsymbol{\epsilon_j} + \boldsymbol{h_j}$ est la somme de sa représentation $\boldsymbol{h_j}$ par un Auto-Encodeur ainsi qu'un terme $\boldsymbol{\epsilon_j}$ optimisé sur les données collaboratives. Spécifiquement, le terme de contenu $\boldsymbol{h_j}$ correspond à la représentation intermédiaire d'un Auto-Encodeur profond avec dé-bruitage, entraîné par une approche bayesienne pour reconstruire le document partant de la représentation sac de mots (Figure 3.2a). Le terme $\boldsymbol{\epsilon_j}$ est optimisé pour ajuster la reconstruction de la matrice $\mathcal{R}$ par le produit scalaire de P et Q.

À base de connaissances : Les systèmes à base de connaissances s'appuient sur des filtres spécifiques du domaine, définis a priori, pour dégager les caractéristiques des objets qui sont susceptibles de répondre aux besoins spécifiques des utilisateurs. Nous sommes alors dans le cas de la recommandation directe, qui sort de l'objet de notre étude.

Les problèmes de recommandation peuvent aussi être liés à des problèmes de traitement des langues naturelles telle que la recherche d'information (Büttcher et al., 2016). Dans ce cas le modèle apprend comment apparier les mots clefs des utilisateurs et des objets. Ainsi la description d'un utilisateur correspond à une requête et l'ensemble des objets sont les éléments à ordonner dans le cas de recherche d'information. Les appariements sont éventuellement appris à l'aide de la matrice collaborative, mais la matrice $\mathcal{R}$ n'est plus utilisée lors de la prédiction du score. Cependant le problème de recommandation est rarement considéré comme un problème pur de traitement des langues naturelles [10]. Nous pouvons aussi considérer le problème du point de vue de l'implication textuelle (Recognizing Textual Entailment) (Dagan et al., 2006; Bowman et al., 2015; Rocktäschel et al., 2015; Parikh et al., 2016), mais à notre connaissance, cette approche n'a pas été considérée dans la littérature.

Méthodes	Description
Pondéré	Les scores sont combinés pour obtenir une seule recommandation.
Meta	Le système alterne entre les différentes techniques suivant la situation courante.
Mixe	Les recommandations provenant de différents systèmes sont présentées en même temps.
Combinaison de représentations	Combine l'information collaborative avec le contenu.
Augmentation de caractéristique	Utilise les notes prédites comme entrées pour un nouveau système.
Cascade	Un système affine les recommandations d'un autre système.

TABLE 3.5 – Méthodes de recommandation hybrides

3.2.3 Méthodes hybrides, méthodes d'ensembles

Pour affiner les recommandations, comme souvent en machine learning (Caruana et al., 2004; Triskelion, 201; Gorman, 2016), nous sommes amenés à considérer un ensemble de systèmes de recommandation. Il existe plusieurs façons de créer ce système d'«ensemble». Les systèmes de recommandation hybrides selon la classification de (Burke, 2002) sont représentés (Table 3.5). Nous les décrivons brièvement ici :

— Pondéré : C'est la méthode d'ensemble la plus simple lorsque l'on utilise deux modèles, par exemple l'un basé sur le contenu et l'autre basé sur le filtrage collaboratif. Les notes de recommandation de chaque système sont agrégées par consensus (Pazzani, 1999) ou par combinaison linéaire (Claypool et al., 1999), permettant des gains de performance. Cependant, cette méthode fait l'hypothèse implicite que chaque système se comporte de façon uniforme sur tous les objets [11].
— Meta : Pour pallier le problème rencontré dans le cas pondéré, nous utilisons des critères a priori pour savoir quand choisir un système particulier. Cela ajoute cependant de nouveaux paramètres, augmentant la complexité.
— Mixe : Les recommandations provenant de différents systèmes sont présentées en même temps [12] (Smyth and Cotter, 2000).
— Combinaison des représentations : Pour agréger information du contenu et information collaborative, nous considérons l'information collaborative comme un contenu supplémentaire. Par exemple les systèmes comme CTR et CDL vus ci dessus.
— Augmentation de caractéristiques : Un premier système de recommandation produit des évaluations pour chaque objet du système. Puis un second système utilise les notes prédites comme entrées. Les sorties du premier sont donc les entrées du second.
— Cascade (pre-sélection / heuristique) : C'est un processus séquentiel. Un premier système (souvent une heuristique rapide) produit une pre-sélection et éventuellement un tri grossier

10. En effet il est rare d'avoir des informations textuelles de qualité à la fois pour les utilisateurs et les objets.
11. Cela n'est plus vrai dans le cas d'un système par filtrage collaboratif où le nombre de notes par utilisateur varie, rendant les recommandations de qualité non homogène.
12. Ou bien les meilleures recommandations de chaque système sont agrégées (Aiolli, 2012).

des objets à recommander. Puis un second système plus fin travail sur les résultats du premier filtre et affine les suggestions finales. Le principal bénéfice est d'éviter d'employer le système fin sur les objets qui peuvent être rapidement éliminés par la première heuristique, ce qui permet d'être plus efficace en terme de complexité qu'un système pondéré. Cette approche remédie aux faux positifs générés par l'heuristique (Covington et al., 2016).

3.3 Évaluer un système de recommandation

La recommandation vise des buts divers (satisfaire les utilisateurs, augmenter la visibilité d'un produit), et l'évaluation ultime devra donc prendre en compte l'objectif final. Cependant il est souvent coûteux de tester les algorithmes sur de vrais utilisateurs et d'en mesurer les effets. De plus, certains effets peuvent ne pas être mesurables. L'évaluation nécessite donc l'utilisation de mesures intermédiaires. Dans la littérature des SRs, l'évaluation des algorithmes sur des données publiquement disponibles joue un rôle majeur. Il est en effet commun de calibrer et valider les modèles sur ces données avant de mettre les algorithmes en production. Cette mesure est aussi bénéfique pour tester et comparer différents algorithmes de façon reproductible.

La méthode standard pour évaluer un système est la même que celle utilisée en machine learning, s'appuyant sur un ensemble d'entraînement et un ensemble de test. Comme il s'agit ici de prédire de futures notes, nous séparons artificiellement l'ensemble des notes, ne dévoilant que l'ensemble d'entraînement pour ajuster les paramètres et nous évaluons les systèmes entraînés sur leur capacité à (re)découvrir les notes (cachées) de l'ensemble de test. Si nous avons l'opportunité de déployer le système en production, nous pouvons évaluer la qualité des recommandations en contrôlant le retour effectif des utilisateurs. Cette méthode est couramment nommé test A/B (Kohavi and Longbotham, 2015), nous y reviendrons (Section 4.2.2).

Que ce soit sur les données publiques ou sur les retours d'un système en production, il existe trois familles de mesures pour évaluer les recommandations. Les plus populaires à l'époque du challenge Netflix, dans les années 2000, se fondent sur une mesure de précision de la note, où l'on retrouve les différents scores associés aux problèmes de régression. Cependant ces mesures sont de moins en moins utilisées, car l'important pour un système de recommandation est davantage *l'ordre* − et en particulier l'ordre des première recommandations − que *la valeur* de la note prédite [13]. Les plus couramment utilisés dans la littérature (et particulièrement dans le domaine de la recherche d'information) sont les scores de classification, appelé mesures de support de décision, permettant de quantifier le nombre d'objets pertinents recommandés. Enfin, des mesures plus fines, venant aussi du domaine de la recherche d'information, notamment utilisés lors de la recommandation de pages Web, sont les mesures de rang. Cependant, les mesures de rang comme les mesures de précision présentent un même problème : les recommandations jugées faux positifs peuvent correspondre à de vrais positifs, dans le cas ou la note n'a pas été observée (l'utilisateur aime l'objet mais ne l'a pas vu). Ainsi une pénalisation excessive des faux positifs peut sous-évaluer un système de recommandation.

Dans ce qui suit, les mesures sont définies par utilisateur, puis moyennées sur l'ensemble des utilisateurs.

13. En effet, les erreurs sur de faibles notes n'influencent pas la recommandation finale.

	Pertinent	**Non-pertinent**
Recommandé	Vrai positif	Faux positif
Non-recommandé	Faux négatif	Vrai négatif

TABLE 3.6 – Matrice de confusion de décision

3.3.1 Mesure de précision : prédire la note

Étant donnée la matrice de note $\mathcal{R}$, la première mesure de reconstruction naturelle est celle établie sur le score de régression, évaluant la distance entre notes prédites et notes réelle s :

— *Erreur absolue moyenne*, (MAE pour Mean Absolute Error), donne la moyenne des déviations entre la note prédite et la note réelle :

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |\widehat{r}_{\cdot,i} - r_{\cdot,i}| \tag{3.11}$$

— *Racine de l'erreur quadratique moyenne* (RMSE pour "Root Mean Square Error"), ou distance L2 entre les prédictions et les valeurs réelles (utilisée lors du challenge Netflix) :

$$RMSE = \sqrt{\frac{\sum_i (\widehat{r}_{\cdot,i} - r_{\cdot,i})^2}{N}} \tag{3.12}$$

N représente le nombre d'objets recommandés. $r_{\cdot,i}$ est la note réelle et $\widehat{r}_{\cdot,i}$ la note prédite.

3.3.2 Mesure de support de décision : score de classification

Les mesures de support de décision permettent de quantifier les taux de succès et d'erreur de recommandation, en considérant qu'un système de recommandation génère *une liste* de recommandations d'objets pour chaque utilisateur.

— La **précision** est le rapport entre le nombre de vrais positifs (VP) (i.e. le nombre d'objets recommandés pertinents) et le nombre de recommandations produites (i.e. les vrais positifs plus les faux positifs (FP)). La précision estime donc la probabilité qu'un objet recommandé soit pertinent.

$$\text{Précision@}k = \frac{VP}{VP + FP} \tag{3.13}$$

$$\text{Précision@}k = \frac{|\{\text{objets recommandés parmi les } k \text{ premiers, pertinents}\}|}{|\ \text{k}\ |}$$

— Le **rappel** est le rapport entre le nombre de vrais positifs et le nombre de positifs total, *i.e.* la fraction d'objets pertinents retrouvé dans la liste de recommandation, soit la probabilité qu'un objet pertinent soit recommandé.

$$\textit{Rappel@}k = \frac{VP}{VP + FN} \tag{3.14}$$

$$\text{Rappel@}k = \frac{|\{\text{objets recommandés pertinents}\}|}{|\{\text{objets pertinents}\}|} \tag{3.15}$$

À un seuil k fixé, la valeur de rappel permet de comparer deux systèmes de recommandations. Les courbes de rappel, montrant l'évolution du rappel en fonction du nombre de recommandations sont montrées (Figure 5.1).

— Précision et rappel étant inversement liés, le $F_1 - score$ (F_1) est une mesure combinant précision et rappel :

$$F_1@k = 2 \times \frac{\text{précision@k} \times \text{rappel@k}}{\text{précision@k} + \text{rappel@k}} \tag{3.16}$$

— La **courbe ROC** (receiver operating characteristic) permet de comparer visuellement l'évolution de la pertinence des recommandations en fonction du nombre de recommandations : en abscisse le *taux de faux positifs* $= \frac{FP}{FP+VN}$, en ordonnée le *taux de vrai positifs* $= \frac{VP}{VP+FP}$ (i.e. le rappel), partant du point $(0,0)$ et atteignant $(1,1)$ lorsque tous les objets sont recommandés. Un système de recommandation est d'autant meilleur qu'il maximise l'aire sous la courbe ROC. Un système aléatoire correspond à la première bissectrice, un système parfait correspond à une courbe passant par le point (0,1), recommandant ainsi tous les objets pertinents avant les objets non-pertinents.

3.3.3 Mesure de rang

Comme la qualité des premières recommandations est plus importante que celles des dernières recommandations, en plus de considérer le rappel pour les premières valeurs, nous pouvons utiliser des mesures de rang pour évaluer l'ordre des premières recommandations. Ces mesures issues du domaine de recherche d'information peuvent être adaptées au cas des systèmes de recommandations :

— **NDCG** : (Normalized Discouted Cumulative Gain, ou gain cummulé actualisé et normalisé) (Järvelin and Kekäläinen, 2002) mesure la qualité de l'ordre en prenant en compte essentiellement le rang attribué aux objets les plus pertinents [14]. Elle mesure ainsi le gain qu'apporte chaque document en fonction de sa position dans la liste des résultats

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

où rel_i correspond au degré de pertinence de la i^{eme} recommandation. NDCG est la normalisation de DCG par sa valeur maximale, donnant ainsi un score entre $[0,1]$.

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

— **MRR** (Mean Reciprocal Rank) (Voorhees et al., 1999) est l'inverse du rang du premier objet pertinent recommandé, moyenné sur l'ensemble des utilisateurs. Formellement, le rang du premier objet pertinent peut s'écrire

$$rr_u = \min_{i \in \mathcal{O} | \mathcal{R}^{test}_{u,i}=1} rang(i)$$

La moyenne sur tous les utilisateurs donne ainsi :

$$MRR = \frac{1}{|\mathcal{U}|} \sum_{u=1}^{|\mathcal{U}|} \frac{1}{rr_u}$$

14. Notons que cette mesure est essentiellement utilisée dans le cas où il existe plusieurs degrés de pertinence, par exemple {peu pertinents, assez pertinent, très pertinent}

3.3.4 Autres mesures

Comme mentionné par Burke (2002); Herlocker et al. (2004), ces mesures de performance ne suffisent pas à caractériser entièrement un système de recommandation. D'autres critères sont également définis :

— La *couverture* correspond à la fraction des données pour lesquelles le système de recommandation peut prédire des recommandations. Par exemple les systèmes de filtrage collaboratif ne couvrent pas le cas des nouveaux utilisateurs et des nouveaux objets.

— La *complexité*, le *temps de calcul* et le *temps de réponse* mesurent la rapidité avec laquelle les recommandations sont générées. Dans certaines applications, surtout pour la recommandation en ligne, le temps de réponse peut être un paramètre critique.

— La *nouveauté* ou la *sérendipité* mesure à quel point le système de recommandation suggère des objets nouveaux ou inconnus ; cela correspond au facteur d'exploration de la recommandation.

Chapitre 4

Description des données

Ce chapitre est consacré aux jeux de données étudiées, décrivant les données publiques et les données propriétaires sur lesquelles sont fondé nos recherches.

4.1 Introduction

Il existe de nombreux jeux de données utilisés pour évaluer les systèmes de recommandation. Parmi les plus connus, citons ceux du challenge Netflix qui ont lancé le domaine des systèmes de recommandations, ainsi que les données de MovieLens [1], de Twitter ou d'Amazon. D'autres données portent sur la recommandation de musique (`last.fm`) ou de blagues (`jester.com`). Ces données ne sont pas considérées dans la suite, dans la mesure où ni la description des objets ni celle des utilisateurs ne fait intervenir de description textuelle.

Deux jeux de données, directement pertinents pour notre étude, sont ceux des challenges RecSys 2016 et 2017 (Section 4.2.2). Ces jeux de données publiques concernent également la recommandation d'offres d'emploi et de CVs. Cependant, compte tenu des contraintes de confidentialité, ces jeux de données sont anonymisés, remplaçant les mots par des identifiants numériques.

Un autre jeu de données est celui de CiteUlike [2] (Section 4.2.1) décrivant pour une communauté de chercheurs l'ensemble des articles enregistré dans leur bibliothèque virtuelle sur la plateforme `CiteUlike.org`.

Enfin et surtout, notre recherche a porté sur deux jeux de données propriétaires :

— Dans le domaine des emplois à bas salaires, nous avons bénéficié des données gracieusement mises à notre disposition par la Société *Qapa.fr* (Section 4.2.3), agence d'intérim sur Internet. Ces données décrivent les interactions entre les utilisateur de la plate-forme et les offres d'emplois postés. Ces données permettent de constituer deux jeux de données, le premier s'étalant sur 4 ans, de 2011 à 2015, le second sur 3 mois, de mars à mai 2015. La distinction entre ces deux jeux de données est due au fait que le premier jeu de données ne comprend pas les compétences attachées aux offres et aux CVs. Nous utiliserons seulement le second jeu de données par la suite.

— Dans le domaine de l'emploi à forte valeur ajoutée, nous avons disposé des données fournies par l'Association Bernard Gregory (ABG) (Section 4.2.4). Les données comprennent les

1. `https://grouplens.org/datasets/movielens/`
2. `http://www.citeulike.org/faq/data.adp`

interactions de PhDs et de recruteurs dans l'industrie et dans les instituts de recherche, couvrant une période de 4 ans, de 2010 à 2014.

		Objets		Utilisateurs	$\mathcal{R}$	
	Date	# objets	# mots	# utilisateurs	# clics (densité)	valeurs
RecSys	3 mois	1 497 k	120	1 306 k	314 501 k (.002%)	multiple
CiteUlike	?	16.9 k	93	5.5 k	205 k (.217%)	"a enregistré"
Qapa	3 mois	56 k	175	30 k	226 k (0.013%)	"a postulé"
ABG	3 ans	10.7 k	213	8.9 k	65 k (0.067%)	"a enregistré"

TABLE 4.1 – Description des jeux de données étudiés, Qapa et ABG sont des données propriétaires. Excepté pour CiteUlike, la densité des données collaboratives (ratio du nombre d'information dans la matrice collaborative) est très faible ; à titre de référence, les données du challenge Netflix ont également une densité de .01%. "A enregistrer" signifie que la note correspond au fait d'avoir sauvegarder la page contenant l'offre.

4.2 Jeux de données

4.2.1 CiteUlike

Une base de données bien étudiée par la communauté des systèmes de recommandation est le benchmark *CiteUlike.org*. Le site CiteULike permet à chaque utilisateur (5 551 utilisateurs dans la base) d'enregistrer dans sa bibliothèque virtuelle un ensemble d'articles scientifiques (16 980 articles en tout). La matrice collaborative décrit les sous-ensembles d'articles formant la bibliothèque virtuelle de chaque utilisateur, contenant 204 986 paires utilisateur, article, ayant ainsi une densité de .217%. Chaque article est décrit par un sac de mots, formé à partir de son titre et son résumé, contenant 67 mots en moyenne. Comme dans (Wang and Blei, 2011), les doublons sont éliminés, les articles vides sont retirés ainsi que les utilisateurs ayant moins de 10 articles dans leur bibliothèque.

En moyenne, chaque utilisateur compte 37 articles dans sa bibliothèque. La taille complète du corpus est de 1,6 million de mots. Seuls les 8 000 mots les plus fréquents (après élimination des mots vides, en anglais) sont retenus. Un article apparaît en moyenne dans 12 bibliothèques et 97% des articles apparaissent dans moins de 40 bibliothèques.

4.2.2 ResSys

Le groupe *Systèmes de Recommandation* (RecSys) de l'*Association of Computational Machinery* organise un challenge en recommandation tous les ans dans le cadre de la conférence RecSys. En 2016 [3] et 2017 [4], le challenge, co-organisé avec le site de recrutement en ligne Xing.com, a porté sur la recommandation dans le domaine de l'emploi (Abel et al., 2017).

Le challenge 2017 auquel nous avons participé comprend deux phases :

— **Évaluation hors ligne** (5 mai - 18 avril 2018) : Selon la méthodologie classique pour les challenges de machine learning, les données sont fixées. Un ensemble de validation et un ensemble test sont mis à part ; l'ensemble de validation est utilisé pour calculer le tableau de

3. `http://2017.recsyschallenge.com/`
4. `http://2016.recsyschallenge.com/`

	nombre	avec interactions	moyenne de mots	$n_{\mathcal{V}}$	catégories
Utilisateur	1 497 020	500 k	29	13 k	7
Objets	1 306 054	888 k	32	99 k	15

TABLE 4.2 – Données RecSys.

bord et les rangs des participants au cours de la période de soumission. Six semaines après le début du challenge, l'ensemble de test est utilisé pour calculer le score final de chaque participant ainsi que les rangs. La tâche consiste à recommander, pour chaque *nouvelle* offre d'emploi, les 100 utilisateurs les plus pertinents. Les 20 meilleures équipes au terme de cette première évaluation sont qualifiées pour la seconde phase, qui se déroule en ligne. La phase hors ligne s'étend du 5 mars au 16 avril.

— **Évaluation en ligne** (1er mai - 4 juin) : De nouvelles offres sont mises sur le site chaque jour. Les 20 équipes reçoivent chaque soir la liste des offres et ont alors une journée pour proposer *une* offre à chaque utilisateur. Le score est le retour (feedback) *effectif* des utilisateurs, ayant décidé ou non de cliquer sur les offres soumises [5]. Le score final de chaque équipe est la somme du score de leurs 2 meilleures semaines.

L'ampleur du challenge et la validation en ligne par des utilisateurs réels est à notre connaissance sans précédent dans le contexte des challenges du domaine. En effet, la mise à disposition d'une API pour intégrer les utilisateurs "dans la boucle" fait rarement partie des challenges classiques [6]. La phase en ligne offre un moyen de tester les approches proposées de manière dynamique, comparant les différentes approches sur plusieurs groupes indépendants d'utilisateurs. Cela est très proche des tests A/B (Kohavi and Longbotham, 2015) : dans notre cas, les utilisateurs actifs sont séparés en 20 groupes homogènes, chaque groupe recevant les recommandations d'une équipe. Les algorithmes sont ainsi comparés de manière équitable. Les recommandations sont directement proposées par le système de production en temps réel de *XING*. Cette mise en oeuvre permet de simuler le processus de développement, au niveau de production, d'un système de recommandation.

De même que pour Qapa et ABG, les offres et CVs disposent d'un champ texte libre, ainsi que plusieurs données catégorielles décrites plus bas. Cependant, ces données textuelles sont extensivement retraitées et encodées pour préserver la confidentialité des informations.

Description des données [7]

— **Description des utilisateurs :** 1 497 020 utilisateurs sont décrits. Leur description est formée des éléments suivants :
  — un ensemble de "tags" correspondant aux termes extraits dans le corps du texte, sous forme d'identifiants numériques pour des raisons d'anonymisation ;
  — 7 caractéristiques démographiques : le type d'emploi (temps plein, intérim), la fonction (consultant, RH), le niveau de carrière, le type d'entreprise (automobile, finance), le niveau d'éducation et le nombre d'expériences. La valeur de ces caractéristiques est un intervalle (par exemple pour le niveau de carrière : < 1 an, entre 1 et 3 ans, entre 3 et 6 ans, plus de 7 ans, Nan = valeur manquante), ou une valeur catégorielle (par exemple pour le type d'entreprise : 1 pour industrie, 2 pour économie) ;

5. ce score peut être vu comme le rappel à 1.
6. Comme par exemple ceux organisés par *Kaggle.com*.
7. `http://2017.recsyschallenge.com/dataset`

— 2 informations entreprises : le fait qu'il s'agisse d'un client "premium" et une estimation de mobilité : si Xing estime que l'utilisateur veut changer de job. Cette dernière information n'apparaît que dans la phase hors ligne.
— **Description des objets :** 1 306 054 offres d'emplois sont disponibles. Chaque offre est décrite par deux ensembles de termes, respectivement extraites du titre de l'offre et de sa description. La taille du vocabulaire est de 10 000 termes. Les objets disposent des mêmes type d'informations que les utilisateurs, auxquelles s'ajoute une date de mise en ligne et la localisation de l'emploi.
— **Interactions :** Nous disposons d'annotations pour une partie (0.001%) des paires (u, i), prenant 6 valeurs :
  1 L'offre a été montrée à l'utilisateur (314 501 k) ;
  2 L'utilisateur a cliqué sur l'offre (6 867 k) ;
  3 L'utilisateur a enregistré l'offre (281 k) ;
  4 L'utilisateur a postulé à l'offre (118 k) ;
  5 L'utilisateur a supprimé l'offre de son écran (906 k) ;
  6 Le recruteur (associé à l'offre) a consulté le profil de l'utilisateur (100 k).

Les offres sont postées entre février 2016 et février 2017 ; la localisation est internationale, avec 85% des offres en Allemagne.

Pour des raisons d'anonymisation, les données sont bruitées :
— une fraction inconnus des valeurs binaires est modifiée aléatoirement ;
— un nombre inconnu d'utilisateurs et d'objets artificiels est ajouté ;
— une fraction inconnue d'interactions est perturbée ;
— certains attributs catégoriels sont perturbés ou mis à *Nan*.

Ces manipulations − rendues nécessaires pour la protection des utilisateurs − empêchent les interprétations des modèles et/ou l'usage de modèles pré-entraînés de langue, ou de connaissances a priori.

Score d'évaluation Le score est un score de rappel à 100 pondéré, où chaque recommandation reçoit une récompense de :
— +1 si l'utilisateur a cliqué sur l'offre ;
— +5 si l'offre a été enregistrée ou si l'utilisateur a postulé
— +20 si le recruteur a consulté le CV ;
— **-10** si l'utilisateur a retiré l'offre des suggestions de sa page principale.

De plus, le score associé aux utilisateurs premium est doublé. Enfin, si une offre (respectivement une offre payée) est cliquée par au moins un utilisateur, le score est augmenté de 25 points (resp. 50 points). Cette mesure reflète la fonction de valeur pour le site `Xing.com`. Ce calcul de score présente une spécificité par rapport aux scores classiques, il fait intervenir une valeur négative parmi les récompenses, notons que 8% des interactions sont de type négatifs.

Évaluation hors ligne Pour l'épreuve hors ligne, 50 000 offres sont désignées comme objets tests, il s'agit d'un problème de démarrage à froid objet. La tâche consiste à recommander chacune de ces offres aux 74 840 utilisateurs désignés comme utilisateurs tests. Un tiers de ces utilisateurs tests sont des nouveaux venus (démarrage à froid complet) et 11 000 n'ont qu'une seule interaction. Chaque offre test peut être recommandée à au plus 100 utilisateurs [8]. En pratique, chaque équipe

8. Nous recommandons ici des utilisateurs aux *objets* (contrairement au cas standard). Le rappel à 100 est calculé par colonne et un utilisateur test peut donc recevoir entre 0 et 50 000 recommandations.

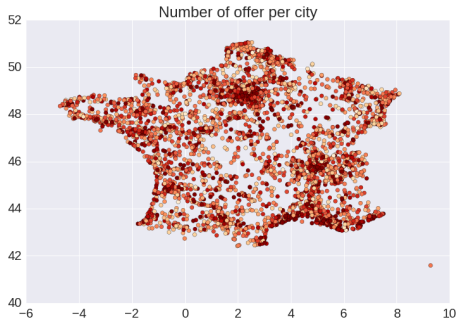


FIGURE 4.1 – Le nom de ville associé aux offres d'emplois *Qapa* permet de retrouver leurs coordonnées géographiques. Ces informations (longitude et latitude) sont utilisées pour compléter la description vectorielle de l'offre ((Section 6)), et pour visualiser la densité géographique des offres. Cette carte restreinte à la France métropolitaine montre une plus grande densité d'offres d'emploi dans les zones à forte population.

peut faire au plus 20 soumissions par jour, chaque fichier de soumission comprend 50 000 entrées (correspondant aux offres test) avec chacune au plus 100 identifiants (désignant les utilisateurs tests).

Évaluation en ligne Cette fois, contrairement à l'évaluation hors ligne, la tâche est de recommander les nouvelles offres d'emploi aux utilisateurs, sur un rythme journalier. Les 20 équipes finalistes se partagent l'ensemble des utilisateurs tests (nombre variable) et reçoivent, chaque soir, la liste des offres publiées dans la journée (nombre variable). Puis elles disposent d'une journée pour proposer *une* offre à chaque utilisateur. Le score est le retour (feedback) *effectif* de ces utilisateurs. Ce score peut être vu comme le rappel à 1, pondéré. Cette phase dure 4 semaines.

La phase en ligne ne comprend donc pas de données test définies par les organisateurs pour lesquelles la vérité terrain est connu : l'évaluation est fonction des cliques des utilisateurs de la plateforme en temps réel. Ainsi que dit plus haut, il s'agit à notre connaissance de la première fois où les systèmes de recommandation d'offres d'emploi sont évalués par les utilisateurs en temps réel lors d'un challenge.

4.2.3 Qapa

La société Qapa, fondée en 2011, est la première plateforme digitale de recrutement intérim en France [9], avec 4.5 millions d'utilisateurs (200 000 nouveaux candidats inscrits chaque mois) et 5 millions d'offres d'emplois (300 000 entreprises se sont inscrites sur qapa.fr, principalement des TPE et des PME). Leur marketing se fonde sur un algorithme d'appariement [10] entre offres d'emplois et CVs.

Nous avons considéré deux jeux de données : le premier Qapa* portant sur une période de 4 ans, de 2011 à 2015, comprend 344 000 CVs et 906 000 offres avec 7.2 millions d'interactions de

9. Entre 8 000 et 12 000 personnes trouvent un emploi ou un stage sur qapa.fr chaque mois.
10. Cet algorithme de matching a obtenu en décembre 2013 le Prix Spécial de l'Intelligence Economique JIEE de l'Ecole Polytechnique.

type "à postulé". Cependant, nous n'utiliserons pas ces données dans nos recherches car il n'y a pas de "compétences" pour les offres. De plus, le traitement de cette masse de données demande des compétences complémentaires à l'apprentissage automatique pour être aisément traité. Le deuxième jeu de données, noté Qapa, correspond à la période de mars 2015 à mai 2015. [11]

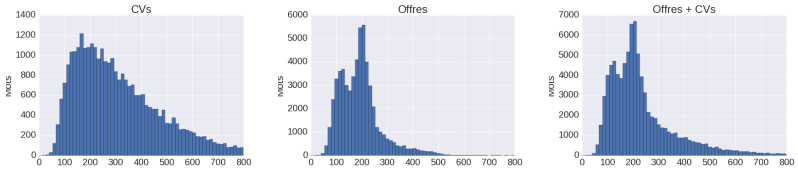
Description des offres d'emploi et des CVs Les éléments inclus dans un document (offre d'emploi ou CV) sont :

— Un *métier* par offre ou une *liste* de métiers par CVs ;
— Une liste de *compétences* ;
— Une *description* en texte libre ;
— Une localisation (nom de ville) et une date de création, pour les offres uniquement.

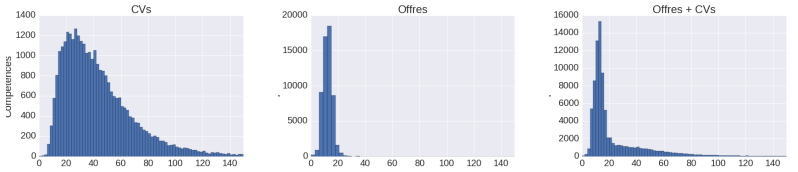
Le métier est un terme standardisé, figurant dans la liste des métiers décrits par Pôle-emploi, nécessairement épicène, c'est à dire non marqué du point de vue du genre. Par exemple :

```
Responsables de magasin; Secrétaire; Secrétaire médical;
Assistant / Assistante de gestion administrative;
Directeur / Directrice de projet en informatique;
Chauffeur-livreur / Chauffeuse-livreuse
```

Le nom de la ville associé aux offres permet de retrouver les coordonnées géographiques du travail proposé avec une grande probabilité ($> 90\%$). L'ensemble des offres peut ainsi être représenté géographiquement (Fig. 4.1).



(a) Distribution du nombre de mots



(b) Distribution du nombre de compétences

FIGURE 4.2 – Histogramme du nombre de mots et de compétences pour les CVs (gauche) et les offres (milieu) et pour les deux (droite) dans *Qapa*. Les utilisateurs ont tendance à être plus détaillés, les recruteurs plus concis et spécifiques.

Chaque offre ou CV est accompagné d'une description textuelle d'une centaine de mots. Les statistiques des documents sont récapitulées (Figure 4.2a). Chaque offre compte en moyenne 14 compétences. Les utilisateurs peuvent aussi attacher des compétences à leur profil, et sont plus

11. Ces données correspondent à l'ensemble noté A^* dans l'article Schmitt et al. (2017)

enclins à mettre le plus de compétences possibles (27 en moyenne) (Figure 4.2b). Quelques offres représentatives sont données en annexe (Table C.1), ainsi que notre analyse subjective comparant les thème identifié par LDA (Section 2.2.4) et LSA (Section 2.2.3) sur ces offres (Section C).

Dans la suite, la liste des compétences présentes est ajouté au sac de mot de l'offre ou du CV. Un aperçu du vocabulaire employé est donné ci-dessous sous forme de nuage de mots, où la police des mots est proportionelle à leur fréquence *tf* ou à leur pondération tf-idf.

Description des interactions Outre les descriptions des offres et des CVs, la base de données comprend la matrice des interactions entre utilisateurs et offres d'emploi, indiquant pour chaque utilisateur la liste des offres qu'il a consultées, ainsi que celles auxquelles il a postulé. Seules les interactions correspondant à l'action de postuler sont conservées dans la suite pour garantir la qualité des données collaboratives. Nous retirons également les CVs et les offres ayant moins de deux interactions. La matrice est ainsi constituée de 30 669 utilisateurs et 56 512 offres. Chaque utilisateur (respectivement chaque offre) a 7 interactions en moyenne (resp. 4).

Pour chaque utilisateur, l'ensemble des offres qu'il a sélectionnées délimitent une zone géographique, ainsi qu'un intervalle de temps. Ainsi, l'écart des dates de publication (respectivement la distance géographique) de l'ensemble des offres sélectionnées par les utilisateurs est représenté (Figure 4.4a) (resp. (Figure 4.4b)). La courbe bleu (Figure 4.4a) montre que l'écart maximal entre les offres sélectionnées varie entre 1 jours et 70 jours pour 90% des utilisateurs. 40% des utilisateurs restent moins de 40 jours sur la plate-forme. La dispersion géographique moyenne (respectivement maximal) entre les offres sélectionnées est de 50 km (resp. 200 km) pour 80% des utilisateurs (Figure 4.4a). Cela signifie que 80% des utilisateurs cherchent du travail dans un rayon moyen de 25 km, s'écartant au maximum de 100 km de chez eux.

4.2.4 ABG

Le deuxième jeu de données provient de l'Association Bernard Gregory [12] (ABG) [13], dont l'objectif est de favoriser le rapprochement entre les mondes économique et académique, ainsi que de faciliter la mobilité professionnelle des PhD.

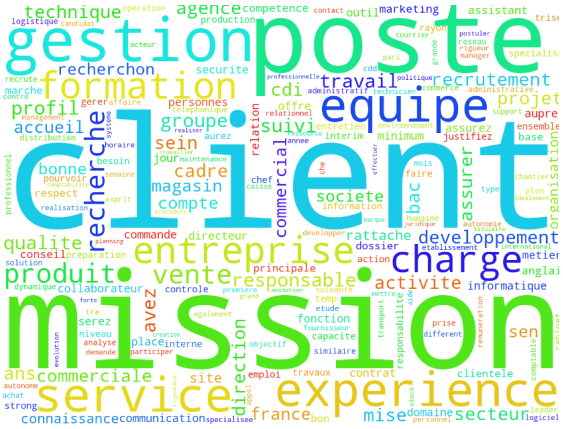
La plateforme réalisée par ABG héberge ainsi des offres de thèse dans l'industrie ainsi que des offres d'emplois dans l'industrie destinées à des PhD. La base de données dont nous avons disposé comprend l'ensemble des offres et des CVs enregistrés sur le site sur la période de 2011 à 2015. Une offre ou un CV est constitué de champs textes [14], augmentés d'une localisation et d'une date de publication dans le cas des offres. Un nuage de mots des offres est donnée en annexe (Figure D.1) Les CVs sont décrits par les champs suivants :

```
1    Titre du profil, Expertise Scientifique, Expertise Technique, Domaine d'
     application des recherches, Enjeux, Savoirs faire organisationnels, Intitulé du
     diplôme
```

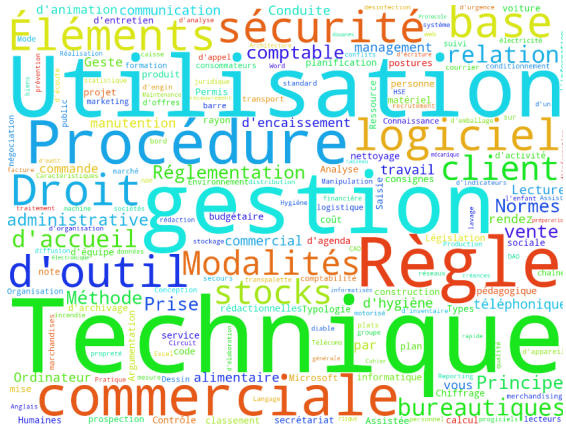
Les offres d'emploi sont décrites par les champs suivants :

```
1    Type, Nature du contrat, Nom établissement recruteur,Domaine compétence 1,
     Domaine compétence 2, Profil des candidats, Mission
2
```

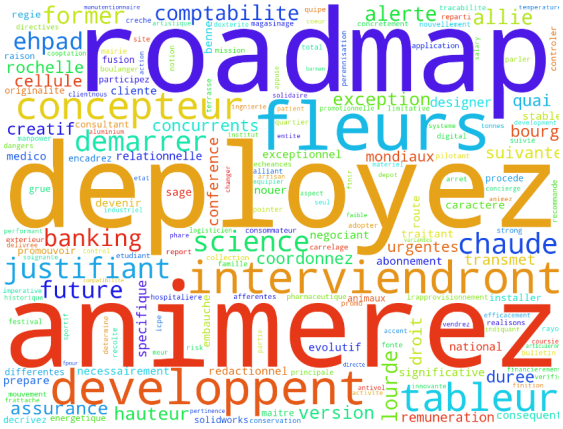
12. `http://www.intelliagence.fr/`
13. Ces données correspondent à l'ensemble noté B^* dans l'article Schmitt et al. (2017)
14. Certains champs sont rédigés en français, d'autres en anglais, et certains sont en français *et* en anglais (cf offre 1 en annexe (Table D.2)).



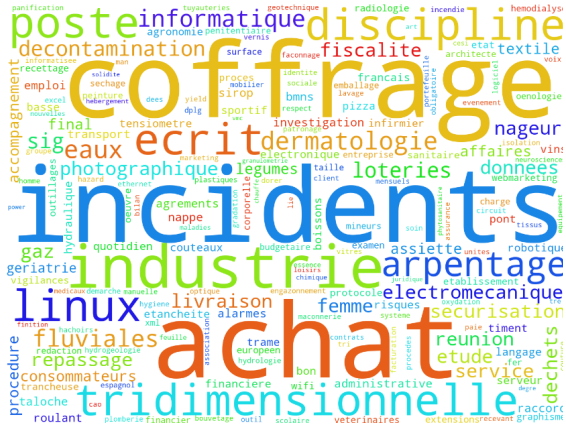
(a) Mots (tf) apparaissant dans les *descriptions* des offres d'emploi.



(b) Mots (tf) apparaissant dans les listes de *compétences* des offres d'emploi.

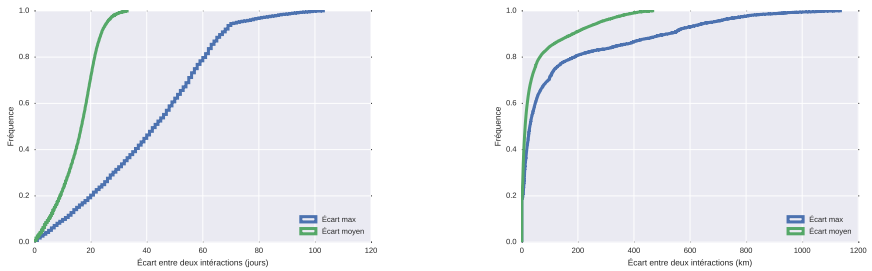


(c) Mots *clefs* (tf-idf) apparaissant dans les *descriptions* des offres d'emploi.



(d) Mots *clefs* (tf-idf) apparaissant dans les listes de *compétences* des offres d'emploi.

FIGURE 4.3 – Nuage de mots pour les *offres* chez Qapa. La taille des mots est proportionnelle à la fréquence (haut) ou au tf-idf (bas). À gauche sont représenter les mots apparaissant dans les descriptions des offres. À Droite les mots apparaissant dans les compétences associées aux offres. Notons que le nuage de mots est construit à partir des unigram.



(a) Histogramme cumulé de l'écart temporel entre deux interactions

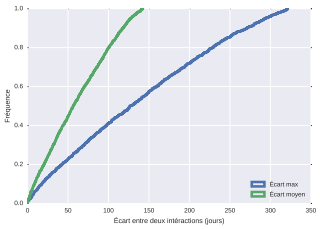
(b) Histogramme cumulé de l'écart géographique entre deux interactions

FIGURE 4.4 – Histogramme cumulé des écarts entre les offres sélectionnés par un même utilisateur. La moitié des utilisateurs restent moins de 42 jours sur la plate-forme Qapa.fr.

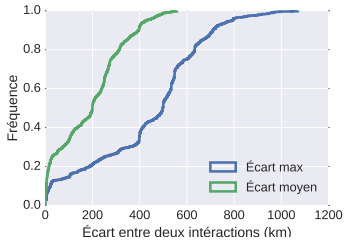
Dans le cas des offres et des CVs, les champs textes sont agrégés en un sac de mots global, suivant ainsi le même format que les données *Qapa*. Une option plus fine est de considérer indépendamment les différents sacs de mots correspondant aux divers champs, conservant ainsi la structure des documents. Cette option est considérée dans les perspectives de recherche ouvertes ; en effet, compte tenu de la petite taille des textes répondant à un champ donné, des méthodes spécifiques sont nécessaires pour en extraire l'information. De même que pour les données Qapa, nous disposons des données correspondant à la date de publication de l'offre ainsi que le nom de la ville.

Création de $\mathcal{R}$ La base de données ABG comprend également l'ensemble des offres d'emploi mises par un utilisateur dans son panier, définissant la matrice collaborative $\mathcal{R}$. Nous notons $\mathcal{R}_{u,i} = 1$ si l'utilisateur u a mis l'offre i dans son panier, $\mathcal{R}_{u,i} = 0$ sinon. Là encore, les offres et les utilisateurs ayant moins de deux interactions sont éliminés. Finalement, la matrice $\mathcal{R}$ comprend 8 981 utilisateurs $\times$ 10 768 objets et 64 787 interactions. En moyenne, les paniers sont constitués de 7 offres et chaque offre apparaît dans 6 paniers d'utilisateurs.

La distribution géographique et temporelle des offres sélectionnées par un même utilisateur est représentée sous forme d'histogramme cumulé (Figure 4.5). Nous remarquons que les écarts de distance ainsi que les écarts temporels sont bien plus important que dans les données Qapa (Figure 4.4). En effet, un utilisateur est amené à chercher un post-doc plus longtemps qu'une offre chez Qapa. 50% des utilisateurs sélectionnent des offres dont les dates de publication ont plus de 100 jours d'écart. L'écart géographique plus important que chez Qapa peut s'expliquer par le fait que les PhD sont facilement amenés à postuler à l'international.



(a) Histogramme cumulé de l'écart temporel entre deux interactions



(b) Histogramme cumulé de l'écart géographique entre deux interactions

FIGURE 4.5 – Histogramme cumulé des écarts entre les offres sélectionnés par un même utilisateur sur la plate-forme ABG. La moitié des utilisateurs postule à une offre dans un rayon de 500 km.

Chapitre 5

Analyse des données

Dans ce chapitre, nous examinons dans un premier temps la qualité des informations collaboratives (Section 5.1). Dans un second temps nous comparons les méthodes de représentation des documents textuels (Section 5.2.1) puis nous évaluons la qualité des textes associés aux offres et aux CVs (Section 5.2) à travers les méthodes de recommandations standard (Section 3). Enfin, nous visualisons les données de Qapa portant sur les métiers les plus populaires (Section 5.3).

5.1 Qualité de $\mathcal{R}$

Dans cette partie nous évaluons la *qualité* des données collaboratives $\mathcal{R}$, vérifiant la robustesse de ces données. Le but est d'apprécier si ces informations permettent de générer des recommandations adaptées.

5.1.1 Qualité du filtrage collaboratif

Nous commençons par évaluer les données collaboratives en terme des performances obtenues par filtrage collaboratif à base de mémoire (comme étant la méthode la plus classique, la plus facile à implémenter et finalement la plus robuste) (Section 3.2.1). Si les données sont trop bruitées, c'est à dire que les interactions des utilisateurs sur les objets sont incohérentes, les résultats de recommandations seront pauvres. La qualité des scores de recommandation obtenus mesure ainsi la cohérence des données.

Méthodologie démarrage à chaud Les données sont classiquement divisées en un ensemble d'entraînement et un ensemble de test. Les données de test $\mathcal{R}^{test}$ sont constituées en sélectionnant de manière uniforme 10% des notes présentes dans la matrice $\mathcal{R}$. La matrice $\mathcal{R}^{train}$ est obtenue à partir de $\mathcal{R}$ en remplaçant ces notes par 0. Rappelons ici la méthode à base de mémoire :

- À partir de $\mathcal{R}^{train}$, nous calculons les similarités objet−objet ou utilisateur−utilisateur (Équation 3.3) ;
- À l'aide de ces similarités et de la convolution avec $\mathcal{R}^{train}$ nous obtenons un score pour chaque couple (utilisateur,objet) (Équation 3.6) ;
- Chaque ligne de la matrice de score est triée par ordre décroissant ;

— Pour chaque utilisateur (ligne), les objets sont recommandés par score décroissant parmi la liste des objets disponibles. La difficulté du problème augmente naturellement avec le nombre d'objets.

Enfin, nous évaluons la qualité de recommandation à l'aide du score de rappel. Les résultats sont reportés (Figure 5.1) et (Table 5.1 et 5.2). Chaque étude est répétée sur dix expériences indépendantes.

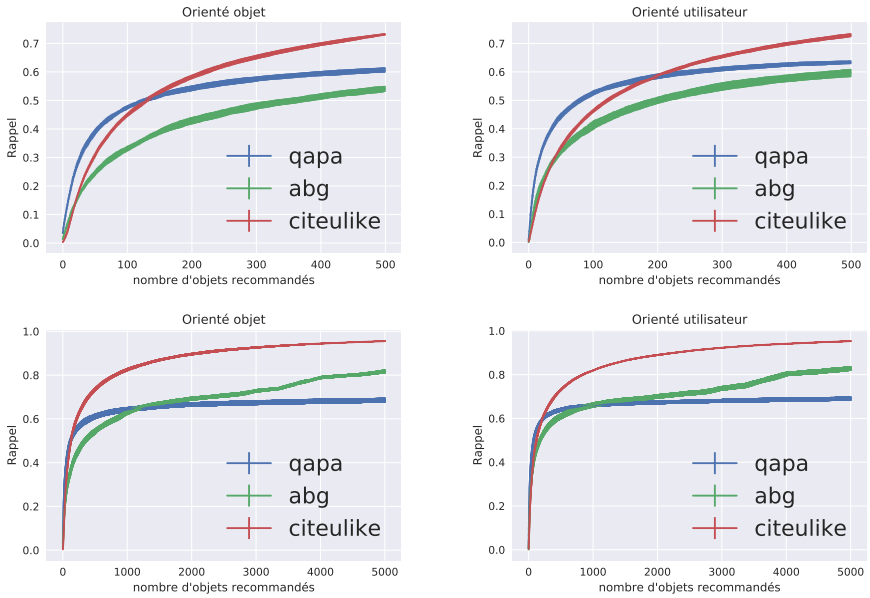


FIGURE 5.1 – Courbe de rappel pour les trois jeux de données : Qapa (en bleu dans toute la suite du manuscrit), ABG (vert) et CiteUlike (rouge), cas orienté objet (gauche) et orienté utilisateur (droite). Les figures du haut (respectivement du bas) sont un zoom sur les 500 (resp. 5 000) premières recommandations. Le comportement sur ces trois jeux de données est semblable : le rappel monte rapidement pour les 100 premiers rappels, ce qui traduit une bonne qualité de recommandation. Après un millier de recommandations, la pente correspond à une recommandation aléatoire. Les barres d'erreurs correspondent à l'épaisseur du trait pour dix expériences indépendantes. Certaines valeurs sont reportées (Table 5.1 et 5.2).

Résultats Pour les données de Qapa, la tâche est de retrouver, pour chacun des 30 669 utilisateurs, le ou les clics de l'ensemble de test parmi les 56 512 objets disponibles. Une recommandation aléatoire obtient ainsi un score de rappel à 100 de l'ordre du dixième de la densité, soit 0.001. Pour ABG, une recommandation aléatoire pour chaque utilisateur, recommandant 100 objets parmi les 10 768 présents obtient un score de rappel à 100 de 0.007. Pour CiteUlike, qui contient un ensemble de données collaboratives plus dense, une recommandation aléatoire sélectionnant 100 objets sur les 16 980 obtient un score de rappel à 100 de 0.022.

Ces courbes montrent que pour Qapa et Citeulike, les clics pertinents sont en moyenne parmi les 120 premières recommandations. Nous concluons que les données collaboratives associées à ces

	qapa	abg	citeulike
Rappel@20	0.255 ± 0.012	0.144 ± 0.007	0.141 ± 0.005
Rappel@100	0.479 ± 0.011	0.334 ± 0.01	0.453 ± 0.007
Rappel@200	0.546 ± 0.01	0.429 ± 0.01	0.589 ± 0.005

TABLE 5.1 – Score orienté objet correspondant à la Fig. 5.1. La première case du tableau se lit : Dans les 20 premières recommandations, nous retrouvons 25,5 % des notes de l'ensemble de test pour les données de Qapa, *i.e.* nous recommandons en moyenne les objets pertinents parmi les 20 premières recommandations pour 9 000 utilisateurs sur les 36 000.

	qapa	abg	citeulike
Rappel@20	0.315 ± 0.011	0.208 ± 0.012	0.179 ± 0.008
Rappel@100	0.524 ± 0.012	0.417 ± 0.013	0.463 ± 0.007
Rappel@200	0.587 ± 0.009	0.502 ± 0.009	0.588 ± 0.007

TABLE 5.2 – Score orienté utilisateur correspondant à la Fig. 5.1. Tous ces scores sont supérieurs aux scores orienté objets (à l'exception du rappel à 200 pour CiteUlike qui est égal).

deux jeux de données sont de très bonne qualité. La qualité des données collaboratives provenant d'ABG est moins bonne, nécessitant 200 recommandations pour satisfaire la moitié des utilisateurs, cela est expliqué en partie par la longue durée sur laquelle s'étendent les interactions.

Pour Qapa, après environ 800 recommandations, le tri de recommandation est aléatoire (cf courbe (quasi) horizontale (Figure 5.1)). De même, après 1 000 recommandations, la courbe devient celle d'une recommandation aléatoire pour CiteUlike et ABG. Les meilleurs résultats sont obtenus pour une approche orientée utilisateur, cela peut être expliqué par le fait que les utilisateurs ont en moyenne plus d'interactions que les objets, permettant ainsi une meilleure similarité entre utilisateurs. Dans le cas orienté objet, le phénomène de recommandation aléatoire vient du fait que les objets pertinents dans l'ensemble de test ne sont pas similaires [1] aux objets sélectionnés par l'utilisateur de l'ensemble d'entraînement. Ces résultats sont très encourageants sur la qualité des données ; leur portée est limitée bien sur par le contexte de démarrage à chaud.

La méthode fondée sur la factorisation de la matrice "SVD" n'est pas représentée ici, car elle n'obtient pas de meilleurs résultats. Une autre approche à base de mémoire se fonde sur la factorisation de la matrice $\mathcal{R}$ pour calculer les similarités, mais cette approche non plus n'améliore pas les scores.

5.1.2 Graphe des interactions et petit monde

Une analyse complémentaire des données collaboratives considère le graphe de voisinage défini sur l'ensemble des objets. Formellement deux objets sont voisins s'ils partagent au moins une note d'un même utilisateur. Ces voisinages induisent un graphe où les objets sont les noeuds et les relations de voisinage sont les arcs. Nous pouvons caractériser ce graphe par la distribution de ses degrés d'arités et par son diamètre. Pour avoir une mesure plus fine que le diamètre, nous proposons la méthode suivante :

1. N'ayant aucune notes communes dans $\mathcal{R}^{train}$, deux objets ont toujours une similarité nulle.

Connectivité Partant d'un utilisateur u, nous avons la liste des offres qu'il a sélectionnées $\mathcal{O}_1(u)$. Notons $\mathcal{O}_{n+1}(u)$ les objets voisins des objets $\mathcal{O}_n(u)$:

$$\mathcal{O}_{n+1}(u) = \bigcup_{v \in \mathcal{O}_n(u)} \mathcal{O}_1(v)$$

Ainsi $\mathcal{O}_1(u) \subset \mathcal{O}_2(u) \subset \ldots \subset \mathcal{O}_r(u)$. Notons que la dernière inclusion non triviale donne le diamètre du graphe. Pour comparer ces valeurs, nous créons deux jeux de données *mélangés*. Dans le premier mélange *rndr* nous appliquons pour chaque utilisateur une permutation sur l'ensemble des objets qu'il a notés. Le second mélange *rnd** est une matrice dont les valeurs sont tirées uniformément. Nous conservons toujours la taille et la densité de la matrice collaborative, cependant les données sont aléatoires et l'information est perdue. L'évolution des ensembles $(\mathcal{O}_n(u))_n$ est donnée (Figure 5.2)

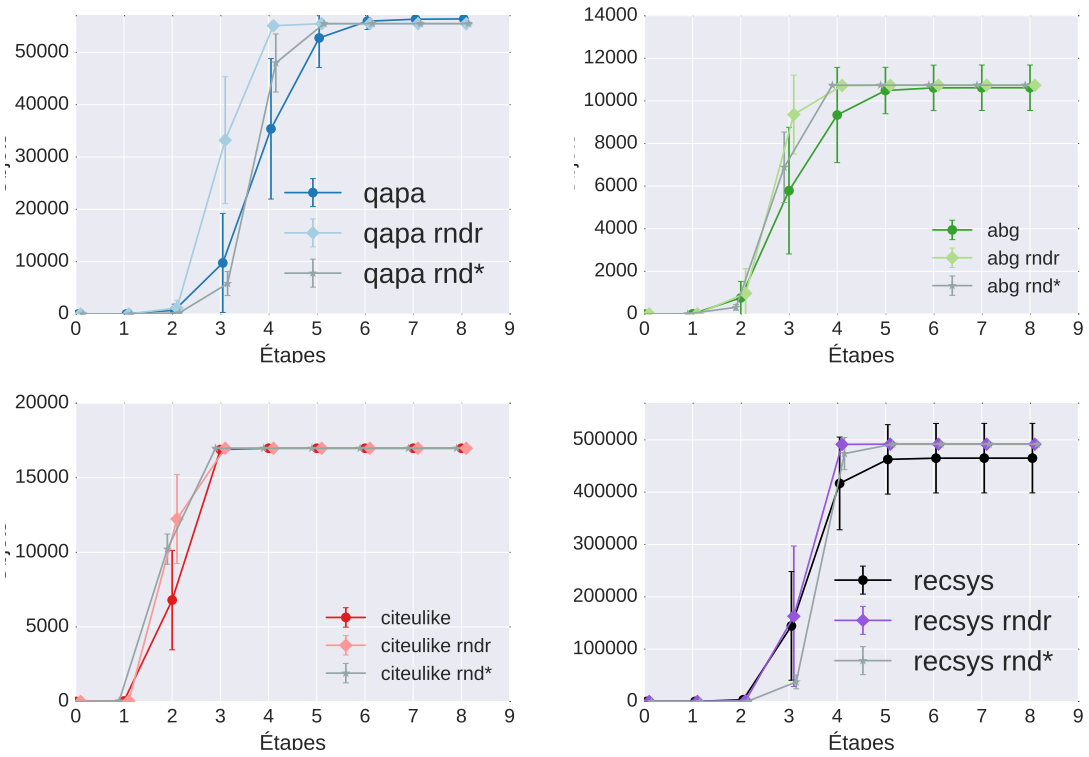


FIGURE 5.2 – Évolution du nombre d'objets atteignables sur le graphe des voisins, comparé avec le jeu mélangé par ligne (*rndr* couleurs plus claires) et tiré uniformément sur la matrice (*rnd ** couleur grise). À l'étape 1, nous avons le nombre d'objets moyen qu'un utilisateur a sélectionnés $|\mathcal{O}_1(u)|$, l'étape 2 correspond aux nombres de voisins de ces objets $|\mathcal{O}_2(u)|$. Tous les graphes sont de type petit monde − comme attendu étant donné la densité et la taille des matrices.

Nous observons une structure de *petit monde* où chaque couple d'objets est voisin avec un degré faible : au maximum 5 pour Qapa et RecSys, 4 pour ABG et 3 pour CiteUlike. Cette structure

de petit monde soulève une difficulté : des voisins à un degré supérieur à deux ou trois ne sont pas nécessairement "proches". L'histogramme des degrés d'arité du graphe est donnée (Figure 5.3). Nous constatons que les noeuds du graphe des objets de Citeulike ont un degré moyen bien plus élevé que celui de Qapa et ABG. Une particularité des données issues d'ABG se lit sur l'histogramme : 988 objets ont plus de 987 voisins, en raison d'un utilisateur ayant 988 notes.

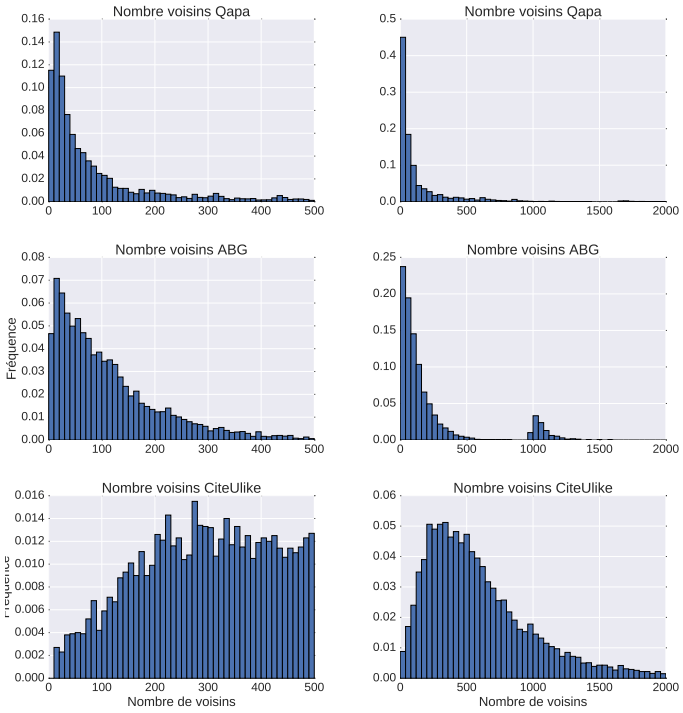


FIGURE 5.3 – Distribution du nombre de voisins (degré d'arité) pour les objets. Les images de gauche (respectivement de droite) sont un zoom sur les 500 (resp. 5 000) premières valeurs. Les degrés moyens et médians sont donnés dans la table ci-dessous. Les degrés sont nettement plus élevés pour CiteUlike dont la densité est plus élevée.

	Degré moyen	Degré médian
Qapa	139	48
ABG	209	97
Citeulike	647	509

5.2 Qualité des documents

Dans cette partie nous analysons la qualité des informations textuelles (complémentaires des informations collaboratives) : les données des textes associés aux CVs et aux offres d'emplois, ainsi que celles des articles CiteUlike.

5.2.1 Comparaisons de tf-idf, LSA, LDA et Doc2Vec

Le choix de la méthode de représentation du texte est toujours guidé par l'application. Nous comparons les différentes méthodes non-supervisées tf, tf-idf, LSA, LDA et Doc2Vec introduites (Section 2). Dans le but de choisir la représentation la plus appropriée pour notre système de recommandation, nous les évaluons sur les trois jeux de données Qapa, ABG, CiteUlike. En effet, rappelons que la valeur d'une représentation n'est pas intrinsèque ; elle est donc évaluée dans le cadre de la recommandation en démarrage à froid, qui est notre objectif.

Méthodologie : démarrage à froid orienté objet Nous décrivons le protocole expérimental suivi pour obtenir les scores du démarrage à froid. Ce protocole diffère de celui utilisé pour le filtrage collaborative (Section 5.1.1). Pour le démarrage à froid orienté objet (Figure 3.1b), nous appliquons une validation croisée sur l'ensemble des objets, générant p groupes disjoints $\mathcal{O}_z$: $\mathcal{O} = \bigsqcup_{z=1}^{p} \mathcal{O}_z$. Nous prenons $p = 10$ pour Qapa et ABG et $p = 5$ pour CiteUlike [2]. $\mathcal{R}^{train,z}$ est la matrice issue de $\mathcal{R}$ où l'ensemble des valeurs associées aux objets de test $\mathcal{O}_z$ sont mises à zéro. Pour chaque utilisateur le rappel indique la fraction des objets tests *pertinents* recommandés parmi l'ensemble des objets test (et non plus l'ensemble de tous les objets comme pour le démarrage à chaud) respectivement de taille 5 651 pour Qapa, pour ABG de taille 1 076, pour CiteUlike de taille 3 396.

(Approche de) Référence Pour chaque représentation, nous appliquons la recommandation basée sur la mémoire, utilisant $\mathcal{R}^{train}$ lors de la convolution (Équation 3.6) :

$$\widehat{\mathcal{R}}_{u,i} = \sum_j sim(i,j) \cdot \mathcal{R}^{train}_{u,j} \tag{5.1}$$

Où la similarité est définie pour chaque représentation :

- tf-idf : $sim_{tf-idf}(i,j) = cos(\boldsymbol{x_i}, \boldsymbol{x_j})$
- LSA : $sim_{LSA}(i,j) = cos(\boldsymbol{x_i}, \boldsymbol{x_j})$
- LDA : $sim_{LDA}(i,j) = \langle \boldsymbol{x_i}, \boldsymbol{x_j} \rangle$
- Doc2Vec : $sim_{Doc2Vec}(i,j) = \langle \boldsymbol{x_i}, \boldsymbol{x_j} \rangle$
- $\mathcal{R}$: $sim_{\mathcal{R}}(i,j) = cos(\boldsymbol{\mathcal{R}^{train}_{\cdot,i}}, \boldsymbol{\mathcal{R}^{train}_{\cdot,j}})$

Notons que ces approches peuvent être utilisées aussi bien dans le cas du démarrage à chaud que du démarrage à froid. Dans le cas des données Qapa et ABG (respéctivement CiteUlike), l'hyper-paramètre définissant la dimension latente k pour LSA, LDA et Doc2Vec est fixé à 300 (resp. 200 pour se comparer équitablement à CTR).

Comparer les représentations Pour chaque objet i, soit $I_i = (j_1, \ldots, j_{100})$ les 100 objets (ordonnés) les plus proches au sens de LSA. $I_i = \kappa\text{-}nn_{LSA}(i)$, $\kappa = 100$. Pour chaque méthode α, nous calculons le vecteur de similarité $\boldsymbol{z_i^\alpha}$ entre l'objet i et ces 100 objets : $\boldsymbol{z_i^\alpha} = (sim(\boldsymbol{x_i}, \boldsymbol{x_{j_1}}), \ldots, sim(\boldsymbol{x_i}, \boldsymbol{x_{j_{100}}}))$. Les différents vecteurs de similarités $\boldsymbol{z_i^\alpha} \in \mathbb{R}^{|\mathcal{O}|}$ et $\boldsymbol{z_i^\beta} \in \mathbb{R}^{|\mathcal{O}|}$ peuvent être comparés à l'aide des indicateurs suivants :

- La *Corrélation de Pearson*

$$p = \frac{Covariance(\boldsymbol{z_i^\alpha}, \boldsymbol{z_i^\beta})}{\sigma_{z_i^\alpha} \sigma_{z_i^\beta}}$$

2. Nous reprenons les même ensemble que CTR Wang and Blei (2011)

	Random	tf	tf-idf	LSA	LDA	Doc2Vec	R	Rappel@100
Random	1.00	0.00	0.00	0.00	0.00	0.01	0.01	0.02
tf	0.00	1.00	0.83	0.51	0.38	0.41	0.11	0.65
tf-idf	0.00	0.83	1.00	0.61	0.49	0.52	0.11	0.68
LSA	0.00	0.51	0.61	1.00	0.33	0.47	0.07	0.69
LDA	0.00	0.38	0.49	0.33	1.00	0.35	0.10	0.61
Doc2Vec	0.01	0.41	0.52	0.47	0.35	1.00	0.09	0.66
R	0.01	0.11	0.11	0.07	0.10	0.09	1.00	0.02

(a) ρ de Spearman pour Qapa, ainsi que le rappel à 100 (dernière colonne) pour chaque représentation.

	Random	tf	tf-idf	LSA	LDA	Doc2Vec	R	Rappel@100
Random	1.00	-0.00	0.00	-0.00	0.00	-0.00	-0.00	0.02
tf	-0.00	1.00	0.66	0.36	0.27	0.28	-0.01	0.65
tf-idf	0.00	0.66	1.00	0.44	0.35	0.36	-0.02	0.68
LSA	-0.00	0.36	0.44	1.00	0.23	0.33	-0.00	0.69
LDA	0.00	0.27	0.35	0.23	1.00	0.24	-0.01	0.61
Doc2Vec	-0.00	0.28	0.36	0.33	0.24	1.00	-0.00	0.66
R	-0.00	-0.01	-0.02	-0.00	-0.01	-0.00	1.00	0.02

(b) τ de Kendall pour Qapa

	Random	tf	tf-idf	LSA	LDA	Doc2Vec	R	Rappel@100
Random	500.00	5.00	5.01	5.01	4.80	4.88	4.87	0.02
tf	5.00	500.00	278.18	13.89	4.64	20.66	3.40	0.65
tf-idf	5.01	278.18	500.00	16.58	5.40	28.37	3.95	0.68
LSA	5.01	13.89	16.58	500.00	2.19	28.92	4.91	0.69
LDA	4.80	4.64	5.40	2.19	500.00	2.45	4.52	0.61
Doc2Vec	4.88	20.66	28.37	28.92	2.45	500.00	4.64	0.66
R	4.87	3.40	3.95	4.91	4.52	4.64	500.00	0.02

(c) Intersection 500 pour Qapa

FIGURE 5.4 – chez Qapa

mesure la dépendance linéaire entre les deux vecteurs $\boldsymbol{z_i^\alpha}$ et $\boldsymbol{z_i^\beta}$, avec $\sigma_{\boldsymbol{z}}$ l'écart type du vecteur $\boldsymbol{z}$;

— Le ρ *de Spearman* est le coefficient de corrélation de Pearson entre *l'ordre* des valeurs de ces deux mesures.

$$\rho = \frac{Covariance(\boldsymbol{rg}_{\boldsymbol{z}_i^\alpha}, \boldsymbol{rg}_{\boldsymbol{z}_i^\beta})}{\sigma_{\boldsymbol{rg}_{\boldsymbol{z}_i^\alpha}} \sigma_{\boldsymbol{rg}_{\boldsymbol{z}_i^\beta}}}$$

$\boldsymbol{rg_z}$ correspond à l'ordre des coordonnées de $\boldsymbol{z}$. Dans ce cas, contrairement à la corrélation de Pearson, les relations non-linéaire sont prises en compte ;

— Le τ *de Kendall* mesure la différence entre les paires concordantes [3] et les paires discordantes.

$$\tau = \frac{|\{pairs\ concordantes\}| - |\{pairs\ discordantes\}|}{|\{pairs\ total\}|}$$

3. La paire (j_1, j_2) est dite concordante pour $\boldsymbol{z}$ et $\boldsymbol{z'}$ si $signe(\boldsymbol{z}^{j_1} - \boldsymbol{z}^{j_2}) = signe(\boldsymbol{z'}^{j_1} - \boldsymbol{z'}^{j_2})$

	Random	tf	tf-idf	LSA	LDA	Doc2Vec	R	Rappel@100
Random	1.00	0.00	0.00	0.00	-0.00	-0.00	-0.01	0.08
tf	0.00	1.00	0.96	0.51	0.63	0.61	0.04	0.34
tf-idf	0.00	0.96	1.00	0.52	0.63	0.64	0.05	0.38
LSA	0.00	0.51	0.52	1.00	0.25	0.42	0.05	0.41
LDA	-0.00	0.63	0.63	0.25	1.00	0.44	0.03	0.33
Doc2Vec	-0.00	0.61	0.64	0.42	0.44	1.00	0.05	0.39
R	-0.01	0.04	0.05	0.05	0.03	0.05	1.00	0.09

(a) ρ de Spearman pour abg

	Random	tf	tf-idf	LSA	LDA	Doc2Vec	R	Rappel@100
Random	1.00	0.00	0.00	0.00	0.00	0.00	-0.00	0.08
tf	0.00	1.00	0.83	0.34	0.44	0.42	0.03	0.34
tf-idf	0.00	0.83	1.00	0.36	0.44	0.45	0.03	0.38
LSA	0.00	0.34	0.36	1.00	0.17	0.29	0.00	0.41
LDA	0.00	0.44	0.44	0.17	1.00	0.30	-0.01	0.33
Doc2Vec	0.00	0.42	0.45	0.29	0.30	1.00	-0.01	0.39
R	-0.00	0.03	0.03	0.00	-0.01	-0.01	1.00	0.09

(b) τ de Kendall pour abg

	Random	tf	tf-idf	LSA	LDA	Doc2Vec	R	Rappel@100
Random	500.00	25.62	25.83	25.86	25.98	26.00	26.23	0.08
tf	25.62	500.00	325.88	23.17	34.61	78.69	37.51	0.34
tf-idf	25.83	325.88	500.00	23.13	34.81	79.35	39.14	0.38
LSA	25.86	23.17	23.13	500.00	11.41	37.73	24.81	0.41
LDA	25.98	34.61	34.81	11.41	500.00	44.51	20.96	0.33
Doc2Vec	26.00	78.69	79.35	37.73	44.51	500.00	23.16	0.39
R	26.23	37.51	39.14	24.81	20.96	23.16	500.00	0.09

(c) Intersection 500 abg

FIGURE 5.5 – chez ABG

— $\cap_{500}$ Le *nombre d'intersection des 500 premiers* représente le nombre d'objets communs parmi les 500 plus proches voisins pour chacune des deux mesures.

$$\cap_{500} = |\kappa\text{-}nn_\alpha \cap \kappa\text{-}nn_\beta| \qquad \kappa = 500$$

L'intérêt de cette approche facilement interprétable est de prendre en compte *l'ensemble* des objets disponibles. Cependant, contrairement aux autres indicateurs ci-dessus, l'ordre relatif n'est pas pris en compte.

Rappelons que les trois premiers indicateurs prennent leurs valeurs entre -1 et 1, deux vecteurs indépendants obtenant une valeur de zéro. $\cap_{500}$ prend des valeurs entre 0 et 500, deux ordres indépendants obtiennent un score d'intersection de l'ordre de $\frac{500}{|\mathcal{O}|}$. Nous donnons également pour chaque représentation son score associé à la méthode de recommandation à base de mémoire en démarrage à froid objet.

Analyse des résultats

	Random	tf	tf-idf	LSA	LDA	Doc2Vec	R	Rappel@100
Random	1.00	-0.00	-0.01	-0.00	0.00	0.00	-0.00	0.03
tf	-0.00	1.00	0.61	0.51	0.41	0.41	0.10	0.52
tf-idf	-0.01	0.61	1.00	0.51	0.31	0.33	0.11	0.59
LSA	-0.00	0.51	0.51	1.00	0.24	0.38	0.07	0.62
LDA	0.00	0.41	0.31	0.24	1.00	0.32	0.12	0.58
Doc2Vec	0.00	0.41	0.33	0.38	0.32	1.00	0.11	0.53
R	-0.00	0.10	0.11	0.07	0.12	0.11	1.00	0.03

(a) ρ de Spearman pour citeulike

	Random	tf	tf-idf	LSA	LDA	Doc2Vec	R	Rappel@100
Random	1.00	0.00	0.00	0.00	0.00	0.00	-0.00	0.03
tf	0.00	1.00	0.44	0.36	0.29	0.29	0.03	0.52
tf-idf	0.00	0.44	1.00	0.36	0.21	0.23	0.04	0.59
LSA	0.00	0.36	0.36	1.00	0.17	0.27	0.02	0.62
LDA	0.00	0.29	0.21	0.17	1.00	0.23	0.03	0.58
Doc2Vec	0.00	0.29	0.23	0.27	0.23	1.00	0.03	0.53
R	-0.00	0.03	0.04	0.02	0.03	0.03	1.00	0.03

(b) τ de Kendall pour citeulike

	Random	tf	tf-idf	LSA	LDA	Doc2Vec	R	Rappel@100
Random	500.00	18.36	18.91	18.60	18.17	18.83	18.52	0.03
tf	18.36	500.00	71.88	42.68	43.89	50.73	20.24	0.52
tf-idf	18.91	71.88	500.00	64.41	42.81	45.18	29.07	0.59
LSA	18.60	42.68	64.41	500.00	23.53	54.51	17.60	0.62
LDA	18.17	43.89	42.81	23.53	500.00	38.73	16.49	0.58
Doc2Vec	18.83	50.73	45.18	54.51	38.73	500.00	19.65	0.53
R	18.52	20.24	29.07	17.60	16.49	19.65	500.00	0.03

(c) Intersection 500 citeulike

FIGURE 5.6 – Chez CiteUlike.

Random correspond à une représentation aléatoire des objets, le score $\cap_{500}$ correspond donc au tirage aléatoire.

tf et tf-idf sont fortement corrélées pour toutes les méthodes et sur tous les ensembles. tf-idf obtient des score nettement meilleurs que tf (+6% sur Qapa, +17% sur ABG et +13% pour CiteUlike). tf-idfchange légèrement l'ordre de tf ainsi que les valeurs de similarité.

LSA améliore systématiquement le score de tf-idf, tout en apportant une représentation dense de plus faible dimension. LSA obtient un taux de corrélation significatif par rapport à tf-idf.

LDA est moins corrélé aux autres méthodes. De plus, la représentation LDA obtient des scores inférieurs à LSA ainsi qu'à tf-idf.

Doc2Vec a une corrélation forte avec tf, tf-idf et LSA. Doc2Vec obtient de meilleurs résultats sur les données Qapa relativement aux autres jeux de données, ce qui est attribué à la plus grande taille des données. Doc2Vec obtient un score supérieur à tf et tf-idf sur ABG.

$\mathcal{R}$ est très peu corrélé avec les autres représentations de texte [4]. Rappelons que le rappel à 100

4. Notons que tous les objets n'ont pas 100 voisins pour $\mathcal{R}$ (particulièrement dans Qapa et ABG) (Figure 5.3).

de $\mathcal{R}$ est dû à l'absence d'information collaborative sur les données de démarrage à froid.

De manière surprenante, les intersections à 500 sont très faibles entre toutes les représentations, hormis entre tf et tf-idf. En effet, sur les données Qapa, LDA range aléatoirement les 500 premières recommandations vis à vis des autres représentations et LSA et tf-idf ne partagent que 16 mêmes offres en moyenne sur leurs 500 premières. Doc2Vec en partage plus avec LSA et tf-idf. *Cela montre que l'ordre est moins important que les valeurs lors de la recommandation à base de mémoire, utilisant la convolution.* La décroissance des valeurs de similarités a un impact étudié (Section 4.2.2). LSA obtenant de manière générale et constante le meilleur résultat, nous nous concentrons dans la suite du manuscrit sur la représentation LSA comme représentation de référence.

5.2.2 Appariement direct offre - CV

La recommandation confronte directement les mots clefs d'un CV et les mots clefs d'une offre. Dans ce cas les données collaboratives ne sont pas nécessaires.

Compétences Qapa Dans le cas de Qapa, les mots clefs sont disponibles et correspondent aux compétences, comptant 168 000 compétences différentes parmi les offres et 2 700 000 parmi les CVs. Nous pouvons donc recommander à un utilisateur les objets avec le plus de compétences communes. Ce système est à la base du moteur de recommandation de Qapa. Nous obtenons la matrice de score $\widehat{\mathcal{R}}_{u,i}$ par l'équation :

$$\widehat{\mathcal{R}}_{u,i} = |\{c | c \in Comp(u) \cap Comp(i)\}| \tag{5.2}$$

où $Comp(u)$ (respectivement $Comp(i)$) désigne l'ensemble des compétences associées à l'utilisateur u (resp. l'objet i). D'après (Figure 5.7), moins de 400 compétences figure dans plus de 100 offres et moins de 1 000 compétences se retrouvent dans plus de 20 offres. Les autres compétences sont entrées à la main, souvent avec des fautes d'orthographe. Pour rappel, un nuage de mots de compétences est donné (Figure 4.3d).

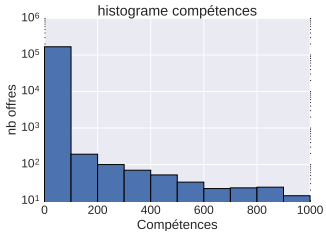


FIGURE 5.7 – Histogramme des compétences partagées par les offres d'emplois chez Qapa en echelle log. Beaucoup de compétences (non représentées ici) sont orphelines. Seulement 100 compétences sont utilisées par plus de 1 000 recruteurs.

Les scores sont reportés (Table 5.3). Notons que ce score est bien en dessous d'un score de filtrage collaboratif ou celui basé sur le contenu (voir plus bas). L'avantage principal de cette méthode est qu'elle est directe, ne nécessite pas de calcul, n'a pas besoin d'apprentissage et apparie offres et CVs à froid. Cependant, cette méthode n'exploite aucunement l'information collaborative. De plus,

Ainsi un ordre aléatoire des objets de similarité nulle tire vers le bas les corrélations de Spearman et Kendall.

Qapa	Compétences communes	LSA
Rappel@100	.17	.34
Rappel@500	.39	.62
Rappel@1000	.53	.69

TABLE 5.3 – Score de rappel (démarrage à chaud) en utilisant l'appariement direct entre CV et offre chez Qapa, à l'aide des compétences communes (Équation 5.2). Pour comparaison, la colonne de droite rappel le score à base de mémoire LSA. Ces résultats sont largement inférieurs aux résultats à base de mémoire.

contrairement à la plateforme Qapa, cette méthode n'exploite ni la localisation ni le salaire ni la date de publication.

5.2.3 Recommandation à base de mémoire

Cette section décrit une approche où seulement un des deux types de document (seulement offres ou seulement CVs) est pris en compte pour générer les recommandations. L'objectif de cette partie est d'appliquer la méthode à base de mémoire pour évaluer la qualité des documents textes. En effet, dans cette approche, la similarité entre documents est à la base de la recommandation : "recommander plus du *même*". Notons que si la similarité entre documents est tout à fait corrélée à la similarité des interactions, nous obtiendrons des résultats proches de ceux obtenus en filtrage collaboratif à base de mémoire.

Rappel@100	Sac de termes	tf-idf	LSA
Mots	.32	.44	.48
Compétences	.64	.65	.67
Mots + Compétences	.63	.68	**.69**

TABLE 5.4 – Apport des compétences dans la recommandation chez Qapa. Les compétences comportent la plupart des informations pour élaborer la similarité. Ajouter les mots apporte néanmoins une amélioration significative dans la représentation tf-idf et LSA. Le score de rappel à 100 est calculé pour le démarrage à froid (Section 5.2.1)

La Table 5.4 représente la contribution des mots et des compétences dans le problème de recommandation à base de mémoire, orienté objet. De ces résultats nous déduisons que la principale source d'information pour notre système se trouve parmi les compétences. En effet, ces termes standardisés permettent conduisent à une similarité plus fiable, tandis que les mots peuvent être bruités et vides.

Pour avoir des scores comparables à ceux du filtrage collaboratif basé sur la mémoire, nous nous plaçons − dans ce cas seulement − dans les mêmes conditions que pour l'évaluation du filtrage collaboratif en démarrage à chaud (Section 5.1.1). Les évaluation sont reportées (Figure 5.8) et (Tables 5.5 et 5.6).

De ces résultats, nous remarquons que :

— La recommandation directe entre compétences échoue à donner des recommandations de qualité. En effet, nous ne disposons pas, contrairement à la société Qapa, des données correspondant aux salaires, à la durée de travail, au temps plein ou au temps partiel. Cette

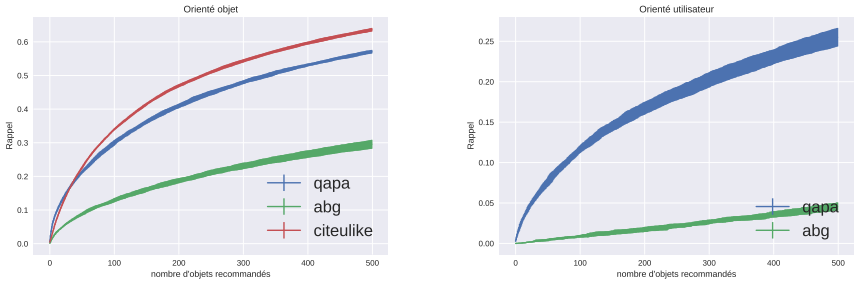


FIGURE 5.8 – Courbe de rappel pour les jeux de données Qapa, ABG et CiteUlike dans le cas orienté objet (à gauche) et orienté utilisateur (à droite). Notons que CiteUlike ne fournit pas de description pour ses utilisateurs. Le cas orienté utilisateur obtient des scores très inférieurs au cas orienté objet. Cela est imputé à la qualité plus faible des CVs. Ces figures sont directement comparables aux scores par filtrage collaboratif (Figures 5.1 et 5.2)

recommandation est donc une première approche grossière ;

— Pour tous les jeux de données, les résultats exploitant le texte sont largement dominés par ceux du filtrage collaboratif. La comparaison étant équitable et la matrice de convolution identique, les seules différences sont les valeurs de similarités $sim(i, j)$ et $sim(u, v)$. Nous analysons la corrélation entre ces similarités dans la section suivante ;

— La méthode orientée objet est nettement meilleure que celle orientée utilisateur, contrairement au cas du filtrage collaboratif. Cela s'explique par une meilleure qualité des offres d'emplois relativement aux CVs, les offres étant plus ciblées et mieux écrites.

Dans la suite, nous nous concentrons sur les offres d'emploi, abandonnant la description des CVs. Nous disposons donc de deux représentations pour une offre : l'ensemble des notes des utilisateurs ainsi que la description texte associée.

	qapa	abg	citeulike
Rappel@20	0.129 ± 0.005	0.049 ± 0.002	0.116 ± 0.003
Rappel@100	0.298 ± 0.008	0.129 ± 0.006	0.338 ± 0.003
Rappel@200	0.409 ± 0.007	0.187 ± 0.008	0.469 ± 0.005
Rappel@200 (cf)	0.546 ± 0.01	0.429 ± 0.01	0.589 ± 0.005

TABLE 5.5 – Score orienté objet (Figure 5.8). La dernière ligne rappelle pour mémoire les scores issus du filtrage collaboratif (Table 5.1)

	qapa	abg
Rappel@20	0.042 ± 0.004	0.002 ± 0.001
Rappel@100	0.116 ± 0.005	0.009 ± 0.002
Rappel@200	0.167 ± 0.008	0.017 ± 0.003
Rappel@200 (cf)	0.587 ± 0.009	0.502 ± 0.009

TABLE 5.6 – Score orienté utilisateur (Figure 5.8). La dernière ligne rappelle pour mémoire les scores du filtrage collaboratif (Table 5.2)

5.3 Analyse Qapa par métiers

Dans cette section nous étudions les deux représentations issues LSA et de $\mathcal{R}$ à l'aide des étiquettes métier des offres et des CVs. Rappelons qu'à chaque offre est associé un métier, et qu'à chaque CV est associée une liste de métiers. La similarité LSA moyenne entre toutes les offres appartenant à un même métiers est donnée (Table 5.7). La similarité dans l'espace $\mathcal{R}$ n'est pas donnée, car les similarités entre offres sont toutes proches de zéro ; les points sont presque tous orthogonaux (Figure 5.12). Nous donnons aussi le score [5], pour LSA et $\mathcal{R}$, du rappel à 30 et à 100 des utilisateurs associés à chaque groupe de métier [6]. La table est ordonnée par le rappel à 100 LSA.

5.3.1 Cartographie des offres et des métiers

Une autre façon d'exhiber la différence de représentations entre $\mathcal{R}$ et LSA, ainsi que d'affiner notre compréhension de ces représentations est de projeter certaines offres et CVs dans le plan $\mathbb{R}^2$.

Méthodologie : Nous sélectionnons des métiers parmi ceux les plus représentés dans Qapa. La légende des couleurs correspondant aux métiers est donnée (Figure 5.9). Pour chaque métier, nous sélectionnons aléatoirement 200 offres et 200 CVs [7]. Nous affichons les offres (Figure 5.10a), les CVs (Figure 5.10b) puis les offres et CVs ensemble (Figure 5.11). La projection s'obtient en appliquant t-SNE [8] (Section 2.4) sur la représentation LSA [9].

5. Les scores correspondent au démarrage à chaud où l'ensemble des objets test est restreint à 10% de l'ensemble des objets. Cela rend les scores LSA et $\mathcal{R}$ comparables .

6. Pour calculer le score associé à un groupe, nous moyennons les scores de chaque utilisateur sur le groupe et non plus sur l'ensemble des utilisateurs.

7. un CV est associé à un métier si ce métier apparaît dans la liste des métiers du CVs

8. La projection t-SNE est optimisé sur l'ensemble des points représenté.

9. Pour s'assurer d'être dans le même espace, la réduction tf-idf et LSA est obtenu sur l'ensemble des vocabulaires CVs et offres. LSA est calculé sur la matrice tf-idf des offres et des CVs. Ainsi la projection a lieu dans un seul espace de vocabulaire, correspondant aux vocabulaire commun.

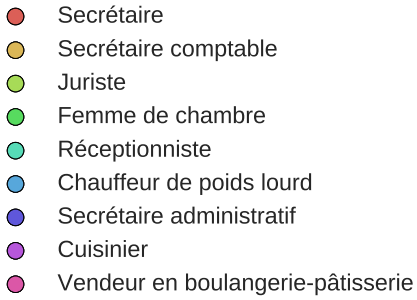
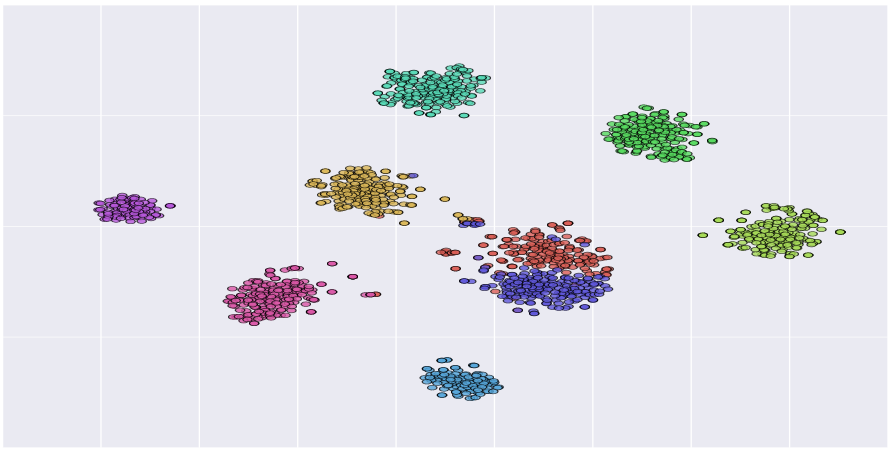


FIGURE 5.9 – Légende de couleur (Figures 5.10, 5.11 et 5.12)

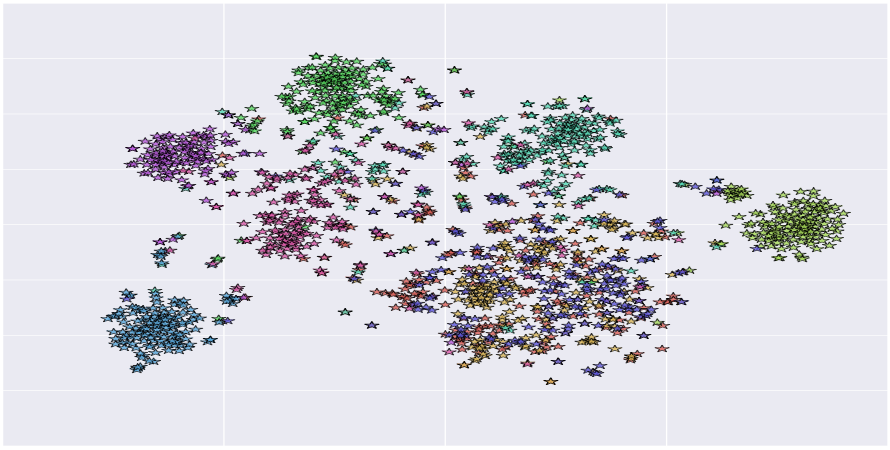
Interprétations La représentation LSA des offres distingue nettement les différents métiers, phénomène amplifié par t-SNE qui force les regroupements. (À l'exception de secrétaire (rouge) et de secrétaire administrative (bleu/magenta) qui sont projeté dans un même groupe, mais encore séparé à l'intérieur de ce groupe.) Pour les CVs, des blocs sont nettement séparés pour chauffeur de poids lourd (bleu), juriste (vert pomme), cuisinier (violet) et femme de chambre (vert foncé). Cependant, il y a indistinction dans la représentation des différents métiers de secrétaires (rouge, bleu/magenta et orange).

Dans le cas où offres et CVs sont représentées ensemble (Figure 5.11), les offres sont regroupées et séparées des CVs. (Excepté encore les amalgames des CVs de secrétaires ainsi qu'un amas plus éclaté de vendeur en boulangerie (rose), réceptionniste (cyan), femme de chambre (vert foncé) et cuisinier (violet) au centre.) Les CVs et les offres de juriste (vert pomme) et de chauffeur de poids lourds (bleu ciel) restent bien regroupés à part, car leur texte est bien différencié des autres textes.

Il apparaît de nombreux petits groupes dans la projection des offres partant des notes (Figure 5.12). Juriste (vert pomme), femme de chambre (vert foncé) et réceptionniste (bleu claire) forment plusieurs groupes, presque toujours homogènes, de même, dans une moindre mesure, avec les points représentant cuisiniers (violet) et vendeurs en boulangerie (rose). À l'intérieur de ces groupes les points des secrétaires (rouge, bleu/magenta et orange) sont mélangés. D'où certainement les meilleurs résultats pour secrétaires dans le cas du filtrage collaboratif que à base de LSA. Les petits regroupements peuvent être dût à cause de salaire, de géo-localité ou de temporalité différentes.



(a) Projection des offres de Qapa



(b) Projection des CVs de Qapa

FIGURE 5.10 – Projection des offres (rond en haut) et des CVs (étoiles en bas) à partir de leur représentation LSA. La réduction de dimension est obtenue à l'aide de t-SNE. Les offres sont bien séparées dans l'espace LSA, les CVs sont également plutôt regroupés, excepté pour les différents métiers de secrétaires qui sont mélangés.

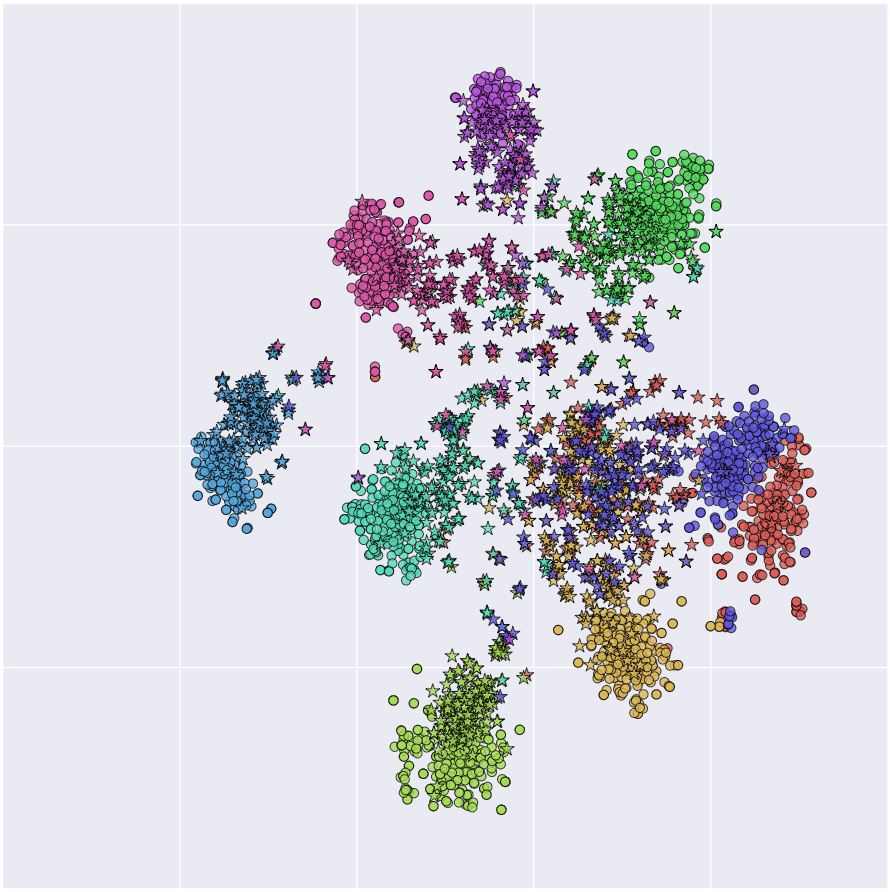


FIGURE 5.11 – Projection des CVs (étoiles) et des offres (ronds) à partir de leur représentation LSA commune.

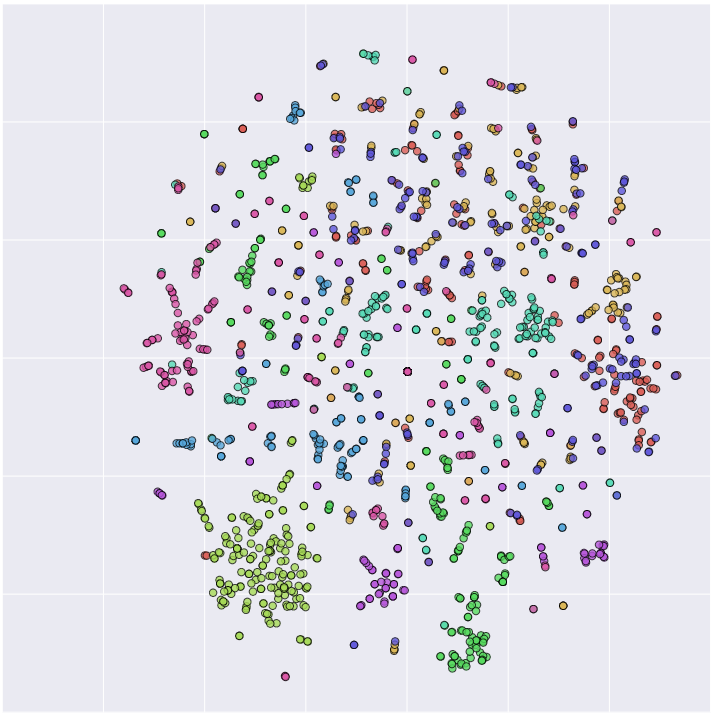


FIGURE 5.12 – Projection induite par $\mathcal{R}$. Beaucoup de paires de points ont une similarité nulle, ce qui explique la fragmentation en petits groupes, selon par exemple la région ou la date de postage.

Metiers	Sim(Offres)	R@30 (LSA)	R@100(LSA)	R@30 ($\mathcal{R}$)	R@100 ($\mathcal{R}$)
Juriste	0.58	0.49	0.91	0.51	0.69
Conducteur / Conductrice de travaux	0.52	0.49	0.87	0.65	0.68
Agent / Agente de sécurité	0.65	0.53	0.86	0.78	0.84
Assistant / Assistante de gestion du personnel	0.45	0.55	0.86	1.0	1.0
Contrôleur / Contrôleuse de gestion	0.58	0.58	0.85	0.61	0.72
Chauffeur-livreur / Chauffeuse-livreuse	0.48	0.43	0.83	0.65	0.74
Magasinier / Magasinière	0.55	0.44	0.76	0.61	0.73
Plongeur / Plongeuse en restauration	0.7	0.41	0.76	0.58	0.64
Valet / Femme de chambre	0.79	0.52	0.74	0.65	0.68
Aide ménager / ménagère à domicile	0.62	0.52	0.74	0.37	0.51
Agent / Agente d'entretien/propreté de locaux	0.73	0.29	0.7	0.5	0.6
Réceptionniste en hôtellerie	0.6	0.39	0.65	0.68	0.78
Chargé / Chargée de recrutement	0.53	0.41	0.65	0.43	0.55
Caissier / Caissière	0.48	0.31	0.64	0.51	0.58
Commercial / Commerciale sédentaire	0.4	0.36	0.62	0.43	0.57
Préparateur / Préparatrice de commandes	0.52	0.38	0.6	0.57	0.67
Hôte / Hôtesse d'accueil	0.4	0.38	0.59	0.57	0.67
Vendeur / Vendeuse en boulangerie-pâtisserie	0.52	0.28	0.58	0.6	0.73
Secrétaire médical / médicale	0.65	0.38	0.57	0.5	0.65
Vendeur / Vendeuse en prêt-à-porter	0.5	0.24	0.55	0.69	0.79
Agent administratif / Agente administrative	0.33	0.36	0.54	0.51	0.67
Assistant commercial / Assistante commerciale	0.36	0.35	0.52	0.52	0.63
Serveur / Serveuse de restaurant	0.66	0.3	0.51	0.48	0.64
Secrétaire	0.3	0.29	0.49	0.59	0.74
Secrétaire comptable	0.47	0.27	0.48	0.63	0.75
Assistant administratif / Assistante administrative	0.31	0.26	0.48	0.59	0.73
Secrétaire administratif / administrative	0.36	0.22	0.43	0.45	0.67
Tous métiers	0.02	0.3	0.63	0.56	0.67
Tous	0.01	0.34	0.63	0 .66	0.72

TABLE 5.7 – Score de rappel pour LSA et $\mathcal{R}$ sur 10% des données (Note 5). Les métiers de secrétaires, secrétaires comptable et secrétaires administratives sont généralement plus difficiles à recommander que les métiers plus ciblés comme juriste, conducteur de travaux et agent de sécurité.

Chapitre 6

Réseaux Siamois LaJam

Nous décrivons ici une méthode originale basée sur les réseaux siamois pour apprendre une nouvelle représentation de documents textuels. Cette méthode est l'évolution de la méthode CLE présentée en annexe. L'idée directrice de ce chapitre n'est plus de conserver la structure locale aux voisinages des objets, mais de rapprocher la représentation des objets ayant une similarité oracle forte.

Le plan de ce chapitre est le suivant. Dans un premier temps nous formalisons l'objectif d'apprentissage de métrique (metric learning) (Section 6.1). Le problème ainsi défini est d'abord attaqué dans un cadre linéaire, par optimisation convexe ; dans un second temps, nous considérons un cadre non linéaire, celui des réseaux siamois (Bromley et al., 1994). Nous décrivons ensuite le modèle de réseau siamois LaJam que nous avons adapté au problème de l'emploi (Section 6.2). Dans la section suivante (Section 6.3) nous décrivons une expérience préliminaire sur des données artificielles. L'implémentation de LaJam ainsi que les résultats obtenus sur les données réelles dans les cas de recommandation (en démarrage à chaud, et à froid) sont ensuite présentés. Enfin nous discutons les résultats ainsi que les perspectives (Section 6.5).

6.1 Apprentissage de métrique

Nous considérons une paire d'objets (i,j) ainsi que leur représentation initiale $(\boldsymbol{x_i}, \boldsymbol{x_j}) \in \mathbb{R}^D \times \mathbb{R}^D$. Ces représentations peuvent être issues des mots dans le cas de documents texte ou de l'ensemble des pixels dans le cas des images. Ces représentations sont a priori de qualité faible pour évaluer les similarités entre objets, dans le sens où la métrique euclidienne ou la similarité cosinus ne définit pas une bonne mesure pour attaquer des objectifs de catégorisation (clustering), de classification ou de recommandation.

L'hypothèse de travail faite est que l'on dispose d'une similarité *oracle* permettant de dire, pour l'ensemble des paires d'objets d'entraînement, s'ils sont similaires ou non. Dans le cadre de la classification, chaque instance est étiquetée par sa classe, et deux objets sont similaires s'ils appartiennent à la même classe. Dans d'autres contextes (en particulier le cas des données *SemEval*[1]), des experts ont annoté la similarité de paires de textes.

1. `http://alt.qcri.org/semeval2017/`. Ce corpus fournit 5 749 paires de phrases, avec leur similarité annotée par des experts humains sur une échelle de 0 à 5.

L'apprentissage de métrique se fonde sur la similarité oracle pour modifier la métrique "naturelle" sur l'espace des instances, dans le but de pouvoir réaliser ensuite les tâches voulues (catégorisation, classification ou recommandation) en utilisant de simples algorithmes de type plus-proche-voisin sur la base de la métrique ainsi modifiée.

L'apprentissage de métrique a de nombreuses applications : en particulier en vision pour la classification d'images (Mensink et al., 2012) ; en reconnaissance d'objets (Verma et al., 2012) et de visages (Lu et al., 2014) ; ou encore en recherche d'information (Lim et al., 2013) ; ainsi que des applications dans le domaine de la recherche d'information (Lim et al., 2013). L'approche la plus classique consiste à modifier la métrique euclidienne définie sur l'espace initial (Bellet et al., 2013).

6.1.1 Approche linéaire : Distance de Mahalanobis

La distance de Mahalanobis (Mahalanobis, 1936) est caractérisée par une matrice $M \in \mathbb{R}^{D \times D}$ semi-définie positive, i.e. une matrice diagonalisable dont toutes les valeurs propres sont positives ou nulles :

$$d_M(\boldsymbol{x_i}, \boldsymbol{x_j}) = \sqrt{(\boldsymbol{x_i} - \boldsymbol{x_j})^T \cdot M \cdot (\boldsymbol{x_i} - \boldsymbol{x_j})} \tag{6.1}$$

Cette propriété assure que d_M est une pseudo-distance (vérifiant les axiomes de symétrie et d'inégalité triangulaire, mais pas l'axiome de séparation) dans l'espace $\mathbb{R}^D$. La matrice M effectue un changement de représentation linéaire, et pondère l'importance des nouvelles dimensions ainsi définies. La pseudo-distance d_M coïncide naturellement avec la distance euclidienne si M est l'identité.

En se restreignant au cas où

$$M = L^T L$$

avec L une matrice $k \times D$, envoyant l'espace des instances $\mathbb{R}^D$ sur $\mathbb{R}^k$ (ou la dimension k est un hyper-paramètre du modèle), d_M peut s'écrire sous la forme :

$$d_M(\boldsymbol{x_i}, \boldsymbol{x_j}) = \sqrt{(\boldsymbol{x_i} - \boldsymbol{x_j})^T \cdot L^T L \cdot (\boldsymbol{x_i} - \boldsymbol{x_j})} \tag{6.2}$$

$$= \sqrt{(\boldsymbol{L x_i} - \boldsymbol{L x_j})^T \cdot (\boldsymbol{L x_i} - \boldsymbol{L x_j})} \tag{6.3}$$

$$= \|\boldsymbol{L x_i} - \boldsymbol{L x_j}\|_2 \tag{6.4}$$

La distance de Mahalanobis d_M correspond ici à la distance euclidienne dans l'espace image de la fonction linéaire L.

Large Margin Nearest Neighbor (LMNN) Dans leurs travaux, Weinberger *et al.* (Weinberger et al., 2006) appliquent l'apprentissage de métrique pour réaliser un apprentissage supervisé de classification (large margin nearest neighbor classification (LMNN)). Formellement, l'objectif consiste à rapprocher un exemple de ses κ plus-proches voisins de même classe, tout en repoussant ceux des κ plus-proches voisins de classes différentes au-delà d'une marge donnée (où le nombre de voisins κ et la marge sont des hyper-paramètres).

LMNN est l'une des méthodes d'apprentissages de métrique les plus utilisées en raison de sa simplicité et de ses bonnes performances (Bellet et al., 2013). L'optimisation de L fait intervenir deux termes : le terme d'attraction, qui cherche à rapprocher un point de ses voisins de même classe, et le terme de répulsion, qui cherche à éloigner un point de ses voisins de classe différente. Formellement, pour chaque point $\boldsymbol{x}$ soit $V_{\kappa+}(\boldsymbol{x})$ le sous-ensemble de ses κ plus proches voisins de

même classe que $\boldsymbol{x}$, appelé ensemble des voisins positifs. Le terme d'attraction $\mathcal{L}_+$ (à minimiser) est donné par

$$\mathcal{L}_+ = \sum_{x \in X} \sum_{x_+ \in V_{\kappa+}} d_M^2(\boldsymbol{x}, \boldsymbol{x}_+)$$

De manière analogue l'ensemble des voisins négatifs est donné par :

$$V_{\kappa-}(\boldsymbol{x}) = \{\boldsymbol{x}_- \in \kappa\text{-}nn(\boldsymbol{x}),\ classe(\boldsymbol{x}) \neq classe(\boldsymbol{x}_-)\}$$

Et le terme de répulsion $\mathcal{L}_-$, à minimiser :

$$\mathcal{L}_- = \sum_{x \in X} \sum_{x_+ \in V_{\kappa+}} \sum_{x_- \in V_{\kappa-}} [1 + d_M^2(\boldsymbol{x}, \boldsymbol{x}_+) - d_M^2(\boldsymbol{x}, \boldsymbol{x}_-)]_+$$

Avec $[x]_+ = \max(0,\, x)$. La minimisation du terme de répulsion cherche à imposer que les distances des couples négatifs $d_M^2(\boldsymbol{x}, \boldsymbol{x}_-)$ soient supérieures aux distances des couples positifs plus une marge fixée à 1 par convention. Notons que cette convention permet de fixer la norme de la fonction linéaire L. Finalement, l'apprentissage de métrique procède en minimisant le problème convexe

$$L^* = \arg\min_L \{\mathcal{L} = (1-\mu) \cdot \mathcal{L}_+ + \mu \cdot \mathcal{L}_-\}$$

Le paramètre $\mu \in [0, 1]$ contrôle l'équilibre entre attraction et répulsion. Il est optimisé par validation croisée.

Du coté positif, LMNN définit un problème d'optimisation convexe, garantissant ainsi l'existence et l'unicité de la solution optimale. Il obtient de bonnes performances en apprentissage supervisé. Du coté négatif, LMNN est susceptible de sur-apprentissage, en particulier dans les problèmes de grande dimension. De surcroît, cette méthode est sensible à la distance initiale, qui définit les voisins positifs et négatifs.

Une extension naturelle proposée par (Weinberger and Saul, 2009) consiste à définir plusieurs distances de Mahalanobis sur différentes régions de l'espace. Pour ce faire, les données sont partitionnées en C clusters (e.g., par exemple par un algorithme de C-moyennes), puis une distance de Mahalanobis est apprise sur chaque cluster. En pratique cette méthode obtient de meilleurs résultats que LMNN ; le coût de calcul augmente linéairement avec C. En revanche, le risque de surapprentissage augmente également avec C.

6.1.2 Approche non-linéaire : les Réseaux Siamois

Les réseaux siamois (Bromley et al., 1994) suivent la même démarche que LMNN, en remplaçant le changement de représentation linéaire L par un changement de représentation non linéaire, réalisé par un réseau neuronal. Cette approche peut également être vue comme une méthode d'estimation par noyau (Parzen, 1962). La distance d'une paire d'objets est donnée par la norme euclidienne de la différence de leurs images par ϕ :

$$d_\phi(\boldsymbol{x_i}, \boldsymbol{x_j}) = \|\phi(\boldsymbol{x_i}) - \phi(\boldsymbol{x_j})\|_2 \tag{6.5}$$

L'appellation de réseau siamois est due au fait qu'un même réseau ϕ est utilisé en parallèle ; cela peut être vu comme deux réseaux partageant les même paramètres (Figure 6.1).

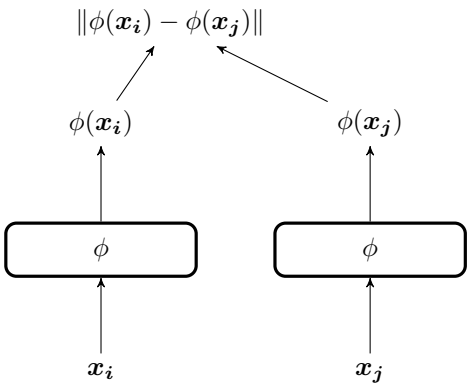


FIGURE 6.1 – Réseau siamois. La représentation ϕ est utilisée pour définir la distance du couple $(\boldsymbol{x_i}, \boldsymbol{x_j})$ comme la distance euclidienne de leurs images dans $\mathbb{R}^d$. La représentation ϕ est optimisée par gradient stochastique pour satisfaire les propriétés voulues de la métrique cherchée.

Historiquement, les réseaux siamois sont introduits par (Bromley et al., 1994), motivés par la vérification de signature. Le rapprochement des exemples positifs est modélisé par un terme d'attraction, calculé sur *l'ensemble* des couples similaires

$$V_+(\boldsymbol{x}) = \{\boldsymbol{x_+} \in X,\ classe(\boldsymbol{x}) = classe(\boldsymbol{x_+})\}$$

$$\mathcal{L}_+ = \sum_{x \in X} \sum_{x_+ \in V_+} d_\phi(\boldsymbol{x}, \boldsymbol{x_+})^2 \tag{6.6}$$

De même que dans le cas de LMNN, l'optimisation du seul terme d'attraction conduit à une solution triviale (par exemple une représentation ϕ constante). Un terme de répulsion est donc également considéré, forçant l'éloignement de *l'ensemble* des couples dissimilaires

$$V_-(\boldsymbol{x}) = \{\boldsymbol{x_-} \in X,\ classe(\boldsymbol{x}) \neq classe(\boldsymbol{x_-})\}$$

$$\mathcal{L}_- = \sum_{x \in X} \sum_{x_- \in V_-} [m - d_\phi^2(\boldsymbol{x}, \boldsymbol{x_-})]_+^2 \tag{6.7}$$

où le paramètre m de marge est un hyper-paramètre de la méthode. La somme des deux termes d'attraction et de répulsion définit la fonction de perte globale, appelée perte contrastive :

$$\mathcal{L} = \mathcal{L}_+ + \mathcal{L}_- \tag{6.8}$$

Contrairement à LMNN, la perte contrastive ne considère pas les voisinages locaux pour les voisins positifs. En effet, tous les voisins positifs sont considéré pour prendre en compte le faible pourcentage de paires positives. Dans le cas des voisins négatifs, seulement les points dont la distance (au sens de la représentation ϕ) est plus petite que m sont pris en compte. Le nombre de voisins négatifs possibles dépend donc de la densité des points, contrairement à LMNN qui fixe a priori le nombre de voisins considérés à κ, indépendamment de la densité du voisinage de chaque point.

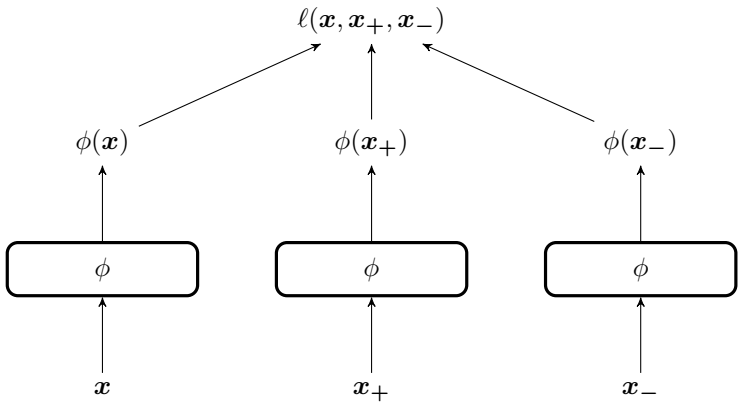


FIGURE 6.2 – Réseaux triplés.

Dans le cas des images (Chopra et al., 2005; Hadsell et al., 2006), le réseau ϕ est un réseau convolutionel (LeCun et al., 1998a), en raison de ses bonnes propriétés dans le domaine des images. L'approche a également été utilisée dans le contexte de la recherche d'information, notamment sur les données *SemEval*. Dans le contexte des documents textuels, des réseaux à base de LSTM (Hochreiter and Schmidhuber, 1997) sont souvent considérés (Mueller and Thyagarajan, 2016). On peut également considérer un LSTM bi-directionnel (couplant deux réseaux LSTM lisant la séquence de droite à gauche et de gauche à droite) (Schuster and Paliwal, 1997) au niveau des caractères (Neculoiu et al., 2016). La fonction de perte est donnée par la "perte contrastive" avec la norme cosinus dans l'espace image.

Réseaux triplés Les réseaux siamois ont été étendus en considérant des réseaux triplés (Triplet network) (Hoffer and Ailon, 2015) (Figure 6.2), permettant de comparer les qualités *relatives* des paires $(\boldsymbol{x}, \boldsymbol{x}_+)$ et $(\boldsymbol{x}, \boldsymbol{x}_-)$ à travers une fonction softmax. La fonction de coût pour le réseau est donnée par

$$\ell(\boldsymbol{x}, \boldsymbol{x}_+, \boldsymbol{x}_-) = \frac{\|\exp(\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}_+))\|_2^2}{\|\exp(\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}_+))\|_2^2 + \|\exp(\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}_-))\|_2^2}$$

Le réseau triplé est appris en minimisant la fonction de perte globale :

$$\mathcal{L} = \sum_{x \in X} \sum_{x_+ \in V_+} \sum_{x_- \in V_-} \ell(\boldsymbol{x}, \boldsymbol{x}_+, \boldsymbol{x}_-)$$

Le réseau triplé a été appliqué à l'apprentissage supervisé, particulièrement pour la classification d'images sur les données de Mnist (LeCun et al., 1990) et de Cifar10 (Krizhevsky and Hinton, 2009). Les paires d'exemples similaires sont toujours des exemples de même classe, et les exemples dissimilaires appartiennent à des classes différentes. L'entraînement du réseau considère des paires d'exemples similaires et dissimilaires échantillonnées aléatoirement.

6.1.2.1 Discussion

Les approches ci-dessus se fondent sur un oracle déterminant *sans ambiguïté* si deux objets sont similaires ou non. Dans l'application considérée concernant la recommandation d'offres d'emplois,

cette condition n'est pas vérifiée. En effet, nous n'avons pas d'exemples négatifs à proprement parler ; un utilisateur ne déclare pas les offres non pertinentes, et il ne clique pas sur *toutes* les offres pertinentes. Nous verrons dans la section suivante comment tout de même définir une similarité oracle.

6.2 Modèle LaJam

L'approche LaJam, présentée dans Schmitt et al. (2017), consiste à apprendre une métrique sur l'espace des textes (offres d'emploi) considéré. La différence par rapport à l'approche CLE (Annexe A) 3.2.1, Schmitt et al. (2016) est la suivante. Dans le cas de CLE, la métrique oracle est donnée par la représentation des offres obtenue par décomposition de la matrice collaborative (métrique latente). Dans LaJam, la métrique oracle est donnée par la similarité cosinus définie à partir de la matrice collaborative ($sim_{\mathcal{R}}(i,j) = cos(\mathcal{R}_{,i}, \mathcal{R}_{,j})$).

Comme dans le cas de CLE, l'apprentissage de métrique est réalisé par la construction d'une représentation ϕ, plongeant l'espace textuel dans l'espace métrique $\mathbb{R}^d$. Ce plongement dans un espace métrique induit une (pseudo)-distance sur l'espace des textes ; la distance de deux textes x_i et x_j est donnée par la distance euclidienne de leurs images, $||\phi(x_i) - \phi(x_j)||$. Cette similarité permettra de recommander à un utilisateur des offres similaires à celles qu'il a déjà choisies (recommandation de type "plus du même", voir Section 5.2.1).

La finalité de cette métrique est de définir un score de recommandation, permettant de classer les objets i pour un utilisateur u selon l'équation (Équation 3.10) :

$$\widehat{\mathcal{R}}_{u,i} = \sum_{j \in \mathcal{O}} sim_{\phi}(\boldsymbol{x_i}, \boldsymbol{x_j}) \cdot r_{u,j}$$

Notons que l'apprentissage de métrique n'est lié à l'optimisation du score de recommandation que de façon indirecte. Typiquement, le score de recommandation ci-dessous conduirait à privilégier un objet faiblement similaire à beaucoup d'objets choisis par l'utilisateur, par rapport à un objet très similaire à un seul des objets choisis par l'utilisateur. Nous reviendrons sur ce point (Section 7).

Revenons à l'apprentissage de la métrique. Celle-ci est réalisée par l'apprentissage de la représentation ϕ, réalisée par un réseau siamois entraîné pour minimiser une fonction de perte inspirée de la perte contrastive.

Ultimement, nous voulons que la similarité *induite* par ϕ ordonne les objets de la même manière que la similarité *induite* par $\mathcal{R}^T$.

En effet, la perte contrastive (Équation 6.8) est définie pour des similarités booléennes, correspondant à un problème de classification. Par contre, les similarités cosinus fondées sur la matrice collaborative sont des similarités continues, correspondant à un problème de ranking. La perte contrastive est ainsi étendue en définissant pour toute paire d'objets (i,j) :

$$\ell(i,j) = \delta_{i,j}(1 - cos(\phi(\boldsymbol{x_i}), \phi(\boldsymbol{x_j})) + (1 - \delta_{i,j})[m - cos(\phi(\boldsymbol{x_i}), \phi(\boldsymbol{x_j}))]_+$$

où $[A]_+ = max(A, 0)$; $\delta_{i,j}$ vaut 1 ssi les objets i et j sont similaires dans l'espace oracle (ont été tous deux choisis par au moins un utilisateur) ; et m est un scalaire correspondant à la marge. Après des expériences préliminaires, m est pris égal à 0 dans la suite.

Une représentation schématique de l'entraînement est donnée (Figure 6.3). En toute généralité, le réseau ϕ est entraîné pour minimiser la somme des $\ell(i,j)$ pour (i,j) décrivant l'ensemble des paires d'objets :

$$\mathcal{L}_* = \sum_{i \in \mathcal{O}} \sum_{j \in \mathcal{O}} \ell(i,j) \tag{6.9}$$

Cependant, le critère (Équation 6.9) présente deux inconvénients. Le premier est lié au coût computationnel, et le fait que $\mathcal{L}_*$ est quadratique en fonction du nombre d'objets. Le deuxième

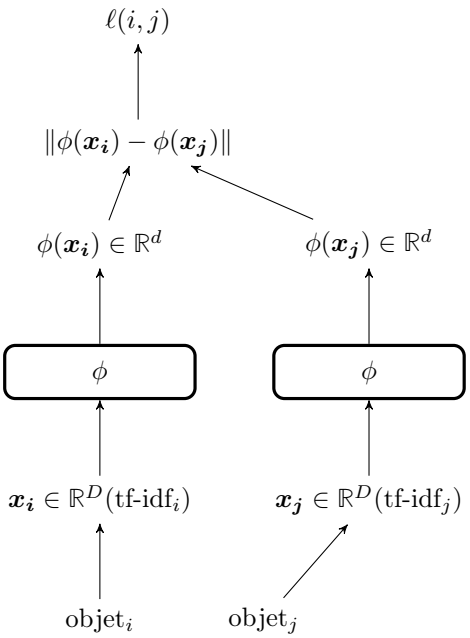


FIGURE 6.3 – Schéma de l'entraînement de LAJAM. Une paire de documents est donnée en entrée du réseau ϕ sous la représentation tf-idf. Le réseau siamois est utilisé afin de calculer la distance cosinus entre les images par ϕ des documents. Enfin, la perte permet de mettre à jour les paramètres du réseau siamois par descente de gradient.

inconvénient est lié au fait que le nombre de paires dissimilaires l'emporte sur le nombre de paires similaires par plusieurs ordres de grandeur en général, ce qui peut distordre la représentation.

De surcroît, les objets sont structurés en petit monde (Section 5.1.2) ; sur les données de CiteULike par exemple, le diamètre est de 2.3, impliquant que deux objets x et x' sont dans la plupart des cas soit directement voisins, soit voisins d'un même objet x". Par transitivité, le critère (Équation 6.9) tendrait donc à contracter la représentation et rapprocher tous les points. Or, le score de recommandation fondé sur une distance $d(x, x')$ ou une distance $h(d(x, x'))$, avec h une fonction scalaire monotone non linéaire, peut être radicalement différent [2].

Il faut ainsi renforcer le rapprochement de deux objets en fonction de leur similarité oracle. Le critère est donc modifié de la façon suivante, permettant de contrôler l'équilibre des voisins positifs et négatifs :

$$\mathcal{L} = \sum_{i \in \mathcal{O}} \sum_{j \in V_+(i) \cup V_-(i)} \ell(i, j) \tag{6.10}$$

—

— Où $V_+(i)$ est l'ensemble des voisins de i par la similarité oracle. Algorithmiquement, n_+ éléments de $V_+(i)$ sont échantillonnés proportionnellement à $sim_{\mathcal{R}}(\cdot, i)$, i.e. selon la loi multinomiale $(p_1, \ldots, p_{|\mathcal{O}|})$ avec $\forall j \in \mathcal{O},\ p_j = \frac{1}{Z} sim_{\mathcal{R}}(i, j)$ (où le terme de normalisation $Z = \sum_j sim_{\mathcal{R}}(i, j)$ assure que les p_i somment à 1). A chaque époque d'entraînement, l'échantillonnage est effectué, permettant de ne pas prendre *tous* les voisins d'un objet. Cet échantillonnage permet de contrôler le nombre de paires voisines et de diminuer l'attraction des objets les plus populaires

— De même, $V_-(i)$ est l'ensemble des non-voisins de i, parmi lesquels n_- objets sont échantillonnés avec $n_- = k_\pm \cdot n_+$. L'hyper-paramètre $k_\pm$ permet de contrôler le poids respectif des termes d'attraction et de répulsion.

La méthodologie suivie, en particulier l'échantillonnage de V_- est très proche de l'échantillonnage négatif (negative sampling) utilisé pour l'apprentissage de la représentation des mots Word2Vec (Mikolov et al., 2013a), où la taille de l'ensemble V_- est également disproportionnée par rapport à celle de V_+ ; où un grand nombre de V_- améliore le score, mais augmente le nombre d'exemples par époque et donc également le temps d'entraînement. Notons que l'échantillonnage proportionnel est équivalent au fait de pondérer chaque terme d'attraction par la similarité $sim_{\mathcal{R}}(i, j)$, avec l'avantage de réduire le coût de calcul.

6.3 Expériences préliminaire sur le rouleau Suisse

L'approche proposée est tout d'abord étudiée sur le problème artificiel du «rouleau Suisse» (Tenenbaum et al., 2000b; Roweis and Saul, 2000).

Pour valider notre approche et notre implémentation de réseau siamois, mais aussi pour visualiser le processus d'apprentissage, nous utilisons un jeu de données artificielles nommé «rouleau Suisse» : on considère un ensemble de points de $\mathbb{R}^3$ décrivant une variété de dimension 2 (Figure 6.4). La similarité oracle entre les points est donnée par leur distance euclidienne dans l'espace $\mathbb{R}^3$. L'objectif est d'établir une représentation $\phi : \mathbb{R}^3 \to \mathbb{R}^2$ conservant les voisinages. L'avantage de ces données est d'être facilement visualisable : nous pouvons juger à l'oeil, à l'aide du code couleur, si

2. Exemple : un utilisateur u a noté trois objets j_1, j_1' et j_2 ($\mathcal{R}_{u,j_1} = \mathcal{R}_{u,j_1'} = \mathcal{R}_{u,j_2} = 1$) tels que $sim(j_1, i_1) = sim(j_1', i_1) = \frac{2}{3}$ et $sim(j_2, i_2) = 1$. Ainsi $\widehat{\mathcal{R}}_{u,i_1} = 1$ et $\widehat{\mathcal{R}}_{u,i_2} = 2 \cdot \frac{2}{3}$ d'où $\widehat{\mathcal{R}}_{u,i_2} > \widehat{\mathcal{R}}_{u,i_1}$. Le fait d'élever la similarité au carré donne la recommandation inverse : $\widehat{\mathcal{R}}_{u,i_1} = 1$ et $\widehat{\mathcal{R}}_{u,i_2} = 2 \cdot (\frac{2}{3})^2$ donc $\widehat{\mathcal{R}}_{u,i_2} < \widehat{\mathcal{R}}_{u,i_1}$.

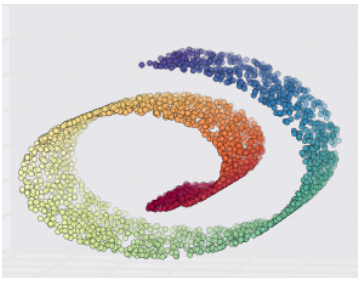
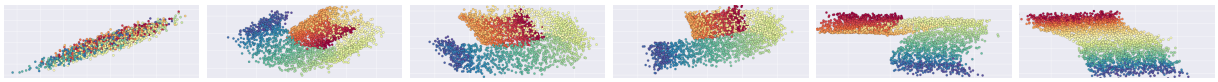


FIGURE 6.4 – Point initiaux dans $\mathbb{R}^3$ du "rouleau Suisse". Les couleurs permettent de visualiser les différentes zones.



(a) Première initialisation

(b) Seconde initialisation

FIGURE 6.5 – Évolution des représentations au cours de l'entraînement dans la configuration élargie. Les images de gauches correspondent à une projection aléatoire des variables d'entrées, les images de droites aux résultats de la représentation après convergence. Les entrées sont constituées des valeurs élargie $X_e = (x, y, z, x^2, y^2, z^2, r_1, r_2, r_3, r_4, r_5, r_6)$. Le réseau comprend une seule couche et réussit à "déplier" le rouleau Suisse.

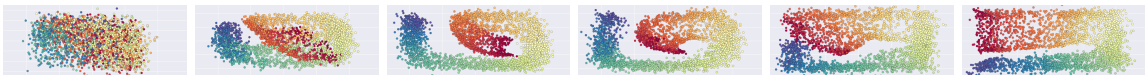


FIGURE 6.6 – Évolution de la représentation dans la configuration réduite (avec les entrées en dimension 3). Le réseau est constitué de deux couches cachées et possède des non-linéarités ($tanh$) entre chaque couche cachée.

la représentation apprise est fidèle aux données. Nous nous attendons en effet à ce que les points rouges soient groupés ensemble et loin des points bleus.

Entraînement Dans ce jeu de données, il n'existe pas de classes associées aux points. Nous définissons donc les exemples positifs de chaque point $\boldsymbol{x}$ comme ses κ plus proches voisins basés sur la mesure euclidienne dans $\mathbb{R}^3$. Les exemples négatifs sont échantillonnés aléatoirement parmi les non-voisins. Nous avons ainsi :

$$\begin{aligned} &\forall \boldsymbol{x} \in X \\ &V_+(\boldsymbol{x}) = \{\boldsymbol{x}_+ \in \kappa\text{-}nn_{\mathbb{R}^3}(\boldsymbol{x})\} \\ &V_-(\boldsymbol{x}) = \{\boldsymbol{x}_- \in X \backslash V_+(\boldsymbol{x})\} \end{aligned} \tag{6.11}$$

Nous définissons les fonctions de pertes suivantes, adaptées au cas euclidien de notre problème :

$$\begin{aligned} \ell_+(\boldsymbol{x}, \boldsymbol{x}_+) &= \|\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}_+)\|_2^2 \\ \ell_-(\boldsymbol{x}, \boldsymbol{x}_-) &= [m - \|\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}_-)\|_2]_+^2 \end{aligned} \tag{6.12}$$

ℓ_+ a pour objectif de réduire la distance *euclidienne* entre deux points voisins, ℓ_- a pour objectif de maintenir une distance euclidienne supérieure à la marge m entre deux points non-voisins. m est choisi égal à la moyenne des distances des points dans l'espace initial et $\kappa = 10$.

Deux configurations sont considérées en entrée du réseau, une option élargie, plus facile avec plus d'entrées, et une option réduite plus difficile avec seulement trois coordonnées :

— **configuration élargie** : Dans la première configuration, les entrées du réseau sont composées de données élargies $X_e = (x, y, z, x^2, y^2, z^2, r_1, r_2, r_3, r_4, r_5, r_6)$. Les entrées r_i sont des valeurs de bruit blanc gaussien, permettant de vérifier que le réseau n'apprend rien à partir de ces entrées. Les entrées (x^2, y^2, z^2) facilite l'apprentissage d'une metrique linéaire. Dans cette configuration, le réseau est composé d'une seule couche, comportant 26 paramètres et réalise donc une projection affine.

— **configuration réduite** : Dans la configuration plus difficile, nous utilisons seulement les coordonnées $(x, y, z) \in \mathbb{R}^3$. Le réseau utilise cette fois une architecture profonde, avec 2 couche cachées de dimensions 3, soit 32 paramètres.

Discussion Nous visualisons l'évolution des représentations (Fig. 6.5 et 6.6). Avec les entrées élargies X_e, nous retrouvons la représentation donnée par LLE, à laquelle nous nous attendons (Figure 6.5). Avec les entrées en dimension 3, le réseau déplie le rouleau suisse, sans toutefois obtenir un dépliement parfait (Figure 6.6). Dans ce cas l'optimisation du réseau est «coincée» dans un minimum local. Ces premiers résultats suggèrent néanmoins que la fonction de perte est pertinente pour apprendre de nouvelles représentations.

6.4 LaJam

Nous décrivons ici notre protocole expérimental (Section 6.4.1) ainsi que notre implémentation adaptée aux documents texte. Nous évaluons LaJam sur les données réelles dans le cas du démarrage à chaud (Section 6.4.2). Enfin, nous l'appliquons dans le cas du démarrage à froid (Section 6.4.3), en nous comparant à l'approche CTR (Section 3.2.2.1), qui a été initialement évaluée sur CiteUlike (Wang and Blei, 2011), sur les données de Qapa et d'ABG.

6.4.1 Protocole expérimental

Cette partie décrit les deux architectures proposées, les hyper-paramètres testés ainsi que la méthode pour intégrer le temps et la géolocalité des offres dans LSA et LaJam.

Architecture L'implémentation a été réalisé à l'aide des librairies de deep learning Tensorflow et Keras (Abadi et al., 2016; Chollet et al., 2015). La représentation ϕ est apprise à l'aide d'un réseau de neurones, pour lequel deux architectures sont considérées.

Hyper-paramètre	valeur choisie	valeurs testées
Taille du vocabulaire	10 000	5 000 - 100 000
Entrée	tf-idf	tf-idf, LSA, LDA
d	200 et 300	50 à 1200
Nombre de couche cachées	0 et 2	0 à 4
Non-linéarité	$tanh$	Relu, sigmoïd, linéaire
Optimisation	Adam	Adam ; SGD
Pas d'apprentissage	.005	.001 à .0001
Taille des batch	200	100 à 20 000
Initialisation	LSA et Glorot	Id
$k_{\pm}$	20	1 à 50
Dropout	0	0 à .9
Batch-normalisation	0	0 et 1

TABLE 6.1 – Table résumant les hyper-paramètres utilisés pour entraîner le réseau siamois LAJAM.

— La première architecture, nommée LAJAM, considère un réseau linéaire dense d'une couche, ce qui correspond à une projection affine :

$$\phi(\boldsymbol{x}) = W \cdot \boldsymbol{x} + \boldsymbol{b} \tag{6.13}$$

$W \in \mathbb{R}^{n_{\mathcal{V}} \times d}$ avec d la taille de l'espace latent et $n_{\mathcal{V}}$ la taille de $\boldsymbol{x}$ dans l'espace initial. Le nombre de paramètres est de 3 000 300 [3]. Notons que l'ajout du biais $\boldsymbol{b} \in \mathbb{R}^d$ n'influence pas dans le calcul de similarité.

— La seconde architecture, nommée deep LAJAM, considère trois couches choisies d'après les résultats de CLE (Annexe). La fonction ϕ s'écrit :

$$\begin{aligned} \chi(\boldsymbol{x}) &= \sigma(W \cdot \boldsymbol{x} + \boldsymbol{b}) \\ \chi'(\boldsymbol{z}) &= \sigma(W' \cdot \boldsymbol{z} + \boldsymbol{b'}) \\ \phi(\boldsymbol{x}) &= \chi' \circ \chi' \circ \chi(\boldsymbol{x}) \end{aligned}$$

Avec la fonction d'activation $\sigma = tanh$. $W \in \mathbb{R}^{n_{\mathcal{V}} \times d}$ et $W' \in \mathbb{R}^{d \times d}$. Le nombre de paramètres est de 3 090 900.

Enfin, dans ces deux cas, la similarité sur l'espace image est la similarité cosinus (*i.e.* $sim_\phi(i,j) = cos(\phi(\boldsymbol{x_i}), \phi(\boldsymbol{x_j}))$.

Le choix des hyper-paramètres explorés et sélectionnés est récapitulé (Table 6.1).

Avec Temps et Géolocalité Un des avantages des réseaux siamois est la possibilité d'ajouter des données numériques d'origine différente. Cela a été validé avec les données du rouleau suisse, où les carrés des coordonnées ont été utilisés de façons pertinente, alors que les bruits blancs ont été ignorés. Nous nous posons alors la question de savoir si les données auxiliaires disponible pour chaque offre d'emploi, comme la date de publication et la position géographique, permettent aussi d'améliorer les recommandations.

Sur les données de Qapa et d'ABG, nous ajoutons ainsi aux descripteurs d'entrée (mots représentés par tf-idf) la date de mise en ligne de l'offre ainsi que les coordonnées géographique associées

3. En considérant l'espace initial des mots de dimension 10000 et l'espace latent résultat de dimension 300

à l'offre (longitude et latitude) [4]. Afin de se comparer de manière équitable à la représentation LSA nous intégrons également ces données dans le calcul de similarité LSA en definissant une nouvelle similarité sim^*_{LSA} :

$$sim^*_{LSA} \leftarrow sim_{LSA} - \lambda_g d_g(i,j) - \lambda_t d_t(i,j) \tag{6.14}$$

Où $d_g(i,j)$ représente la distance géographique entre les objets i et j et $d_t(i,j)$ est la distance temporelle des deux objets. Les deux paramètres λ_g et λ_t sont optimisés par une recherche en grille sur l'ensemble d'entraînement pour maximiser le rappel à 100. Ces pénalités ont pour effet, lors de la recommandation à base de mémoire, d'éviter d'attribuer une similarité trop forte à deux offres géographiquement ou temporellement éloignées. Notons que nous utilisons ici un savoir d'expert sur l'importance de la distance et du temps pour améliorer la similarité LSA. Ce savoir n'est pas fourni *a priori* lors de l'entraînement de LaJam. De plus, les coordonnées longitude et latitude sont données brutes en entrée du réseau de neurones, alors que sim^*_{LSA} dispose directement de la distance entre offre et demande.

6.4.2 Démarrage à chaud

Nous illustrons les performances de LaJam comparativement au filtrage collaboratif (Équation 3.6) ainsi qu'à la recommandation basée sur le contenu LSA pour le problème du démarrage à chaud (Section 5.1.1). Bien que le réseau soit conçu pour le problème du démarrage à froid, il peut en effet également être appliqué dans ce cas plus simple, même s'il n'utilise pas les informations complémentaires disponibles (les informations de la matrice collaborative disponibles pour le nouvel individu).

Les expérimentations, reportées (Figure 6.7), montrent que LaJam obtient de meilleurs résultats que LSA quelque soit le jeu de données et le niveau de rappel considéré. Sur Qapa, les scores LSA sont nettement améliorés, sans toutefois atteindre les scores de filtrage collaboratif pour les rappels inférieurs à 400. Au-delà de 400 recommandations, LaJam présente plus d'objets pertinents que le filtrage collaboratif. Cela vient du fait que les similarités collaboratives entre objets sont nulles pour presque toutes les paires (50% des objets ont moins de 50 voisins (Figure 5.3)), alors que les similarités LSA entre objets sont denses [5].

Sur ABG, nous observons une grande différence entre les scores du filtrage collaboratif et ceux de LSA. En effet, l'étalement temporel des données n'est pas pris en compte dans le cas de LSA ni de LaJam dans ces expériences. Nous remarquons que notre approche permet une meilleure représentation du langage des offres de thèse (objet sur ABG).

Sur CiteUlike, les scores de LaJam améliorent ceux de LSA et se rapprochent des scores du filtrage collaboratif. Tous ces résultats sont toutefois très resserrés du fait de la bonne qualité des documents textes pour mesurer les similarités.

6.4.3 Démarrage à froid

Nous présentons ici les résultats principaux de notre approche, à savoir les performances de recommandation lors du démarrage à froid objet, reprenant le protocole défini (Section 5.2.1).

4. Le plus souvent, nous avons le nom d'une ville de France, pour laquelle nous retrouvons ces coordonnées à l'aide d'une table de correspondances (Section 4.2.3) et (Figure 4.1).

5. Notons que la représentation dense venant de la décomposition de la matrice $\mathcal{R}$ à l'aide d'une SVD (Équation 3.2) n'obtient pas de scores meilleurs que la similarité sur $\mathcal{R}$.

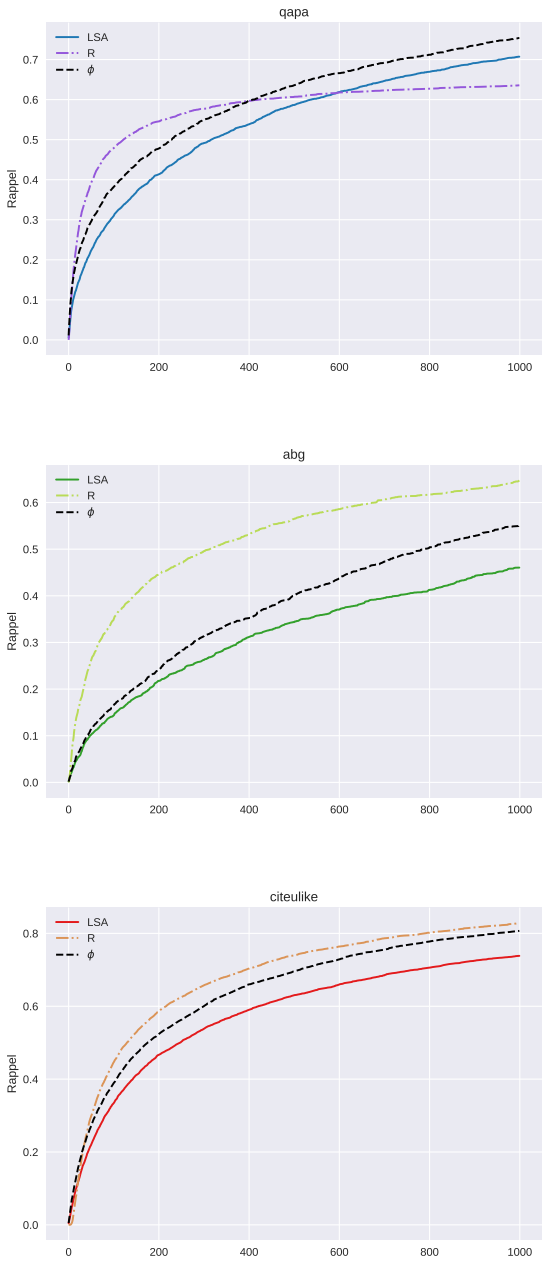


FIGURE 6.7 – Les courbes de rappel de LSA (train plein), du filtrage collaboratif (pointillé) ainsi que LAJAM (noir pointillé) dans le cas du démarrage à chaud. L'apprentissage du réseau permet d'améliorer les scores de LSA, s'approchant des scores du filtrage collaboratif.

	R@20	R@100
LSA	$.332 \pm .001$	$.631 \pm .003$
LDA	$.247 \pm .005$	$.584 \pm .005$
CTR	$.271 \pm .001$	$.587 \pm .001$
LaJam	$\mathbf{.341 \pm .003}$	$\mathbf{.650 \pm .004}$
deep LaJam	$.279 \pm .005$	$.606 \pm .006$

Table 6.2 – Performance en démarrage à froid objet sur CiteUlike, pour LSA, LDA, CTR et LaJam et deep LaJam. Validé sur 5 validation croisée. Le temps d'entraînement pour LaJam (respectivement deep LaJam) est de 10 minutes (resp. 20 minutes).

CiteUlike Les résultats obtenus pour les valeurs de rappel à 20 et à 100 sur les données issus de CiteUlike dans le cas du démarrage à froid objet sont regroupés (Table 6.2). De manière surprenante, CTR n'améliore que peu LDA, et LSA obtient des performances suppérieures à LDA (comme remarqué (Section 5.2.1)) et CTR. La représentation LaJam obtient les meilleurs scores, en particulier par rapport à LSA. Notons que l'architecture profonde n'améliore pas les performances.

Qapa & ABG Les performances de LaJam sur les données de Qapa et ABG sont comparées avec LSA [6] (Table **??**). Dans le cas où les entrées ne considèrent pas d'informations autres que les mots (colonne de gauche dans Table **??**), sur Qapa LaJam améliore de manière significative sur LSA d'environ 9% pour le rappel à 20 et de 5% pour le rappel à 100. Sur ABG, LaJam est dominé par LSA de 3% sur le rappel à 20 et améliore LSA de 2% sur le rappel à 100.

Analyse des hyper-paramètres Nous résumons ici les observations sur la sensibilités des hyper-paramètres :

- L'*entrée* est constituée des poids issus de tf-idf. De même que pour CLE, la représentation LSA en entrée obtient des performances proches, de l'ordre d'un point inférieur, réduisant le nombre de paramètres d'un facteur 30.
- d est fixé à 200 dans le cas de CiteUlike afin de pouvoir nous comparer équitablement à CTR. Pour les autres jeux de données, nous utilisons $d = 300$ (Section 5.2.1).
- La non-linéarité *tanh* est plus stable lors de l'optimisation des autres hyper-paramètres.
- L'optimisation utilise la méthode Adam (Kingma and Ba, 2014), plus rapide que SGD précédemment considéré.
- La taille des batch, fixé à 200, n'est pas un paramètre sensible.
- L'initialisation de la première couche aux valeurs LSA (Section 2.2.3) permet au réseau de converger plus rapidement et obtient de meilleurs performances.
- Les performances augmentent avec $k_\pm$ puis sont stables pour $k_\pm \geq 10$.
- Batch-normalisation et Dropout n'ont pas permis d'améliorer les performances.

Analyse de la sensibilité avec la date et la géo-localité Le résultat le plus frappant concerne l'utilisation de la date et de la position géographique dans les données d'entrée, en particulier l'ajout de la géo-localité pour les données de Qapa. Ce résultat montre une augmentation de 20% par rapport à LSA pour le rappel à 20, et de 15% pour le rappel à 100. Cependant, l'impact de la géo-localité est insignifiant pour les données d'ABG. Tout cela est compréhensible à la lumière des

6. Les autres méthodes sont omises car leurs résultats sont plus faibles que LSA, voir (Section 5.2.1)

Qapa					
	–	geoloc	date	geo+date	CPU
LSA R@20	.404 ± .007	**.636 ± .003**	.439 ± .007	**.656 ± .003**	5 min
LSA R@100	.694 ± .005	**.829 ± .003**	.713 ± .005	**.835 ± .002**	
LaJam R@20	**.495 ± .006**	.549 ± .006	**.523 ± .005**	.558 ± .006	200min
LaJam R@100	**.743 ± .007**	.784 ± .005	**.764 ± .006**	.790 ± .005	
ABG					
	–	geoloc	date	geo+date	CPU
LSA R@20	**.254 ± .010**	**.258 ± .008**	**.574 ± .008**	**.579 ± .009**	5 min
LSA R@100	.522 ± .012	.528 ± .013	**.814 ± .007**	**.817 ± .007**	
LaJam R@20	.226 ± .008	**.260 ± .009**	.391 ± .009	.392 ± .011	25min
LaJam R@100	**.544 ± .008**	**.599 ± .006**	.761 ± .009	.755 ± .008	

Table 6.3 – Démarrage à froid objet, rappel à 20 et à 100 pour Qapa et ABG . Validé sur 10 validation croisés. Le temps d'entraînement est reporté sur la colonne de droite ; une fois les scores calculés, générer les recommandations (pour tous les utilisateurs) sur Qapa est de l'ordre de la dizaine de secondes et de l'ordre de la seconde pour ABG.

analyse sur la géo-localité (Figure 4.4) et (Figure 4.5). En effet, les emplois à faible valeur ajoutée sont en général moins amenés à se déplacer, contrairement aux jeunes docteurs.

Inversement, l'information de la date de publication a un impact fort pour les recommandations chez ABG, augmentant le score de rappel à 20 de 32% par rapport à LSA, et 30% pour le rappel à 100. Cet impact est plus modéré pour les résultats chez Qapa (améliorant de 2%). Ces résultats sont encore bien expliqués par l'analyse (Figure 4.4). En effet, les données de Qapa s'étendent sur une période de 3 mois, alors que les offres chez ABG s'étendent sur 5 ans. Ces résultats confirment également l'un des avantages principaux du filtrage collaboratif [7] (Section 3.2.1) par rapport aux heuristiques d'ajustement, à savoir sa capacité de s'adapter aux spécificités des secteurs de chaque base de données.

Les bonnes performances de LSA avec les descripteurs complémentaires viennent du fait que la géo-localité et la date de publication sont exploitées avec des poids optimaux (Équation 6.14), contrairement à LaJam qui apprend, à partir de zéro et sans a priori, comment prendre en compte le meilleur de ces données temporelles et géographie (cela correspond à 3 valeurs scalaires d'entrée parmi les 10 000 valeurs représentant les mots).

6.5 Discussion

LaJam apprend une nouvelle représentation des offres d'emploi et des articles scientifiques, prenant en compte les informations issues des données collaboratives. LaJam intègre facilement les données de différents types, comme la géo-localité et la date de publication, ce qui améliore

7. Car LaJam est entraîné sur les données collaboratives.

significativement ses recommandations. Ses performances sont bien établies sur les jeux de données propriétaires ainsi que sur les données publiques de CiteUlike. Un point reste cependant à explorer, les sous-performances du réseau profond. Dans ce cas, une régularisation plus sophistiquée doit être mise en place. Dans le cas particulier de CiteUlike, l'effet petit monde rencontré, de degrée 2.4, nous avertit par ailleurs que deux objets voisins de degrée 2 ne doivent pas a priori être considérés comme similaires.

Différence entre CLE **et** LaJam Dans le cas de CLE les voisins positifs sont fixés aux 10 premiers voisins, avec une pondération $\mathcal{W}$ optimisée pour reconstruire chaque point comme *barycentre* de ses voisins. Dans ce cas, si trois proches voisins sont colinéaires, seulement un de ces voisins sera affecté d'un poids fort, contrairement à LaJam qui prend en compte tous les voisins positifs, avec une pondérations dépendant maintenant de leur similarité collaborative. Ainsi, si l'approche CLE tente de conserver la *structure* de voisinage de l'espace collaboratif,LaJam tente de mimer les *similarités* collaboratives. Enfin, l'échantillonnage négatif dans LaJam est tronqué par une marge dans la fonction de perte, contrairement à celle de CLE qui repousse tous les points indépendamment.

Chapitre 7

Challenge RecSys

Ce chapitre est consacré au challenge RecSys 2017 (dont les données sont décrites Section 4.2.2). Ce chapitre décrit l'approche gagnante, ainsi que notre approche à base de mémoire classée 15^{eme} (sur 100 participants) [1]. Ce challenge a mis en avant les différentes problématiques d'ingénierie lié à la gestion de grandes bases de données.

Modèle gagnant La première place est obtenue à l'aide d'un modèle d'appariement direct, reposant sur un boosting d'arbres de décision [2] (Volkovs et al., 2017). Ce modèle est entraîné sur le problème de classification booléen : la classe positive correspond à une des interactions positives et la classe négative à l'action de supprimer l'objet. Les entrées du modèle sont les caractéristiques de l'utilisateur, les caractéristiques de l'objet ainsi que des caractéristiques manuellement construites appelées "interactions croisées utilisateur-objet" pour chaque paire d'utilisateur-objet. Ces interactions croisées qui représente l'une des principales contributions de ces travaux sont données par :

$$\boldsymbol{inter}(u,i) = \frac{1}{|\mathcal{O}(u)\backslash\{i\}|} \sum_{j \in \mathcal{O}_{(u)\backslash\{i\}}} \boldsymbol{s_{ij}}$$

$\boldsymbol{s_{ij}}$ est un vecteur de dimension 11×6 (6 représente chaque type d'interactions), mesurant les similitudes des objets i et j sous différents aspects : les 11 dimensions correspondent aux nombres de termes communs entre les "tags" et les titres ; à l'égalité des valeurs de niveau de carrière ; de fonction ; de type d'emploi ; de pays et de région ; la différence du niveau de carrière et enfin, la plus importante, la distance géographique entre ces deux objets.

Ces caractéristiques croisées permettent un gain considérable de score (300 %) par rapport à la seule considération des caractéristiques brutes des objets et des utilisateurs. Cependant, ces informations ne sont disponibles que dans le cas où l'utilisateur a déjà noté des objets. En ce sens, ces caractéristiques sont proches des informations collaboratives (Équation 3.6), où $\widehat{\mathcal{R}}_{ui} = \sum_{j \in \mathcal{O}_{(u)\backslash\{i\}}} sim(i,j)$. Ici, $\boldsymbol{s_{ij}}$ est un vecteur et $\widehat{\mathcal{R}}_{ui}$ est un fonction de $\boldsymbol{inter}(u,i)$.

Une autre contribution est l'intégration de la dernière "interaction croisée utilisateur-objet" en date ($\boldsymbol{s_{ij}}$ pour la dernière interactions j), permettant de prendre en compte les éventuelles évolutions temporelles des utilisateurs. L'inspection a posteriori [3] des caractéristiques du modèle met en avant

1. Nous avons participé sous le nom d'équipe *Taoist*.
2. Utilisant l'excellente librairie *xgboost* (Chen and Guestrin, 2016)
3. En termes d'importance des caractéristiques sur les arbres de décision une fois le modèle entraîné.

l'utilisation principale de l'information de la *distance* de la *dernière* offre noté par un utilisateur.

Du plus, pour avoir une distribution de classe équilibrée, un échantillonnage aléatoire parmi les objets est effectué pour augmenter la classe négative. De plus, l'apparition de chaque objet est équilibrée, évitant la sur-représentation des objets populaires et permettant un gain de 10%. Notons que cette même méthode utilisant un réseau de neurones n'a pas donné de résultats satisfaisants. L'instabilité est mise en cause, aussi remarquée par (Covington et al., 2016).

Notre modèle Nous utilisons la matrice collaborative pondéré avec les poids de la fonction de valeur RecSys (Section 4.2.2) pour la recommandation à base de mémoire (Équation 3.10). Nous apprenons donc une similarité entre objets sans utiliser les informations des utilisateurs autres que leurs interactions. Les données du texte libre des objets sont concaténées avec les données catégorielles (e.g. tranche de salaire, niveau d'étude). Ces descripteurs sont ensuite pondérés par tf-idf et enfin la représentation des objets est le résultat de la factorisation de cette matrice par une décomposition en valeurs singulière, en cela semblable à LSA. La similarité dans l'espace latent est l'objet d'une étude particulière. En effet, nous modifions la formule de mémoire (Équation 3.6) de manière à affiner la similarité :

$$\widehat{\mathcal{R}}_i = \sum_j \mathcal{R}_{uj}\ h(\boldsymbol{x_i}, \boldsymbol{x_j}) \tag{7.1}$$

h évalue la similarité entre les deux représentations des objets i et j. Dans le cas de la similarité cosinus nous avons $h(\boldsymbol{x_i}, \boldsymbol{x_j}) = cos(\boldsymbol{x_i}, \boldsymbol{x_j})$. Dans ce chapitre, les fonctions suivantes sont comparées :

- $h_{cos}(\boldsymbol{x_i}, \boldsymbol{x_j}) = cos(\boldsymbol{x_i}, \boldsymbol{x_j}) \cdot \mathbb{1}_{j \in \kappa\text{-}nn(i)}$
- $h_{euclidien}(\boldsymbol{x_i}, \boldsymbol{x_j}) = -\left\|\boldsymbol{x_i} - \boldsymbol{x_j}\right\|_2^2 \cdot \mathbb{1}_{j \in \kappa\text{-}nn(i)}$
- $h_{exp}(\boldsymbol{x_i}, \boldsymbol{x_j}) = \exp(\frac{-\|\boldsymbol{x_i} - \boldsymbol{x_j}\|_2}{\sigma})$
- $h_{Gauss}(\boldsymbol{x_i}, \boldsymbol{x_j}) = \exp(\frac{-\|\boldsymbol{x_i} - \boldsymbol{x_j}\|_2^2}{\sigma^2})$
- $h_{LaJam}(\boldsymbol{x_i}, \boldsymbol{x_j}) = cos(\phi(\boldsymbol{x_i}), \phi(\boldsymbol{x_j}))$

Notons que le paramètre κ [4] permet de spécifier le nombre de termes intervenant dans le calcul de similarité (Équation 7.1). Le paramètre σ contrôle l'influence des plus proches voisins dans le score global $\widehat{\mathcal{R}}$. κ et σ sont optimisés sur un ensemble de validation. ϕ est optimisé de la même manière que dans la Section 6. Dans les cas du filtrage collaboratif (Section 3.2.1), de LSA (Section 5) et de LaJam (Section 6), la similarité est donnée par h_{cos} avec $\kappa = \infty$. Cependant, l'enrichissement de la représentation avec les données catégorielles entraîne une similarité cosinus trop élevé, d'où la nécessité empirique de restreindre le nombre de voisins κ lors de la recommandation. Notons que (Aiolli, 2012) utilise la famille exponentielle pour optimiser la décroissance de la similarité cosinus $h_q = cos(\boldsymbol{x_i}, \boldsymbol{x_j})^q$, non étudiée ici.

Double mémoire Une des limitations de notre approche vient du fait que nous ne pouvons pas recommander d'offres aux utilisateurs sans interactions (démarrage à froid utilisateur, matrice F (Table 7.1)). Une option pour contourner cette limitation est considérée : nous remplissons *artificiellement* certaines interactions de ces utilisateurs démarrant à froid (matrice C) par la méthode de filtrage collaborative orientée utilisateur (nous utilisons ainsi les matrice A et B pour reconstruire la matrice C). Cela créée un sous-problème de recommandation, où les interactions générées sur C

4. Proche de celui donnée dans (Deshpande and Karypis, 2004) (Équation 3.8)

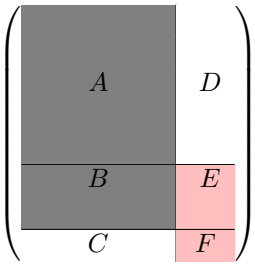


TABLE 7.1 – Représentation de la matrice $\mathcal{R}$. Les matrices A et B correspondent aux données d'entraînement. Les matrices E et F sont les valeurs à reconstruire pour générer des recommandations. Les matrices D, E et F (correspondant aux objets de tests) sont vides, cas du démarrage à froid objet. La matrice C est vide, elle correspond au démarrage à froid utilisateur.

servent de donnée d'entraînement pour le système final. Cependant, des problèmes de normalisations apparaissent, en effet, comme nous devons recommander une liste d'*utilisateurs* par offre, une trop forte reconstruction de la matrice C donne un poids plus important à F par rapport à E. De plus, il est fort possible que les utilisateurs correspondant aux matrices C et F soient biaisés, ayant une plus faible activité (i.e. que F soit moins remplie que E). De plus, l'erreur commise sur $\hat{C}$ se multiplie sur l'erreur sur $\hat{F}$. Les résultats préliminaires montrent qu'il est toujours plus avantageux de recommander des utilisateurs associés à E plutôt qu'à F.

Ensemble Nous avons également élaboré une méthode d'ensemble, en ajoutant une approche s'appuyant sur le score direct entre les termes des objets et des utilisateurs. La solution d'ensemble est obtenue en moyennant les deux scores $\widehat{\mathcal{R}} = \alpha\widehat{\mathcal{R}}_1 + (1-\alpha)\widehat{\mathcal{R}}_2$. α est choisit par validation croisée. Cependant, la première heuristique étant de qualité trop faible, le score de notre méthode principale n'est pas amélioré.

xgboost La méthode à base de boosting sur les arbres de décision implémenté par *xgboost* est également considéré. Avec les entrées correspondant aux représentations brutes des utilisateurs et des objets. Cependant, comme remarqué par les gagnants (Volkovs et al., 2017), prédire la matrice de score et établir des recommandations prend un jour de calcul sur un serveur de calcul à 32 CPU. Ces expériences préliminaires n'ont pas abouti.

Résultats & leçons apprises De manière inattendue et décevante, notre réseau LAJAM adapté pour le challenge n'a pas obtenu de meilleurs résultats que LSA (Table 7.2). Une explication de la sous-performance de LAJAM, aussi notée dans (Covington et al., 2016; Volkovs et al., 2017) est que certains paramètres mal calibrés (comme l'équilibrage des distribution d'objets ou la mise à l'échelle des valeurs d'entrées) peuvent s'avérer fatals pour le modèle. De plus, les informations de date et de géo-localité sont très importantes. Également, notre échantillonnage négatif n'a peut-être pas été assez bien calibré, entraînant sur-représentation de certains objets.

Les principales leçons de ce challenge sont :

	tableau des score	Rang final
(Volkovs et al., 2017)	75 782	1^{er}
h_{exp} ($\sigma = .1$)	36 129	16^{eme}
h_{Gauss} ($\sigma = .1$)	35 713	16^{eme}
$h_{euclidien}$ ($\kappa = 500$)	31 731	20^{eme}
$LaJam_{euclidien}$	27 472	23^{eme}
h_{cos} ($\kappa = 500$)	25 835	26^{eme}

TABLE 7.2 – Résultats de la phase hors ligne. Les scores ainsi que les rangs sont mesurés sur le tableau des score `https://recsys.xing.com/leaders#offline`

— L'importance de l'optimisation du code, aussi bien au niveau des limites mémoire [5] qu'au niveau des temps de calcul [6] ;
— L'utilisation des serveurs de calcul ainsi que des planificateurs de tâches (job schelduler)
— L'utilisation extensive de structure pour collaborer, utilisant le logiciel de gestion de version git ;
— La mise en place et la maintenance d'un workflow branché sur une API délivrant des informations sur un rythme journalier.

5. les produits matriciels sont trop coûteux pour tenir en RAM. Nous les avons décomposé par bloc, ce qui complexifie le code.
6. Remplir une matrice de la taille des données de test peut prendre une demi-journée si cela est fait naïvement.

Conclusion

La thèse présentée, consacrée à la recommandation de postes ouverts à des demandeurs d'emploi, comporte plusieurs aspects originaux, théoriques et applicatifs.

Acquisition et traitement de données réelles

Du point de vue applicatif, notre première spécificité est d'avoir travaillé sur des données réelles, de grande taille et très hétérogènes quant aux secteurs d'emploi considérés. Que les personnes nous ayant donné accès à ces données soient ici encore chaleureusement remerciées pour leur confiance et leur aide. Seules les informations des clics des demandeurs d'emploi sur les offres (et non les auditions des candidats par les entreprises) ont été considérées dans le cadre de notre étude ; nous reviendrons sur ce point dans la partie Perspectives.
Les données Qapa concernent le domaine du travail par intérim et sont relatives au secteur dit des "cols bleus", des emplois à faible valeur ajoutée. Comme nous l'avons souligné, il s'agit d'un secteur applicativement très important : En 2015, 65% des nouveaux contrats durent moins de 12 mois [7]. Il importe de noter que ce secteur de recommandation d'emploi à notre connaissance a été peu ou pas étudié dans la littérature (chapitre 3) comparé à la recommandation d'emplois à haute valeur ajoutée ; une raison en est que ce secteur de recommandation est peu profitable [8]. Indépendamment, les données disponibles présentent des difficultés particulières. En premier lieu, les CVs des demandeurs sont plus hétérogènes et moins formatés que pour les emplois à haute valeur ajoutée. En second lieu, les CVs contiennent aussi moins de signal bien répertorié − par exemple en termes de parcours éducatif et de postes occupés par les demandeurs dans le passé. Enfin, les choix faits par les demandeurs témoignent d'une plus grande diversité (une personne peut candidater à la fois sur des postes de vendeur et de réceptionniste), ce qui pénalise l'identification des profils collaboratifs.
Nous avons également considéré des données issues du secteur des "cols blancs" et plus spécifiquement, concernant les emplois de docteurs (PhD) dans l'industrie (données de l'Association Bernard Gregory, ABG). Les caractéristiques des données sont bien différentes des précédentes. En premier lieu, les CVs font intervenir des vocabulaires rares (e.g., les titres des thèses). En second lieu, ces domaines de spécialistes sont à évolution rapide (la période couverte par les données était 2010-2014).
Deux autres contextes applicatifs ont été considérés au titre de la validation comparative avec les algorithmes de l'état de l'art : celui de la recommandation d'articles scientifiques (données CiteULike

7. http://dares.travail-emploi.gouv.fr/IMG/pdf/2017-028.pdf
8. Pour fixer les idées, le coût d'un chasseur de tête sur un emploi donné est de l'ordre du salaire annuel correspondant à l'emploi considéré.

Wang and Blei (2011)) et le challenge RecSys 2017, lui aussi consacré à la recommandation d'emploi, mais avec la différence que le pre-traitement des données a été effectué au préalable, chaque offre d'emploi/CV se ramenant à un vecteur représentant des données catégorielles. Le premier domaine confirme que les documents offres d'emplois et CVs diffèrent singulièrement des articles scientifiques, comme on pouvait s'y attendre. Le second domaine met en valeur la qualité des approches algorithmiques que nous avons considérées (notre algorithme de base se qualifiant comme le 15^e de la compétition, qui a attiré plus de 100 participants), ainsi que l'importance du passage à l'échelle dans ce domaine (les algorithmes originaux que nous avons développés, voir ci-dessous, n'ont pu être appliqués sur les données RecSys).

Position de la recherche

Du point de vue algorithmique, notre recherche se situe au confluent de deux domaines, la recommandation et le filtrage collaboratif d'une part, et le traitement des langues naturelles d'autre part : il s'agit de "recherche d'information par méthode collaborative" (collaborative information retrieval) à proprement parler.
Deux hypothèses de travail restrictives ont été considérées. Tout d'abord, en toute généralité la recommandation d'emploi s'attaque au problème de la recommandation à froid, cherchant à recommander des offres nouvelles à des demandeurs nouveaux. Nous nous sommes limités au cas de la recommandation d'offres nouvelles à des demandeurs connus ; cet objectif est approprié au cas du secteur de l'intérim, dont nous avons rappelé ci-dessus l'importance. En second lieu, sous l'angle du traitement des langues naturelles nous nous sommes limités à des approches de type sacs-de-mot, telles l'analyse sémantique latente (LSA) et l'allocation latente de Dirichlet (LDA) d'une part, et à des approches neuronales d'autre part : l'analyse fine de la sémantique des textes n'a pas été abordée.

Contributions algorithmiques

Fondamentalement, la tâche visée consiste à définir une représentation des textes disponibles, et une métrique sur cette représentation, telles que la stratégie consistant à recommander une offre proche des offres déjà sélectionnées par un CV au sens de cette métrique soit expérimentalement satisfaisante.
Plusieurs pistes algorithmiques ont été abordées et deux d'entre elles ont conduit à des approches efficaces. La première, CLE (Schmitt et al. (2016)), reprend les principes des changements de représentation localement linéaires Roweis and Saul (2000). Spécifiquement, CLE considère la représentation de la matrice collaborative (indiquant quel demandeur a cliqué sur quelle offre) comme une représentation "oracle". La question devient alors de plonger la représentation primaire (sac de mots, tf-idf) ou secondaire (LSA, LDA, Doc2Vec) des documents dans la représentation oracle. La limite de cette approche est la qualité de la représentation oracle, reflétant *in fine* la qualité des données collaboratives. Les expériences de recommandation à chaud (considérant des offres et des demandeurs déjà connus) mettent en évidence un plateau de qualité de la recommandation : même dans le cas favorable de la recommandation à chaud, les recommandations plafonnent et circa 45% des demandeurs n'obtiennent pas de recommandation pertinente dans les 1,000 premières suggestions. La qualité insuffisante de l'oracle pénalise naturellement les performances d'un apprentissage

cherchant à reproduire l'oracle.
La seconde approche, LaJam (Schmitt et al. (2017)), tire les conclusions des expériences précédentes pour chercher directement à partir des représentations primaires ou secondaires des documents d'offres d'emploi, un changement de représentation reflétant la similarité donnée par la matrice collaborative − où deux offres sont similaires si elles sont sélectionnées par au moins un utilisateur. Une approche de type réseaux siamois Bromley et al. (1994) est adaptée à cet objectif, se focalisant sur l'apprentissage de représentation d'offres d'emploi reflétant la similarité ci-dessus. L'approche consistera donc à recommander des offres similaires aux offres déjà considérées par l'utilisateur ("plus du même"). Cette approche donne de bons résultats, d'une part sur les données publiques CiteULike avec une amélioration de près de 20% sur le score de recommandation à 20 (le Recall@20 passant de 27% à 33%), et d'autre part sur les données Qapa et ABG, où la proportion de demandeurs n'obtenant pas de recommandation pertinente dans les 1,000 premières suggestions passe à 25%. Ces performances reposent sur une étude et une optimisation approfondie de la sélection des exemples négatifs (paires d'offres non similaires) permettant d'entraîner le réseau siamois. Les paires négatives forment la très grande majorité des paires d'offres, comme c'est le cas général, et ces paires doivent donc être sélectionnées agressivement dans un souci de qualité computationnelle. Or la sélection des paires négatives soulève un problème lié à la structure de "petit monde" de l'ensemble des offres. Formellement, compte tenu de la dispersion et de la versatilité des clics des utilisateurs − particulièrement sur les données de Qapa, le diamètre du graphe de voisinage des offres est réduit (de l'ordre de 5 ; pour fixer les idées, il est de l'ordre de 3 sur CiteULike et de 4 sur RecSys). Nous avons donc proposé une stratégies de tirage des paires d'offres négatives.

D'un point de vue applicatif, l'approche LaJam permet d'intégrer naturellement les informations hétérogènes figurant dans les offres, telles la localisation géographique et le temps depuis lequel l'offre a été postée. L'intérêt d'une approche par apprentissage automatique, dédiée au secteur d'applications, est confirmée par l'importance extrêmement différente de ces informations temporelles et géographiques en fonction du contexte. Typiquement, la proximité géographique est essentielle dans le contexte Qapa et d'importance négligeable dans le contexte ABG, ce qui s'explique par la mobilité géographique attendue des PhDs et post-docs.

Une seconde leçon intéressante issue des données concerne la différence de langage entre les offres d'emploi et les CVs. Ces deux corpus pourraient littéralement être vus comme appartenant à des langues différentes : une analyse LSA classique situe les offres et les CVs d'un même sous-secteur d'emploi (e.g. le designer graphique) dans des régions différentes de l'espace $\mathbb{R}^d$. Cette différence explique et confirme *a posteriori* la nécessité d'exploiter les données collaboratives pour recommander les offres d'emploi.

D'un point de vue algorithmique, une des leçons tirées des données concerne les notions de distance et de norme les plus appropriées au contexte et à l'objectif visé. Ainsi, la notion de similarité cosinus se révèle plus robuste que la distance L_2 pour l'apprentissage de réseau siamois, dans le but de résister à l'hétérogénéité des données et de s'adapter à des textes de longueur différente (pour lesquels le fait d'imposer une représentation de norme 1 aurait pu être contre-productive). De manière liée, dans le contexte de CiteULike la représentation LDA se révèle surprenamment moins efficace qu'une représentation LSA classique ; une explication avancée pour cette contre-performance est le fait que LDA projette les documents dans la boule L_1.

Enfin, une piste qui a été explorée dans notre étude consistait à enrichir les données en fonction de ressources externes, telles l'ontologie ROME [9]. De manière inattendue, l'enrichissement des textes

9. Répertoire Opérationnel des Métiers et des Emplois, disponible à l'adresse `http://www.pole-emploi.fr/`

ainsi effectué n'a pas permis d'améliorer les performances, pour deux raisons. En premier lieu, l'information des offres d'emploi était redondante avec celle des fiches métier dans la plupart des cas. En second lieu, dans les cas des métiers relativement nouveaux, tels les métiers de communication, l'information ROME était trop générale (e.g. ne distinguant pas les niveaux hiérarchiques) pour être utile.

Perspectives

Notre travail ouvre des perspectives de recherche selon plusieurs facettes, à court, moyen et long terme.

Parmi les perspectives de recherche algorithmique à moyen terme les plus essentielles à notre avis, figure l'étude des biais des données. Plus précisément, deux types de biais sont identifiés. Le premier est du au fait que les données disponibles sont en général issues de systèmes de recommandation (par exemple, issu d'un algorithme propriétaire pour Qapa). Les offres recommandées aux demandeurs, filtrées par les algorithmes du système de recommandation, présentent ainsi un problème de données censurées. Typiquement, l'utilisateur ne clique pas sur les offres qui ne lui sont pas proposées. En second lieu, les systèmes collaboratifs futurs devront aussi exploiter les informations relatives aux préférences des entreprises, et des candidats auditionnés sur un poste. Ainsi que nous l'avons dit précédemment, seule l'information des clics des demandeurs sur les offres d'emploi a été utilisée dans notre recherche, en se fondant sur le fait que les deux problèmes de recommandation aux recruteurs et aux demandeurs diffèrent fortement en pratique (sinon d'un point de vue abstrait). Or, les préférences des entreprises sont susceptibles d'être biaisées (comme les préférences des candidats d'ailleurs) ; la reproduction de tels biais par l'intermédiaire des systèmes de recommandation constitue potentiellement un des dommages majeurs des algorithmes de science des données du point de vue sociétal (O'Neil, 2017).

Une autre perspective de recherche algorithmique concerne la prise en compte des intérêts multiples des différents utilisateurs. Dans l'approche actuelle, le fait qu'un utilisateur ait pu cliquer sur deux offres de métiers différents suffit à créer un lien entre ces deux offres, et par transitivité entre ces deux métiers. La prise en compte de la diversité des intérêts permettrait de ne pas créer des liens hasardeux, de créer des représentations des offres et des métiers plus nettement différenciées, et d'accommoder les préférences multiples au niveau des utilisateurs au lieu de la prendre en compte au niveau plus général de la représentation.

Une autre perspective de recherche, à la fois algorithmique et applicative, concerne la modélisation causale de l'emploi et l'usage du raisonnement contre-factuel pour examiner la question de la formation. Plus précisément, la question est de savoir comment les offres d'emploi qui pourraient être proposées à une personne donnée seraient modifiées si cette personne acquérait des compétences supplémentaires (What if ?). Le fait de savoir répondre à cette question permettrait par induction de passer de la recommandation d'offres d'emploi à la recommandation de formations. La recommandation de formations − d'un point de vue personnalisé et avec les connaissances factuelles des besoins des entreprises sur un bassin d'emploi − définit un prochain objectif prioritaire, tant scientifiquement que dans le souci des impacts de l'apprentissage sur la société. Certains objectifs auxiliaires, tels que la cartographie des demandes des entreprises, géographiquement et par secteur, et leur évolution dans le temps, pourraient être des informations précieuses tant pour les

`candidat/les-fiches-metiers-@/index.jspz?id=681` et dont se sert Pôle-emploi pour orienter les demandeurs d'emploi.

choix d'orientation en fonction des objectifs individuels, que pour les choix des politiques publiques. Dans cet ordre d'idée, les données disponibles pourraient permettre de reconsidérer les ontologies des métiers, de les mettre à jour, en notant par exemple que certains métiers sont assurés par les mêmes personnes en fonction de la demande (e.g., *graphic designer* et *programmeur de jeux vidéos*), ou que certaines rubriques de l'ontologie sont de granularité trop insuffisante.

Enfin, une perspective applicative et algorithmique concerne le déploiement des algorithmes sur plate-forme réelle en vraie grandeur. Les défis sont de deux natures. Ils concernent d'abord le passage à l'échelle des algorithmes d'apprentissage et l'exploitation en ligne des modèles appris. En second lieu, l'algorithme d'appariement lui-même peut modifier les usages des utilisateurs. Par exemple, le fait que les CVs puissent être exploités automatiquement devrait permettre de relaxer les contraintes sur leur longueur, et le niveau de sophistication requis pour leur permettre de passer les filtres des directions des ressources humaines.

Annexe A

Collaborative Local Embedding (CLE) : Distorsion locale de la représentation

Dans cette annexe, nous proposons une méthode permettant d'améliorer la représentation des textes en tenant compte de la structure locale des données collaboratives. Nous nous inspirons de la méthode de dimension de réduction Localy Linear Embedding (LLE) (Roweis and Saul, 2000) permettant de respecter la structure locale des plus proches voisins.

L'analyse des différences entre les représentations des objets, l'une issue du texte, l'autre représentation issue des données collaboratives nous conduit à ajuster la représentation du texte. Ainsi, l'idée directrice est d'apprendre une modification de la représentation des documents, permettant de travailler à froid, pour *in fine* améliorer la recommandation à base de mémoire.

Nous présentons dans un premier temps la méthode de réduction de dimension Locally Linear Embedding (LLE) (Section A.1). Puis nous présentons un critère analytique permettant d'évaluer la qualité d'une représentation, étant donné une représentation oracle. Nous introduisons ainsi la notion de distorsion (Section A.2), directement inspirée de la fonction de perte de LLE. Cette distorsion est utilisée pour optimiser une représentation du langage dans le cadre d'un système de recommandation, formalisant ainsi le modèle Collaborative Local Embedding (CLE) (Schmitt et al., 2016) (Section A.3).

A.1 Locally Linear Embedding

LLE est une méthode de réduction de dimension non supervisée, introduite par Roweis and Saul (2000), détaillée dans (Saul and Roweis, 2003; Chen and Liu, 2011). Cette méthode préserve la structure *locale* des données, conservant les relations linéaires de voisinage des plus proches voisins. Cette méthode a l'avantage d'être invariante par homothétie, translation et rotation de l'espace initial. La fonction de perte de cette méthode inspire l'approche CLE présentée dans la section suivante.

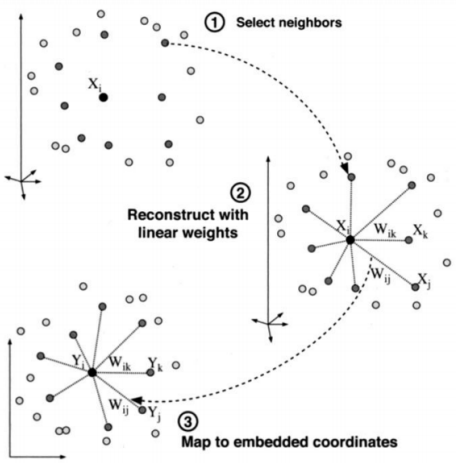


FIGURE A.1 – À partir de l'ensemble des points de départ X (image 1) nous calculons la matrice $\mathcal{W}$ de sorte que chaque point $\boldsymbol{x_i}$ soit combinaison linéaire de ses voisins (Équation A.1) (image 2). Puis nous définissons de nouveaux points Y respectant la structure locale définie par $\mathcal{W}$ (Équation A.2) (image 3). Figure extraite de (Roweis and Saul, 2000)

Calcul des voisinages $\mathcal{W}$ Formellement, soit Z un ensemble de points en grande dimension $Z = (\boldsymbol{z_1}, \ldots, \boldsymbol{z_n}), \boldsymbol{z_i} \in \mathbb{R}^D$. A chaque point $\boldsymbol{z_i}$ est associé l'ensemble de ses κ plus proches voisins, où κ est un hyper-paramètre de l'algorithme, et on détermine la combinaison linéaire de ces voisins approchant au mieux $\boldsymbol{z_i}$. La matrice $\mathcal{W} \in \mathbb{R}^{n \times n}$ est définie par :

$$\mathcal{W} = \arg\min_{W} |\sum_{i=1}^{n} \boldsymbol{z_i} - \sum_{j} W_{i,j} \boldsymbol{z_j}|^2 = \arg\min_{W} \|Z - WZ\|_2^2 \tag{A.1}$$

sous deux contraintes :

i) Les poids $W_{i,j}$ sont nuls pour les points n'appartenant pas aux κ plus proches voisins de $\boldsymbol{x_i}$;

ii) La somme sur i des $W_{i,j}$ est égale à 1.

La contrainte i) permet de ne considérer que les voisinages locaux des points, et garantit l'invariance par rotation et par homothétie. La contrainte ii) garantit l'unicité de $\mathcal{W}$.

En résumé,

$$X \approx \mathcal{W}X$$

$\mathcal{W}$ caractérise ainsi la métrique locale des points de Z. Chaque point i est ainsi approximé par le barycentre de ses κ plus proches voisins, pondérés par les valeurs $\mathcal{W}_{i,.}$.

Calcul de la représentation Y Une fois $\mathcal{W}$ déterminé, l'image $Y = (\boldsymbol{y_1}, \ldots \boldsymbol{y_n}), \boldsymbol{y_i} \in \mathbb{R}^d$ des points de X est obtenue en imposant que les $\boldsymbol{y_i}$ vérifient les mêmes propriétés locales que les $\boldsymbol{x_i}$:

$$Y = \arg\min_{\mathcal{Y}} \|\mathcal{Y} - W \cdot \mathcal{Y}\|_2^2 \tag{A.2}$$

sous deux contraintes qui garantissent l'unicité de la solution :

i) $\sum_i \boldsymbol{y_i} = 0$ (le barycentre des points est en zéro) ;
ii) $\frac{1}{N}\sum_i \boldsymbol{y_i} \cdot \boldsymbol{y_i}^T = I_d$ (la covariance des $\boldsymbol{y_i}$ est fixée à la matrice identité).

Les avantages de LLE sont de faire intervenir un seul hyper-paramètre κ, et de définir un problème d'optimisation convexe, garantissant l'existence et l'unicité de la solution.

Les **limitations** de cette méthode sont les suivantes :
- sensibilité au bruit et aux points dégénérés (outliers) (Chang and Yeung, 2006) ;
- sensibilité à la valeur de κ ;
- l'hypothèse sous-jacente est que les données habitent une même variété et sont connexes, ce qui n'est pas nécessairement le cas pour des données à plusieurs classes par exemple.
- enfin, le fait de calculer l'image de nouveaux points x_i demande de recalculer l'ensemble des points y_i.

A.2 Distorsion

Cette partie formalise le concept sous-jacent à LLE, que nous nommons *distorsion*. Cela nous permet d'avoir un critère analytique permettant de définir la qualité d'une représentation vis-à-vis d'une métrique oracle. Dans ce contexte et pour la suite, nous considérons la métrique issue des données collaboratives comme métrique oracle. La représentation oracle de chaque objet est donnée par $Z = \mathcal{R}^T$, où $\mathcal{R}^T = (\boldsymbol{\mathcal{R}_{.,1}}, \ldots, \boldsymbol{\mathcal{R}_{.,n}}), \boldsymbol{\mathcal{R}_{.,i}} \in \{0,1\}^{|\mathcal{U}|}$ code la représentation de chaque objet dans l'espace des utilisateurs.

Nous avons vu (Équation A.1) une manière de calculer une relation de voisinage local, résumé dans une matrice noté $\mathcal{W}$, rappelée ici :

$$\mathcal{W} = \arg\min_W |\sum_{i=1}^{n} \boldsymbol{\mathcal{R}_{.,i}} - \sum_j W_{i,j}\boldsymbol{\mathcal{R}_{.,j}}|^2 = \arg\min_W \left\|\mathcal{R}^T - W\mathcal{R}^T\right\|_2^2$$

Sous les mêmes contraintes que (Équation A.1), à savoir que le nombre de valeurs non nulles par ligne de $\mathcal{W}$ est au maximum de κ, conservant ainsi seulement les κ plus proches voisins (propriétés locales) et la somme des lignes de $\mathcal{W}$ est de 1, ce qui permet de fixer l'échelle des valeurs servant à calculer les barycentres.

$$\forall i,\ \sum_j \mathcal{W}_{i,j} = 1 \text{ et } \sum_j \mathbb{1}_{\{\mathcal{W}_{i,j}\neq 0\}} = \kappa$$

Ainsi, la représentation Y issue de l'algorithme de LLE correspond à $Y = \arg\min_{\mathcal{Y}} \|\mathcal{Y} - \mathcal{W}\mathcal{Y}\|_2^2$, avec les contraintes associées à l'équation A.2.

Soit $\Delta(Y)$ défini par

$$\begin{aligned}\Delta(Y) &= \sum_{i=1}^{n} \left\| (\boldsymbol{y_i} - \sum_j \mathcal{W}_{i,j}\boldsymbol{y_j}) \right\|_2^2 \\ &= \|Y - \mathcal{W}Y\|_2^2\end{aligned}$$

Nous appelons erreur de distorsion pour une représentation Y la quantité $\Delta(Y)$, relativement à la métrique oracle encodé par $\mathcal{W}$.

Ainsi cette distorsion estime comment la structure locale de Y diffère par rapport à la structure locale de la représentation oracle.

La distorsion a l'avantage d'être calculable analytiquement (ce qui permet à l'algorithme LLE d'être résolue sous forme fermée). Contrairement à la fiabilité@κ (Équation 2.3) qui impose un calcul de rang (donc de plus proches voisins) ou à la mesure issue de κ-nn (Section 2.4) qui nécessite aussi le calcul de plus proches voisins ainsi que l'utilisation de classe, la formule de la distorsion est calculable en temps linéaire par rapport au nombre de points dans l'ensemble Y.

Une autre propriété intéressante de cette distorsion est le fait que nous pouvons définir une erreur de distorsion δ^j pour chaque dimension j, moyenné sur l'ensemble des points $\boldsymbol{y_i}$:

$$\delta^j(Y) = \sum_{i=1}^{n} (\boldsymbol{y_i^j} - \sum_{l=1}^{n} \mathcal{W}_{i,l} \cdot \boldsymbol{y_l^j})^2 \tag{A.3}$$

L'erreur de distorsion par dimension nous indique comment chaque dimension contribue à l'erreur générale de distorsion.

A.3 Modèle CLE

Notre objectif est d'apprendre une fonction paramétrique ψ qui sera utilisé pour induire une similarité entre documents, permettant une recommandation à base de mémoire orientée objet. La fonction ψ part de la représentation du texte $\mathbb{R}^D$ (seule donnée disponible pour le cas de démarrage à froid) vers un espace latent image $\mathbb{R}^d$, mimant la topologie induite par $\mathcal{R}^T$ (décrite par $\mathcal{W}$), en minimisant l'erreur de distorsion. Ce modèle est nommé Collaborative Local Embedding (CLE) en référence à LLE. d est un hyper paramètre du modèle, de même que dans LSA, spécifiant la dimension de l'espace représentant les objets. Remarquons que si LSA est une représentation du texte aveugle aux données collaborative, nous voulons apprendre une représentation du texte qui prend en compte les données collaboratives.

En notant $\psi(X) = (\psi(\boldsymbol{x_1}), \ldots, \psi(\boldsymbol{x_n}))$ l'ensemble image de la représentation X, nous pouvons définir la distorsion de ce nouvel ensemble par

$$\Delta(\psi(X)) = \sum_{i \in \mathcal{O}} \left\| \psi(\boldsymbol{x_i}) - \sum_{j} \mathcal{W}_{i,j} \cdot \psi(\boldsymbol{x_j}) \right\|_2^2 \tag{A.4}$$

ainsi que la variance de cet ensemble de point $Var(\psi(X)) = \sum_i \|\psi(\boldsymbol{x_i}) - \boldsymbol{m}\|_2$, avec $\boldsymbol{m} = \frac{1}{|\mathcal{O}|} \sum_{i \in \mathcal{O}} \psi(\boldsymbol{x_i})$ la moyenne de $\psi(X)$.

La matrice $\mathcal{W}$ est calculée sur l'ensemble oracle $\mathcal{R}^T$ (Équation A.1). Une première étape consiste à prendre la fonction ψ^* minimisant l'erreur de distorsion (Équation A.4). Mais cela conduit à une représentation triviale, projetant tous les objets sur un seul point de l'espace $\mathbb{R}^d$. En effet, la fonction $\psi = 0$ minimise l'erreur de distorsion. Dans le cas de LLE, les contraintes sur la covariance de la nouvelle représentation empêchent une telle représentation triviale (Équation A.2). En effet, la dispersion des points est imposée par le fait que chaque dimension est de variance 1. Cependant, dans notre approche paramétrique, nous ne pouvons pas forcer les contraintes sur Y à chaque itération de la descente de gradient. Nous ajoutons donc la contrainte de la variance à travers un

terme de régularisation Reg :

$$L(\psi) = \sum_{i \in \mathcal{O}} \left\| \psi(\boldsymbol{x_i}) - \sum_j \mathcal{W}_{i,j} \cdot \psi(\boldsymbol{x_j}) \right\|_2^2 - \lambda Reg(\psi) \tag{A.5}$$

Ici, le terme de régularisation est fixé à la variance [1] de $\psi(X)$: $Reg(\psi) = Var(\psi(X))$. La force du terme de régularisation est contrôlée par l'hyper-paramètre λ. Si $\lambda \to \infty$, la descente de gradient a tendance à disperser les points dans l'espace, inversement, si $\lambda \to 0$, tous les points sont concentrés en un seul point image.

Le paramètre κ fixe le nombre de voisins à considérer lors de la création de $\mathcal{W}$, qui code la structure de la métrique oracle. Si $\kappa = 1$, les voisins autres que le plus proche (dans la métrique oracle) ne sont pas pris en compte dans la fonction de perte, nous perdons ainsi de l'information. Cependant, si κ est trop grand, chaque point est représenté par un voisinage qui n'est plus local, les paramètres de $\mathcal{W}$ peuvent être biaisé, car certains voisins proches peuvent avoir un poids très faible.

Fonction ψ La fonction paramétrique ψ est apprise à l'aide d'un réseau de neurones. Cette fonction fournissant une nouvelle représentation des offres est au coeur de la méthode de recommandation basée sur la mémoire. L'implémentation est réalisée à l'aide des librairies de deep learning Theano (Theano Development Team, 2016) et Lasagne (Dieleman et al., 2015). La fonction de perte L est optimisée par descente de gradient stochastique et la régularisation est calculée sur chaque mini-bloc. L'initialisation des poids suit les recommandations données par Xavier Glorot (Glorot and Bengio, 2010b) excepté pour la première couche. En effet, l'entrée tf-idf est d'abord projetée dans un espace à 300 dimensions, en utilisant l'initialisation correspondant à LSA [2] (Figure A.2). Notons que les poids de la première couche peuvent être figés lors de l'apprentissage, cela revient alors à considérer la représentation LSA des documents en entrée du réseau.

A.3.1 Résultats

Configuration expérimentale Les hyper-paramètres choisis pour entraîner le réseau de neurones sont résumés (Table A.1)

- La valeur du paramètre κ, évalué par validation croisé pour des valeurs comprises entre 1 et 100 est fixé à 10. En effet, au-delà de 10, les performances se dégradent progressivement.
- L'*entrée* est constituée des poids issus de tf-idf. Notons que la représentation LSA en entrée obtient des performances proches, de l'ordre d'un point inférieur, réduisant le nombre de paramètres d'un facteur 30 [3].
- d est fixé à 300 pour être comparable à LSA ;
- La non-linéarité *tanh* est plus stable que *relu* lors de l'optimisation des autres hyper-paramètres ;

1. Notons que le centre de l'ensemble des points n'est pas régularisé. En effet, cette statistique n'intervient pas lors du calcul de recommandation (Équation 3.6), où la similarité est calculée entre *paires* de points, donc invariante par translation de l'ensemble des points.

2. Les poids de la première couche sont initialisé aux valeurs V^T de la SVD (Équation 2.1). Ces poids sont ensuite modifiés durant l'entraînement.

3. Cette contrainte est considérée, rendant le modèle plus léger et plus rapide à entraîner (ayant 180 600 paramètres libres plutôt que 3 180 900), cependant les scores obtenus sont inférieurs.

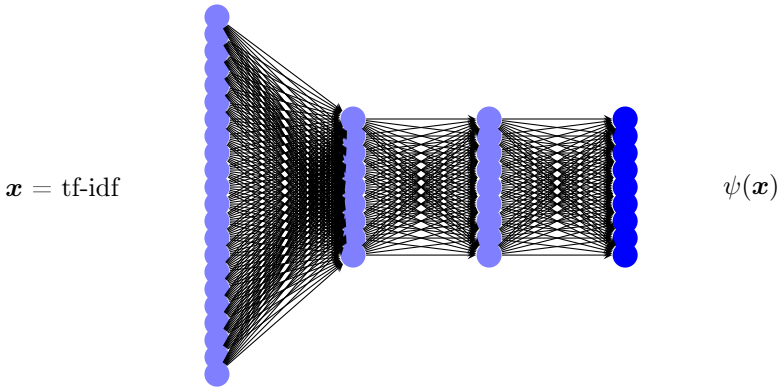


FIGURE A.2 – Le réseau ψ dispose de deux couches cachées avec la fonction d'activation *tanh*. La sortie de la dernière couche est linéaire, cela donne la représentation ψ

Hyper paramètre	valeur
κ	10
Taille du vocabulaire	10 000
Dimension des couches cachées	300
Nombre de couches cachées	2
Nonlinéarité	*tanh*
λ	1
Optimisation	descente de gradient stochastique (SGD)
Taille des blocs	100
Initialisation	LSA et Glorot

TABLE A.1 – liste des hyper-paramètres utilisés pour entraîner le réseau CLE.

— le terme d'équilibrage entre la distorsion et la régularisation λ est fixé à 1.
— L'optimisation utilise la descente de gradient stochastique (SGD) ;
— La taille des blocs, fixée à 100, n'est pas un paramètre sensible ;
— L'initialisation de la première couche aux valeurs LSA (Section 2.2.3) permet au réseau de converger plus rapidement et d'obtenir de meilleures performances.

Résultats Nous évaluons notre modèle sur les données de Qapa dans le cas du démarrage à froid objet (décrit Section 5.2.1) sur 10 validation croisées (Table A.2). Nous obtenons des résultats significativement meilleurs (3%) que l'approche de référence $\text{LSA}_{\frac{1}{2}}$ ainsi que LSA_{δ}. La nouvelle représentation du texte, ajustée en fonction des données permet ainsi une meilleure recommandation

	$\text{LSA}_{\frac{1}{2}}$	CLE
Rappel @ 100	.694 ±.003	**.720 ± .004**

TABLE A.2 – Rappel pour LSA et CLE en démarrage à froid. CLE améliore nettement le rappel par rapport à LSA.

lorsque nous recommandons "plus du même". Ces résultats sont obtenus au bout de 10 époques, en 40 minutes sur un Intel Core i7, 2.10GHz x 4.

Discussion Cependant, cette approche ne considère que les κ plus proches voisins dans $\mathcal{W}$, cela ne prend donc pas en compte tous les voisins positifs (augmenter le nombre de voisins pris en compte au delà de 10 n'augmente pas les performances). Une explication à ce phénomène est la façon dont nous calculons les valeurs de $\mathcal{W}$. En effet, l'évolution des termes intervenant dans ce calcul de barycentre, minimisant la fonction (Équation A.1), n'est pas contrôlée.

CLE souffre également des mêmes limitations que LLE, à savoir :

— Sensible au bruit et aux points dégénérés (Chang and Yeung, 2006) ;

— CLE, comme LLE, suppose que les données résident sur *une seule* variété continue.

De plus, la variance comme régularisation augmente le coût de calcul. Pour remédier à ces limitations, nous considérons les réseaux siamois décrits dans le chapitre 6.

Annexe B

Différence d'orientation entre objet et utilisateur

Prenons un cas extrême pour apprécier les différences d'orientation utilisateur et objet. Partons de la matrice (ou éventuellement la sous matrice) collaborative donnée (Table B.1). Calculons $\mathcal{R}_{ui}$ et $\mathcal{R}_{uj}$ à l'aide des deux orientations :

— *orienté objet* : Nous avons, pour tout i' appartenant à $[i_1, \ldots, i_a]$:

$$sim(j, i') = sim(j, i_1) = \frac{1}{\sqrt{2} \cdot \sqrt{b+1}}$$
$$sim(i, k) = \frac{b}{\sqrt{b} \cdot \sqrt{b+1}}$$
$$sim(j, i) = 0$$

Ainsi :

$$\hat{r}_{ui} = sim(i, k) = \frac{b}{\sqrt{b} \cdot \sqrt{b+1}}$$

	i_1	..	i_a	k	i	j
u_1				1	1	
..				..	..	
u_b				1	1	
u	1	..	1	**1**	$\frac{b}{\sqrt{b}\sqrt{b+1}}$	$\frac{a}{\sqrt{2}\sqrt{b+1}}$
v	1	..	1		**0**	**1**

(a) orienté objet. Le signe de $r_{ui} - r_{uj}$ dépend de a et b

	i_1	..	i_a	k	i	j
u_1				1	1	
..				..	..	
u_b				1	1	
u	1	..	1	**1**	$\frac{b}{\sqrt{2}\sqrt{a+1}}$	$\frac{a}{a+1}$
v	1	..	1		**0**	**1**

(b) orienté utilisateur. $\forall(a, b)\ r_{ui} > r_{uj}$

TABLE B.1 – filtrage collaboratif basé sur la mémoire : Différence *orienté objet* contre *orienté utilisateur*

	i_1	i_2	i_3	k	i	j
u_1				1	1	
u_2				1	1	
u_3				1	1	
u	1	1	1	**1**	$\frac{3}{\sqrt{12}}$	$\frac{3}{\sqrt{8}}$
v	1	1	1		**0**	**1**

(a) orienté objet.

	i_1	i_2	i_3	k	i	j
u_1				1	1	
u_2				1	1	
u_3				1	1	
u	1	1	1	**1**	$\frac{3}{\sqrt{8}}$	$\frac{3}{4}$
v	1	1	1		**0**	**1**

(b) orienté utilisateur.

TABLE B.2 – a=3, b=3. Dans le cas orienté objet, nous recommanderons à l'utilisateur u d'abord l'objet j avant i. Inversement dans le cas orienté utilisateur.

	i_1	i_2	k	i	j
u_1			1	1	
u	1	1	**1**	$\frac{1}{2}$	$\frac{1}{2}$
v	1	1		**0**	**1**

(a)

	i_1	i_2	k	i	j
u_1			1	1	
u	1	1	**1**	$\frac{1}{2}$	$\frac{1}{2}$
v	1	1		**0**	**1**

(b)

TABLE B.3 – a = 1 et b=1

et

$$\hat{r}_{uj} = \sum_{i' \in [i_1, \ldots, i_a]} sim(j, i') = \frac{a}{\sqrt{2}\sqrt{b+1}}$$

— *orienté utilisateur* : Nous avons, pour tout u' appartenant à $[u_1, \ldots, u_b]$:

$$sim(u, u') = sim(u, u_1) = \frac{1}{\sqrt{2} \cdot \sqrt{a+1}}$$
$$sim(u, v) = \frac{a}{\sqrt{a+1} \cdot \sqrt{a+1}} = \frac{a}{a+1}$$

Ainsi :

$$\hat{r}_{ui} = \sum_{u' \in [u_1, \ldots, u_b]} sim(u, u_1) = b \times sim(u, u') = \frac{b}{\sqrt{2} \cdot \sqrt{a+1}}$$

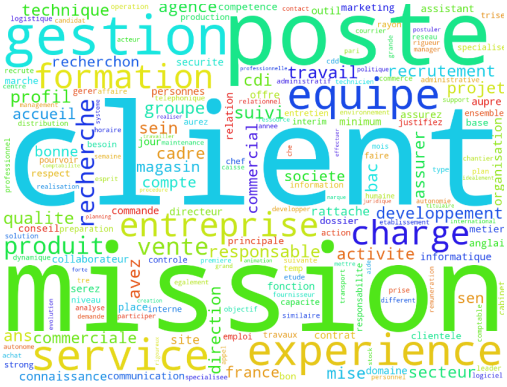
et

$$\hat{r}_{uj} = \frac{a}{a+1}$$

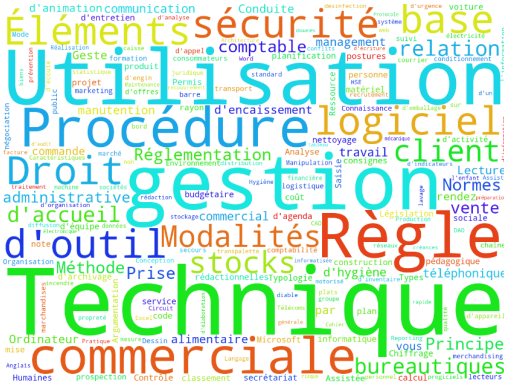
Notons que si nous utilisons une similarité sans normalisation, comptant ainsi le nombre de cliques en communs (aussi appelé similarité de Jaccard), alors l'approche de filtrage collaborative à base de mémoire est équivalent orienté utilisateur et orienté objet. Dans les deux cas, nous avons $\widehat{\mathcal{R}} = \mathcal{R} \cdot \mathcal{R}^T \cdot \mathcal{R}$. Ces différences sont étudiées et généralisées dans (Verstrepen and Goethals, 2014).

Annexe C

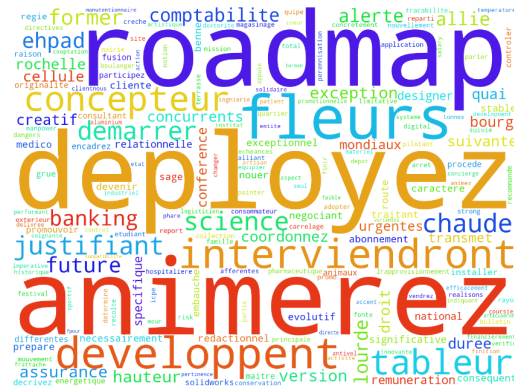
Données Qapa



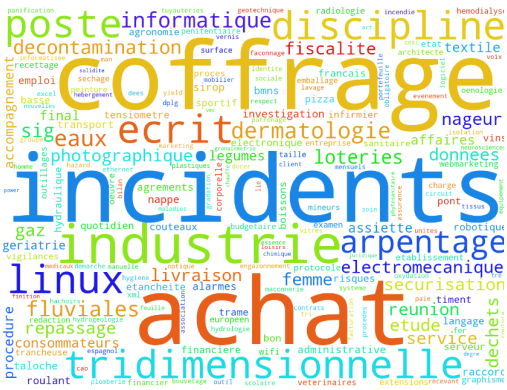
(a) Mots (tf) apparaissant dans les *descriptions* des offres d'emploi.



(b) Mots (tf) apparaissant dans les listes de *compétences* des offres d'emploi.



(c) Mots *clefs* (tf-idf) apparaissant dans les *descriptions* des offres d'emploi.



(d) Mots *clefs* (tf-idf) apparaissant dans les listes de *compétences* des offres d'emploi.

FIGURE C.1 – Nuage de mots pour les *offres* chez Qapa. La police est proportionnelle à la fréquence (haut) ou au tf-idf (bas). À gauche sont représentés les mots de la description des offres, à droite les mots des compétences associées aux offres.

ID de l'offre	métiers	Compétences	Description
2	Serveur / Serveuse en restauration - Blois	Principes de la relation client, Anglais, Modalités d'accueil, Pratique de discipline sportive	Nous recrutons pour 20 postes de serveurs en restaurant (H/F) situés dans la région de Blois.CDD plein temps de 6 mois (2 mars au 11 septembre) en contrat de professionnalisation : alternance associant une formation théorique et une période de professionnalisation en entreprise. Le but du contrat de porfessionnalisation est de faciliter la qualification, l'intégration professionnelle et l'employabilité. [...]
10	Réceptionniste en hôtellerie - Strasbourg	Principes de la relation client, Anglais, Modalités d'accueil, Utilisation de logiciel de réservation, Règles de sécurité des biens et des personnes, Planning d'occupation des chambres, Utilisation de logiciel de facturation hôtelière, Environnement culturel et touristique, Stratégie commerciale, Modes de paiement, Allemand	Pour établissement 3* - 120 chambresSecteur : Département 67 - 68 Contrat 39h Compétences : Allemand – Anglais (parlé – écrit)Connaissance logiciel Opéra – MicrosPolyvalence restauration / bar / petit déjeuner
52	Chef de chantier - Fleury-Mérogis	Chiffrage/calcul de coût, Principes de la relation client, Économie de la construction, Techniques de construction, Techniques d'animation d'équipe, Règles et consignes de sécurité, Méthodes d'approvisionnement, Normes de la construction, Gestion administrative du personnel, Lecture de plan, de schéma, Procédures de contrôle des matériels et équipements	"Chef de Chantier GO (H/F) - CDI ((H/F)) - LTD International recherche pour le compte de ses clients un chef de chantier GO en bâtiment, dans le cadre du développement de son activité. Entreprise basée dans l'Essonne, 20M euros de CA, principalement du logement. Vos missions seront les suivantes : - Préparer les travaux en participant à la mise au point du projet d'exécution, - Participer à la définition des ressources, - Participer à l'élaboration des plans nécessaires au démarrage du chantier
27558	Désamianteur / Désamian-teuse - Équeurdreville-Hainneville	Permis B (voiture), Réglementation en Hygiène, Sécurité, Environnement -HSE-, Montage d'échafaudage, Utilisation de matériel de contrôle et mesure, Méthodes d'assainissement radioactif, chimique et bactériologique, Radioprotection, Techniques de confinement, Techniques de calorifugeage, Techniques de démantèlement, Normes de désamiantage	vous êtes chef de chantier désamiantage. vous possédez impérativement la formation "amiante encadrement de chantier" à jour.Vous intervenez sur la préparation et l'encadrement des chantiers ; principalement dans un rayon de 400km.Prise de poste immédiate.
87	Responsable de rayon produits non alimentaires - Centre	Utilisation de logiciels de gestion de stocks, Règles de gestion de stocks, Techniques commerciales, Techniques de management, Éléments de base en gestion des Ressources Humaines, Utilisation de lecteurs de codes-barres, Techniques de merchandising, Techniques d'animation d'équipe, Éléments de base en statistiques, Lecture de tableau de bord et d'indicateurs de suivi d'activité, Utilisation d'outils bureautiques, Technique de négociation commerciale, Sens du service client, Pratique du tir sportif	Travailler chez DECATHLON : C'est rejoindre un groupe international qui compte plus de 50.000 collaborateurs dans 18 pays, leader européen de la conception, de la fabrication et de la distribution d'articles de sport. C'est rejoindre des passionnés de sport dans une entreprise où convivialité et professionnalisme se concilient à merveille. [...] POSTE ET MISSION : Au sein d'un magasin, responsable de l'activité commerciale de votre rayon, vous prenez en charge le recrutement, la formation et l'animation de votre équipe. Passionné par le client, vous élaborez et animez votre projet commercial grâce à une bonne gestion de vos stocks, de vos linéaires et de vos gammes de produits. [...]

Table C.1 – Exemple d'offres d'emploi sur Qapa.fr

Thème LDA	Mots clefs	Interprétation du thème
0 :	experience, poste, formation, etes, avez, profil, ans, bac, recrutement, equipe, entreprise, recherche, minimum, missions, sens, justifiez, developpement, client, charge, cabinet	Expérience
1 :	entretien, travail, jours, poste, service, domicile, restaurant, aide, semaine, charge, horaires, nettoyage, pourvoir, serez, samedi, dimanche, sein, temps, place	Entretien
2 :	clients, commercial, vente, produits, developpement, secteur, client, france, aupres, developper, groupe, clientele, entreprise, affaires, ventes, services, objectifs, portefeuille, commerciaux	Commercial
3 :	poste, experience, accueil, cdi, emploi, recherche, charge, recherchons, clients, cdd, etes, interim, lundi, devez, temps, client, commandes, serez, pourvoir, travail	Accueil
4 :	securite, qualite, maintenance, production, travail, technicien, groupe, recrutement, interim, responsable, clients, industrie, cdi, france, temporaire, technique, conducteur, informatique, recherche, equipements	Sécurité
5 :	social, droit, rh, vie, travail, humaines, ressources, personnes, sante, groupe, juridique, collaborateurs, equipe, enfants, activites, paie, personnel, sociale, service, accompagnement	Ressource humaine
6 :	magasin, vente, produits, rayon, client, equipe, service, etes, clients, caisse, enseigne, distribution, sens, accueil, hote, mise, avez, clientele, encaissement, responsable	Magasinier
7 :	communication, marketing, developpement, web, projets, mise, strategie, creation, projet, groupe, outils, actions, site, place, charge, sites, business, internet, chef, marche	Developpement web
8 :	gestion, suivi, direction, missions, assistant, clients, assurer, responsable, organisation, dossiers, charge, directeur, service, client, mise, recherche, gerer, rattache, comptable, principales	Gestion
9 :	travaux, etudes, projets, projet, technique, chantiers, batiment, techniques, realisation, chantier, construction, charge, client, conception, plans, chef, publics, bureau, ingenieur, execution	Bâtiment / Chantier

TABLE C.2 – Thèmes découverts par LDA sur les données de Qapa. Les mots clefs correspondent aux mots ayant le plus de probabilité d'apparaître dans le thème. L'interprétation est subjective, à partir des mots clefs.

ID	metier	θ	Thèmes présents
2	Serveur	[0.28 0.3 0. 0.41 0. 0. 0. 0. 0. 0.]	3, 1, 0 : **Accueil**, Entretien, Expérience
10	Réceptionniste	[0.01 0.39 0.01 0.17 0.01 0.01 0.01 0.01 0.4 0.01]	8, 1, 3 : **Gestion**, **Entretien**, Accueil
52	Chef de chantier	[0. 0. 0. 0. 0.24 0. 0.15 0.2 0. 0.4]	9, 4, 7 : **Bâtiment**, Sécurité, Développement web
27558	Désamianteur	[0.26 0. 0. 0.24 0.06 0. 0.07 0. 0. 0.35]	9, 0, 3 : **Bâtiment**, Expérience, Accueil
87	Responsable de rayons	[0.19 0.17 0.18 0. 0. 0. 0.38 0.07 0. 0.]	6, 0, 2 : **Magasinier**, Expérience, Commercial

TABLE C.3 – Distribution des exemples d'offres Qapa (Table C.1) sur les sujets découverts par la LDA (Table C.2). Pour chacun nous donnons sa distribution dans l'espace des sujets (de dimension 10). La colonne de droite est l'interprétation des trois principales coordonnées de θ à l'aide de (Table C.2), en gras les sujets représentant plus de 33% du document.

Thème LSA	Mots clefs	Interprétation
0 :	gestion, clients, experience, poste, etes, missions, client, formation, equipe, avez, entreprise, suivi, recherche, charge, responsable, service, developpement, groupe, produits, profil	?
1 :	gestion, suivi, direction, assistant, dossiers, missions, organisation, administrative, comptabilite, poste, comptable, administratif, controle, preparation, accueil, reunions, documents, ressources, redaction, assistanat	comptable
2 :	communication, marketing, developpement, responsable, projets, entreprise, projet, equipe, directeur, actions, strategie, groupe, qualite, equipes, direction, web, place, production, outils, mise	projet
3 :	gestion, produits, magasin, vente, rayon, responsable, qualite, equipe, service, mise, distribution, carrefour, suivi, enseigne, assurer, stocks, respect, controle, employe, caisse	vente/magasin
4 :	clients, commercial, suivi, marketing, gestion, commerciale, services, recrutement, developpement, communication, portefeuille, aupres, france, solutions, comptes, assistant, marche, developper, besoins, offres	commerciale
5 :	qualite, securite, service, client, production, groupe, assurer, travail, travaux, cdi, carrefour, caisse, maintenance, projet, projets, competences, hote, emploi, respect, operations	securité
6 :	communication, marketing, accueil, caisse, hote, groupe, carrefour, poste, rayon, produits, organisation, direction, sens, anglais, cdi, operations, sse, mois, web, distribution	acceuil
7 :	recrutement, magasin, groupe, developpement, rh, vente, gestion, france, collaborateurs, humaines, ressources, travail, equipe, candidats, responsable, cdi, entreprise, conseil, missions, formation	RH
8 :	communication, travail, marketing, produits, charge, poste, mise, accueil, clients, magasin, serez, responsable, assurer, cdi, emploi, temps, commandes, interim, agence, securite	acceuil
9 :	client, etes, charge, communication, marketing, recrutement, responsable, interim, emploi, agence, recherchons, gestion, cdd, cdi, travaux, compte, randstad, annee, rejoignez, portes	RH ?

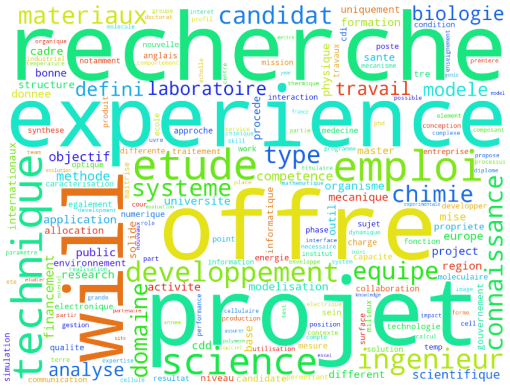
TABLE C.4 – Thèmes découverts par LSA sur les données de Qapa. L'interprétation, plus difficile que pour LDA, est subjective, à partir des mots clefs.

ID	metier	V	commentaire
2	Serveur	[1.94 -0.3 -1.15 -0.46 -0.66 0.26 0.46 0.32 0.13 0.68]	0,9,6 : ?, RH ?, Acceuil
10	Réceptionniste	[0.2 0.09 -0.08 -0.06 -0.07 0.08 0.14 -0.11 0.09 -0.1]	1 : ?
52	Chef de chantier	[1.66 -0.15 0.46 -0.38 0.63 0.8 -0.31 0. -0.25 0.14]	0, 5, 4 , 2 : ?, Sécurité, Commerciale, Projet
27558	Désamianteur	[0.81 -0.02 -0.2 -0.23 -0.76 -0.18 0.02 -0.35 -0. 0.01]	0 : ?
87	Responsable de rayons	[4.36 -1.15 0.32 1.2 -1.43 -0.4 0.06 1.83 -0.64 0.39]	0, 3,9,2 : ?, Vente/Magasin, RH ?, Projet

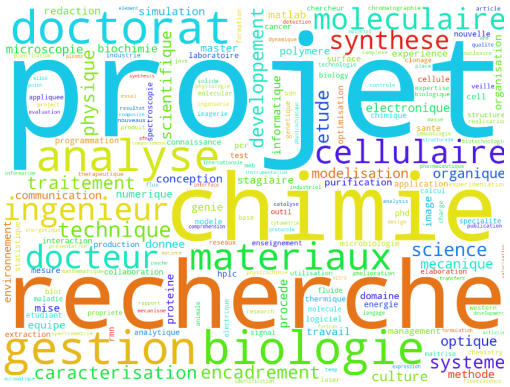
TABLE C.5 – Distribution des offres d'exemples Qapa (Table C.1) sur les sujets découvert par LSA (Table C.4)

Annexe D

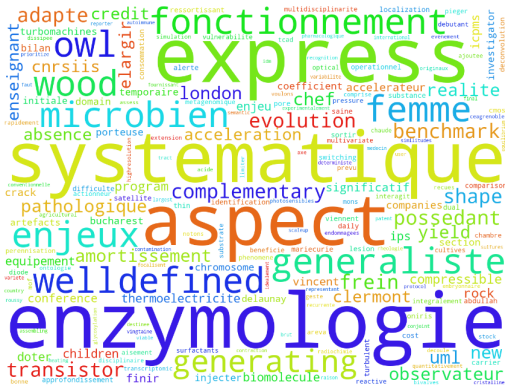
Données ABG



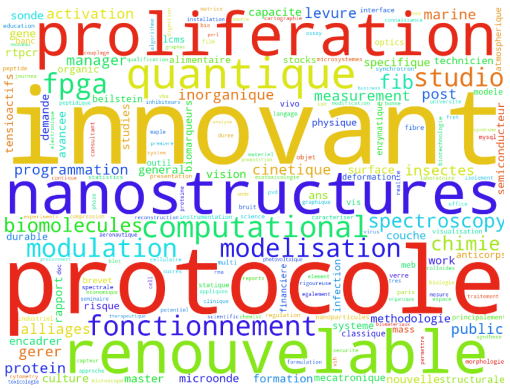
(a) Mots des Offres



(b) Mots des CVs



(c) Mots *clefs* des Offres



(d) Mots *clefs* des CVs

FIGURE D.1 – Nuage de mots des données ABG. Taille en fonction de la fréquence (en haut) et taille en fonction des poids tf-idf (en bas)

Champs	CV1	CV2
Titre du profil	Ingénieur R&D en Biologie et Biophysique	doctorat en physique specialité optique et photonique
Expertise Scientifique	Biologie cellulaire ; Microbiologie ; Microalgues ; Biochimie ; Transfert de gènes ; Apoptose ; Rayonnement non ionisant ; Champs électriques pulsés ; Plasma froid ; Décontamination bactérienne (eau, surface)	Optique non-linéaire, acousto-optique, électro-optique, magnéto-optique, électromagnétisme, physique de la matière condensée, traitement du signal, analyse spectrale, composants optiques et optoélectroniques, transmissions Faisceaux Hertziens.
Expertise Technique	Culture cellulaire ; Culture bactérienne ; Culture de tissus ; Microscopie ; Cytométrie en flux ; Transfection (in vitro, in vivo, ex vivo) ; Immunomarquage ; Electroporation de cellules	Utilisation d'appareils de mesures optiques, Travail avec des lasers de puissance, Programmation informatique et simulations numériques, Equipements télécoms Ericsson de transmission (MiniLink, OMS)
Domaine d'application des recherches	Champs électriques pulsés : décontamination, extraction ; Rayonnements non ionisants	Optique, Télécoms, Matériaux, Calculs scientifiques
Enjeux	Limiter l'utilisation de molécules chimiques en utilisant des méthodes physiques	Mise en place de dispositif télécoms tout-optique afin d'accroitre la rapidité de transfert des données.
Savoirs faire organisation-nels	Collaborations internationales (Etats-Unis, Israel, Allemagne...) ; Encadrement de stagiaires ; Gestion de stocks et de projets	Pilotage et coordination d'un projet scientifique et technique ; Collecte, trie, analyse et synthèse de l'information.
Intitulé du diplôme	Doctorat Biologie-Sciences-Santé	Z-scan technique : development and application to the nonlinear optical response study of nanocomposite materials

TABLE D.1 – Exemple de deux CVs issus des données ABG sur un sous ensemble des champs renseignées.

Champs	Offre 1	Offre 2
Type	Offre de thèse	Offre de thèse
Nature du contrat	Financement public uniquement (type allocation de recherche ; gouvernement, région, Europe, organismes internationaux. . .)	financement partiellement ou entièrement privé (Entreprise, fondation, association. . .)
Nom Etablissement recruteur	Université Paris Est Créteil Val de Marne	Institut de Radioprotection et de Sureté Nucléraire - Groupe
Domaine competence 1	Informatique, électronique	Chimie
Domaine competence 2	Mathématiques	Non défini
Profil des candidats	Compétences requises / Skills -Vision par ordinateur/ Computer vision, -Traitement d'image/ Image processing, -Programmation C/C++, Matlab/ C/C++ programming, Matlab, -Très bon niveau en mathématiques/ very good skills in mathematics, -Très bon niveau d'anglais/ Fluency in english. Conditions particulières/ particular conditions -Le candidat doit obligatoirement être inscrit en Master 2 (année universitaire 2012-2013), The application concerns only students with Master 2 in progress (2012-2013), -Excellents résultats en Master, Excellent exam grades are required.	Master en chimie-physique, chimie théorique, modélisation, cinétique chimique
Mission	La thèse proposée s'effectuera au sein du groupe biométrie. Elle s'inscrit dans le contexte de la reconnaissance faciale tridimensionnelle dynamique. Plus précisément, on s'intéresse à l'extraction, à partir d'un flux vidéo (Haute-Définition), des caractéristiques géométriques du visage, en vue d'une reconstruction 3D. L'approche à développer doit faire appel à un modèle mathématique intégrant des informations a priori. En utilisant une base de données de modèles de visages 3D, l'application peut être utilisée, entre autres, dans un contexte d'identification et/ou l'authentification d'individus. The proposed thesis will be supervised within the "Biometrics research group" where the subject deals with 3D face recognition in dynamic situations. [...]	Le travail de thèse proposé est de type modélisation, il concerne le développement d'un modèle de transport de chimie du ruthénium dans les conditions du circuit primaire d'un réacteur nécléaire en conditions oxydantes et dans la gamme de température 2000C-150C. La partie prépondérante de la thèse sera consacrée à l'obtention des propriétés thermocinétiques complexes à déterminer expérimentalement. Les quatre axes de recherche sont : - La détermination par chimie computationnelle des propriétés thermochimiques des oxydes et hydroxydes de ruthénium gazeux . - La détermination des constantes de vitesse pour des réactions impliquant les oxydes gazeux. - L'intégration d'un schéma réactionnel dans un code de simulation des accidents - La dernière partie porte sur l'interprétation et la simulation d'essais expérimentaux réalisés dans le cadre d'un programme OCDE.

TABLE D.2 – Exemple d'offres issus des données ABG sur un sous ensemble des champs renseignées.

Références

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow : A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283. *2 citations pages 24 et 92*

Abel, F., Deldjoo, Y., Elahi, M., and Kohlsdorf, D. (2017). Recsys challenge 2017 : Offline and online evaluation. *impressions*, 314 :88–719. *Cité page 55*

Aiolli, F. (2012). A preliminary study on a recommender system for the million songs dataset challenge. *Preference Learning : Problems and Applications in AI*, 1. *3 citations pages 44, 49, et 100*

Amatriain, X. (2014). Recommender systems. In *Machine Learning Summer School CMU*. *3 citations pages 37, 38, et 46*

Anderson, C. (2012). *La Longue Traîne*. Pearson. *Cité page 47*

AP, S. C., Lauly, S., Larochelle, H., Khapra, M., Ravindran, B., Raykar, V. C., and Saha, A. (2014). An autoencoder approach to learning bilingual word representations. In *Advances in Neural Information Processing Systems*, pages 1853–1861. *Cité page 25*

Bellet, A., Habrard, A., and Sebban, M. (2013). A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv :1306.6709*. *Cité page 83*

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb) :1137–1155. *2 citations pages 14 et 31*

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb) :281–305. *Cité page 23*

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan) :993–1022. *3 citations pages 14, 25, et 29*

Bordes, A., Chopra, S., and Weston, J. (2014). Question answering with subgraph embeddings. *arXiv preprint arXiv :1406.3676*. *Cité page 18*

Bourlard, H. and Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4) :291–294. *Cité page 20*

Bousquet, O., Gelly, S., Kurach, K., Teytaud, O., and Vincent, D. (2017). Critical hyper-parameters : No random, no cry. *arXiv preprint arXiv :1706.03200*. *Cité page 24*

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. *arXiv preprint arXiv :1508.05326.* *Cité page 48*

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. (1994). Signature verification using a" siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, pages 737–744. *7 citations pages 14, 15, 18, 82, 84, 85, et 105*

Bubeck, S., Cesa-Bianchi, N., et al. (2012). Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1) :1–122. *Cité page 46*

Burke, R. (2002). Hybrid recommender systems : Survey and experiments. *User modeling and user-adapted interaction*, 12(4) :331–370. *2 citations pages 49 et 53*

Büttcher, S., Clarke, C. L., and Cormack, G. V. (2016). *Information retrieval : Implementing and evaluating search engines*. Mit Press. *Cité page 48*

Caruana, R., Niculescu-Mizil, A., Crew, G., and Ksikes, A. (2004). Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on Machine learning*, page 18. ACM. *Cité page 49*

Chang, H. and Yeung, D.-Y. (2006). Robust locally linear embedding. *Pattern recognition*, 39(6) :1053–1065. *2 citations pages 110 et 114*

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., and Robinson, T. (2013). One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv :1312.3005.* *Cité page 31*

Chen, J. and Liu, Y. (2011). Locally linear embedding : a survey. *Artificial Intelligence Review*, 36(1) :29–48. *Cité page 108*

Chen, M., Xu, Z., Weinberger, K., and Sha, F. (2012). Marginalized denoising autoencoders for domain adaptation. *arXiv preprint arXiv :1206.4683.* *Cité page 43*

Chen, T. and Guestrin, C. (2016). Xgboost : A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM. *Cité page 99*

Chollet, F. et al. (2015). Keras. `https://github.com/fchollet/keras`. *2 citations pages 24 et 92*

Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE. *Cité page 86*

Church, K. W. and Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1) :22–29. *Cité page 31*

Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., and Sartin, M. (1999). Combing content-based and collaborative filters in an online newspaper. *Cité page 49*

Collobert, R., Kavukcuoglu, K., and Farabet, C. (2011a). Torch7 : A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*. *Cité page 24*

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011b). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug) :2493–2537. *Cité page 18*

Conneau, A., Schwenk, H., Barrault, L., and Lecun, Y. (2017). Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics : Volume 1, Long Papers*, volume 1, pages 1107–1116. *Cité page 32*

Covington, P., Adams, J., and Sargin, E. (2016). Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 191–198. ACM. *3 citations pages 50, 100, et 101*

Dagan, I., Glickman, O., and Magnini, B. (2006). The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer. *Cité page 48*

Dai, A. M., Olah, C., and Le, Q. V. (2015). Document embedding with paragraph vectors. *arXiv preprint arXiv :1507.07998*. *3 citations pages 31, 33, et 34*

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6) :391. *Cité page 28*

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38. *Cité page 30*

Deshpande, M. and Karypis, G. (2004). Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1) :143–177. *2 citations pages 45 et 100*

Dieleman, S., Schlüter, J., Raffel, C., Olson, E., Sobjetnderby, S. K., Nouri, D., Maturana, D., Thoma, M., Battenberg, E., Kelly, J., Fauw, J. D., Heilman, M., de Almeida, D. M., McFee, B., Weideman, H., Takács, G., de Rivaz, P., Crall, J., Sanders, G., Rasul, K., Liu, C., French, G., and Degrave, J. (2015). Lasagne : First release. *Cité page 112*

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*. *Cité page 12*

Farabet, C., Couprie, C., Najman, L., and LeCun, Y. (2013). Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8) :1915–1929. *Cité page 18*

Fouss, F., Pirotte, A., Renders, J.-M., and Saerens, M. (2007). Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on knowledge and data engineering*, 19(3) :355–369. *Cité page 43*

Glorot, X. and Bengio, Y. (2010a). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. *Cité page 22*

Glorot, X. and Bengio, Y. (2010b). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256. *Cité page 112*

Glorot, X., Bordes, A., and Bengio, Y. (2011). Domain adaptation for large-scale sentiment classification : A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520. *Cité page 20*

Good, I. J. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3-4) :237–264. *Cité page 30*

Gori, M., Pucci, A., Roma, V., and Siena, I. (2007). Itemrank : A random-walk based scoring algorithm for recommender engines. In *IJCAI*, volume 7, pages 2766–2771. *2 citations pages 41 et 43*

Gorman, B. (2016). A kaggler's guide to model stacking in practice. `http://blog.kaggle.com/2016/12/27/a-kagglers-guide-to-model-stacking-in-practice/`. [Online ; accessed 31-September-2017]. *Cité page 49*

Gupta, P., Goel, A., Lin, J., Sharma, A., Wang, D., and Zadeh, R. (2013). Wtf : The who to follow service at twitter. In *Proceedings of the 22nd international conference on World Wide Web*, pages 505–514. ACM. *Cité page 43*

Guyon, I., Bennett, K., Cawley, G., Escalante, H. J., Escalera, S., Ho, T. K., Macia, N., Ray, B., Saeed, M., Statnikov, A., et al. (2015). Design of the 2015 chalearn automl challenge. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–8. IEEE. *Cité page 24*

Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 1735–1742. IEEE. *Cité page 86*

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers : Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034. *Cité page 22*

Hecht-Nielsen, R. (1992). Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier. *Cité page 17*

Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1) :5–53. *Cité page 53*

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786) :504–507. *Cité page 19*

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv :1207.0580*. *Cité page 23*

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8) :1735–1780. *2 citations pages 34 et 86*

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression : Biased estimation for nonorthogonal problems. *Technometrics*, 12(1) :55–67. *Cité page 24*

Hoffer, E. and Ailon, N. (2015). Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer. *Cité page 86*

Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. Ieee. *Cité page 42*

Huffman, D. A. (1952). A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9) :1098–1101. *Cité page 33*

Ioffe, S. and Szegedy, C. (2015). Batch normalization : Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv :1502.03167*. *Cité page 23*

Jahrer, M., Töscher, A., and Legenstein, R. (2010). Combining predictions for accurate recommender systems. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 693–702. ACM. *Cité page 44*

Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4) :422–446. *Cité page 52*

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv :1607.01759*. *2 citations pages 25 et 32*

Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., and Wu, Y. (2016). Exploring the limits of language modeling. *arXiv preprint arXiv :1602.02410*. *2 citations pages 14 et 34*

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2016). On large-batch training for deep learning : Generalization gap and sharp minima. *arXiv preprint arXiv :1609.04836*. *Cité page 21*

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv :1408.5882*. *2 citations pages 25 et 32*

Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. (2016). Character-aware neural language models. In *AAAI*, pages 2741–2749. *2 citations pages 25 et 32*

Kingma, D. and Ba, J. (2014). Adam : A method for stochastic optimization. *arXiv preprint arXiv :1412.6980*. *2 citations pages 21 et 96*

Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE. *2 citations pages 30 et 31*

Kohavi, R. et al. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Stanford, CA. *Cité page 24*

Kohavi, R. and Longbotham, R. (2015). Online controlled experiments and a/b tests. *Encyclopedia of machine learning and data mining*, pages 1–11. *2 citations pages 50 et 56*

Koren, Y. (2008). Factorization meets the neighborhood : a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM. *Cité page 44*

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8). *4 citations pages 14, 41, 42, et 43*

Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F., and Leyton-Brown, K. (2017). Auto-weka 2.0 : Automatic model selection and hyperparameter optimization in weka. *The Journal of Machine Learning Research*, 18(1) :826–830. *Cité page 24*

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. *Cité page 86*

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105. *Cité page 18*

Kruskal, J. B. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1) :1–27. *Cité page 34*

Kusner, M., Sun, Y., Kolkin, N., and Weinberger, K. (2015). From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966. *Cité page 32*

Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196. *3 citations pages 25, 31, et 33*

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553) :436–444. *Cité page 17*

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., and Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2*, pages 396–404. Morgan-Kaufmann. *Cité page 86*

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998a). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324. *Cité page 86*

LeCun, Y., Bottou, L., Orr, G. B., and Müller, K.-R. (1998b). Efficient backprop. In *Neural networks : Tricks of the trade*, pages 9–50. Springer. *2 citations pages 17 et 21*

Lempel, R. and Moran, S. (2000). The stochastic approach for link-structure analysis (salsa) and the tkc effect. *Computer Networks*, 33(1) :387–401. *Cité page 43*

Levy, O. and Goldberg, Y. (2014). Linguistic regularities in sparse and explicit word representations. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 171–180. *Cité page 31*

Levy, O., Goldberg, Y., and Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3 :211–225. *3 citations pages 27, 29, et 31*

Li, L., Chu, W., Langford, J., and Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM. *Cité page 46*

Li, S., Kawale, J., and Fu, Y. (2015). Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 811–820. ACM. *Cité page 43*

Lim, D., McFee, B., and Lanckriet, G. R. (2013). Robust structural metric learning. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 615–623. *Cité page 83*

Ling, W., Luís, T., Marujo, L., Astudillo, R. F., Amir, S., Dyer, C., Black, A. W., and Trancoso, I. (2015). Finding function in form : Compositional character models for open vocabulary word representation. *arXiv preprint arXiv :1508.02096*. *Cité page 25*

Lu, J., Zhou, X., Tan, Y.-P., Shang, Y., and Zhou, J. (2014). Neighborhood repulsed metric learning for kinship verification. *IEEE transactions on pattern analysis and machine intelligence*, 36(2) :331–345. *Cité page 83*

Lu, Y., El Helou, S., and Gillet, D. (2013). A recommender system for job seeking and recruiting website. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 963–966. ACM. *Cité page 11*

Ma, J., Sheridan, R. P., Liaw, A., Dahl, G. E., and Svetnik, V. (2015). Deep neural nets as a method for quantitative structure–activity relationships. *Journal of chemical information and modeling*, 55(2) :263–274. *Cité page 18*

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics. *Cité page 14*

Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov) :2579–2605. *3 citations pages 19, 35, et 36*

Mahalanobis, P. C. (1936). On the generalised distance in statistics. *Proceedings of the National Institute of Sciences of India, 1936*, pages 49–55. *Cité page 83*

Manning, C. D., Raghavan, P., and Schütze, H. (2008). Matrix decompositions and latent semantic indexing. *Introduction to Information Retrieval*, pages 403–417. *2 citations pages 27 et 28*

Manning, C. D., Schütze, H., et al. (1999). *Foundations of statistical natural language processing*, volume 999. MIT Press. *Cité page 14*

Manning, D. (1995). Introduction. In *Introduction to Industrial Minerals*, pages 1–16. Springer. *Cité page 25*

Mary, J., Gaudel, R., and Preux, P. (2015). Bandits and recommender systems. In *International Workshop on Machine Learning, Optimization and Big Data*, pages 325–336. Springer. *Cité page 46*

Masters, D. and Luschi, C. (2018). Revisiting small batch training for deep neural networks. *arXiv preprint arXiv :1804.07612.* *Cité page 21*

Mazari, Z. and Recotillet, I. (2013). Generation 2004 : des débuts de trajectoire durablement marqués par la crise ? In *Cereq Bref, n. 311.* *Cité page 11*

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4) :115–133. *Cité page 17*

Mensink, T., Verbeek, J., Perronnin, F., and Csurka, G. (2012). Metric learning for large scale image classification : Generalizing to new classes at near-zero cost. *Computer Vision–ECCV 2012*, pages 488–501. *Cité page 83*

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv :1301.3781.* *3 citations pages 31, 33, et 90*

Mikolov, T., Deoras, A., Povey, D., Burget, L., and Černockỳ, J. (2011). Strategies for training large scale neural network language models. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 196–201. IEEE. *Cité page 18*

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119. *2 citations pages 31 et 33*

Mikolov, T. and Zweig, G. (2012). Context dependent recurrent neural network language model. *SLT*, 12 :234–239. *Cité page 25*

Mueller, J. and Thyagarajan, A. (2016). Siamese recurrent architectures for learning sentence similarity. In *AAAI*, pages 2786–2792. *Cité page 86*

Neculoiu, P., Versteegh, M., Rotaru, M., and Amsterdam, T. B. (2016). Learning text similarity with siamese recurrent networks. *ACL 2016*, page 148. *Cité page 86*

Olah, C. (2015). Visualizing representations : Deep learning and human beings. `http://colah.github.io/posts/2015-01-Visualizing-Representations/`. [Online ; accessed 31-September-2017]. *Cité page 35*

O'Neil, C. (2017). *Weapons of math destruction : How big data increases inequality and threatens democracy*. Broadway Books. *Cité page 106*

Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking : Bringing order to the web. Technical report, Stanford InfoLab. *Cité page 43*

Pan, R., Zhou, Y., Cao, B., Liu, N. N., Lukose, R., Scholz, M., and Yang, Q. (2008). One-class collaborative filtering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 502–511. IEEE. *Cité page 38*

Parikh, A. P., Täckström, O., Das, D., and Uszkoreit, J. (2016). A decomposable attention model for natural language inference. *arXiv preprint arXiv :1606.01933.* *2 citations pages 25 et 48*

Parzen, E. (1962). On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3) :1065–1076. *Cité page 84*

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. *Cité page 24*

Pazzani, M. J. (1999). A framework for collaborative, content-based and demographic filtering. *Artificial intelligence review*, 13(5-6) :393–408. *Cité page 49*

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn : Machine learning in Python. *Journal of Machine Learning Research*, 12 :2825–2830. *Cité page 27*

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove : Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543. *2 citations pages 27 et 31*

Pissarides, C. A. (2011). Equilibrium in the labor market with search frictions. *The American Economic Review*, 101(4) :1092–1105. *Cité page 11*

Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. (2009). Bpr : Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press. *2 citations pages 41 et 43*

Ricci, F., Rokach, L., and Shapira, B. (2011). Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer. *Cité page 38*

Rocktäschel, T., Grefenstette, E., Hermann, K. M., Kočiskỳ, T., and Blunsom, P. (2015). Reasoning about entailment with neural attention. *arXiv preprint arXiv :1509.06664*. *Cité page 48*

Rosenblatt, F. (1958). The perceptron : a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6) :386. *2 citations pages 17 et 18*

Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500) :2323–2326. *5 citations pages 15, 90, 104, 108, et 109*

Rubner, Y., Tomasi, C., and Guibas, L. J. (1998). A metric for distributions with applications to image databases. In *Computer Vision, 1998. Sixth International Conference on*, pages 59–66. IEEE. *Cité page 32*

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv :1609.04747*. *Cité page 21*

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science. *Cité page 17*

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088) :533. *2 citations pages 17 et 21*

Ruotsalo, T., Jacucci, G., Myllymäki, P., and Kaski, S. (2015). Interactive intent modeling : information discovery beyond search. *Commun. ACM*, 58(1) :86–92. *Cité page 14*

Rush, A. M., Chopra, S., and Weston, J. (2015). A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv :1509.00685*. *Cité page 25*

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM. *Cité page 44*

Saul, L. K. and Roweis, S. T. (2003). Think globally, fit locally : unsupervised learning of low dimensional manifolds. *Journal of machine learning research*, 4(Jun) :119–155. *Cité page 108*

Schmitt, T., Caillou, P., and Sebag, M. (2016). Matching jobs and resumes : a deep collaborative filtering task. In *GCAI 2016. 2nd Global Conference on Artificial Intelligence*, volume 41. *4 citations pages 15, 88, 104, et 108*

Schmitt, T., Caillou, P., and Sebag, M. (2017). Language modelling for collaborative filtering : Application to job applicant matching. In *ICTAI 2017. 29nd International Conference on Tools with Artificial Intelligence Intelligence.* *5 citations pages 15, 59, 60, 88, et 105*

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11) :2673–2681. *Cité page 86*

Schwenk, H., Rousseau, A., and Attik, M. (2012). Large, pruned or continuous space language models on a gpu for statistical machine translation. In *Proceedings of the NAACL-HLT 2012 Workshop : Will We Ever Really Replace the N-gram Model ? On the Future of Language Modeling for HLT*, pages 11–19. Association for Computational Linguistics. *Cité page 25*

Sedhain, S., Menon, A. K., Sanner, S., and Xie, L. (2015). Autorec : Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*, pages 111–112. ACM. *Cité page 43*

Seide, F. and Agarwal, A. (2016). Cntk : Microsoft's open-source deep-learning toolkit. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2135–2135. ACM. *Cité page 24*

Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Oliver, N., and Hanjalic, A. (2012). Climf : learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 139–146. ACM. *Cité page 43*

Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Oliver, N., and Hanjalic, A. (2013). Climf : Collaborative less-is-more filtering. In *IJCAI*, pages 3077–3081. *2 citations pages 43 et 46*

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 12*

Siting, Z., Wenxing, H., Ning, Z., and Fan, Y. (2012). Job recommender systems : a survey. In *Computer Science & Education (ICCSE), 2012 7th International Conference on*, pages 920–924. IEEE. *Cité page 11*

Smyth, B. and Cotter, P. (2000). A personalised tv listings service for the digital tv age. *Knowledge-Based Systems*, 13(2) :53–59. *Cité page 49*

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout : A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1) :1929–1958. *Cité page 23*

Strub, F. and Mary, J. (2015). Collaborative filtering with stacked denoising autoencoders and sparse inputs. In *NIPS workshop on machine learning for eCommerce*. *Cité page 43*

Su, X. and Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009 :4. *2 citations pages 43 et 44*

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. *2 citations pages 21 et 22*

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112. *Cité page 18*

Tenenbaum, J. B., De Silva, V., and Langford, J. C. (2000a). A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500) :2319–2323. *Cité page 34*

Tenenbaum, J. B., De Silva, V., and Langford, J. C. (2000b). A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500) :2319–2323. *Cité page 90*

Theano Development Team (2016). Theano : A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688. *2 citations pages 24 et 112*

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288. *Cité page 24*

Triskelion (201Kaggle ensembling guide. `https://mlwave.com/kaggle-ensembling-guide/`. [Online ; accessed 31-September-2017]. *Cité page 49*

Valli, A. and Véronis, J. (1999). Etiquetage grammatical des corpus de parole : problèmes et perspectives. *Revue française de linguistique appliquée*, 4(2) :113–133. *Cité page 27*

van der Maaten, L. (2009). Learning a parametric embedding by preserving local structure. *RBM*, 500(500) :26. *Cité page 35*

Vapnik, V. (1998). *Statistical learning theory*, volume 1. Wiley New York. *Cité page 24*

Verma, N., Mahajan, D., Sellamanickam, S., and Nair, V. (2012). Learning hierarchical similarity metrics. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2280–2287. IEEE. *Cité page 83*

Verstrepen, K. and Goethals, B. (2014). Unifying nearest neighbors collaborative filtering. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 177–184. ACM. *2 citations pages 43 et 116*

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM. *2 citations pages 19 et 20*

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders : Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec) :3371–3408. *Cité page 20*

Volkovs, M., Yu, G. W., and Poutanen, T. (2017). Content-based neighbor models for cold start in recommender systems. *4 citations pages 41, 99, 101, et 102*

Voorhees, E. M. et al. (1999). The trec-8 question answering track report. In *Trec*, volume 99, pages 77–82. *Cité page 52*

Wan, L., Zeiler, M., Zhang, S., Cun, Y. L., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pages 1058–1066. *Cité page 24*

Wang, C. and Blei, D. M. (2011). Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 448–456. ACM. *9 citations pages 15, 39, 41, 42, 47, 55, 69, 92, et 104*

Wang, H., Wang, N., and Yeung, D.-Y. (2015). Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1235–1244. ACM. *2 citations pages 41 et 48*

Wattenberg, M., Viégas, F., and Johnson, I. (2016). How to use t-sne effectively. *Distill*. *Cité page 35*

Weinberger, K. Q., Blitzer, J., and Saul, L. K. (2006). Distance metric learning for large margin nearest neighbor classification. In Weiss, Y., Schölkopf, P. B., and Platt, J. C., editors, *Advances in Neural Information Processing Systems 18*, pages 1473–1480. MIT Press. *Cité page 83*

Weinberger, K. Q. and Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb) :207–244. *Cité page 84*

Witten, I. H. and Bell, T. C. (1991). The zero-frequency problem : Estimating the probabilities of novel events in adaptive text compression. *Ieee transactions on information theory*, 37(4) :1085–1094. *Cité page 30*

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1) :67–82. *Cité page 32*

Wu, Y., DuBois, C., Zheng, A. X., and Ester, M. (2016). Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 153–162. ACM. *2 citations pages 41 et 43*

Xiong, H. Y., Alipanahi, B., Lee, L. J., Bretschneider, H., Merico, D., Yuen, R. K., Hua, Y., Gueroussov, S., Najafabadi, H. S., Hughes, T. R., et al. (2015). The human splicing code reveals new insights into the genetic determinants of disease. *Science*, 347(6218) :1254806. *Cité page 18*

Yin, W., Kann, K., Yu, M., and Schütze, H. (2017). Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv :1702.01923*. *Cité page 34*

Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657. *2 citations pages 25 et 32*

Zhou, T., Ren, J., Medo, M., and Zhang, Y.-C. (2007). Bipartite network projection and personal recommendation. *Physical Review E*, 76(4) :046115. *Cité page 43*

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Appariements collaboratifs des offres et demandes d'emploi

Mots clés : Système de recommandation ; filtrage collaboratif ; Apprentissage de métrique ; traitement des langues naturelles ; Réseaux de neurones

Résumé : Notre recherche porte sur la recommandation aux demandeurs d'emploi de nouvelles offres venant d'être postées (recommandation à froid). L'approche proposée étend les systèmes de recommandations développés pour le commerce électronique, en exploitant d'une part les traces d'usage de l'ensemble des demandeurs d'emploi sur les offres antérieures (données collaboratives) et d'autre part les CVs et les documents des offres d'emploi.
Une des spécificités du travail présenté est d'avoir considéré des données du monde réel, pour lesquelles nous sommes reconnaissant à l'agence et plate-forme d'emploi Qapa, et l'Association Bernard Gregory. Ces données nous ont mis en face des défis dus à l'hétérogénéité des secteurs d'emplois (respectivement le secteur de l'intérim pour Qapa et le secteur des PhD en sciences pour l'ABG), et au bruit des documents textuels.
Notre contribution est la conception et l'implémentation d'un modèle de langage, reflétant l'information des données collaboratives et permettant la recommandation à froid d'une offre nouvelle. Ce modèle de langage, dont la propriété essentielle est de définir une bonne métrique sur l'espace des offres d'emploi, est implémenté par un réseau neuronal et optimisé en définissant deux fonctions de perte. La première vise à préserver la structure locale des informations collaboratives, en s'inspirant des approches de réduction de dimension non linéaires. La seconde s'inspire des réseaux siamois pour reproduire la métrique définie par la matrice collaborative. Le passage à l'échelle de l'approche et ses performances reposent un échantillonnage spécifique des paires d'offres (considérées comme similaires et dissimilaires).
Le mérite de l'approche proposée est validé empiriquement sur les données réelles, ainsi que sur les données publiques du problème CiteULike (visant à recommander des articles scientifiques) permettant la comparaison du système proposé avec les systèmes de l'état de l'art.
Nous avons aussi démontré l'intérêt de notre approche en participant au challenge international RecSys 2017, dont les données comprennent plus d'un million de demandeurs et d'offres d'emploi. Notre rang est 15 sur 100.

Title : Collaborative Matching of Job Openings and Job Seekers

Keywords : Recommender system ; collbaorative filtering ; metric learning ; natural language processing ; neural network

Abstract : The presented work focuses on the recommendation to job seekers of new job positions (cold start recommendation). The proposed approach builds upon the state of the art in recommendation systems for e-commerce, and adapt the existing systems by exploiting i) the collaborative data reporting the interactions of job seekers with previous job openings ; ii) the job opening and CV documents.
One of the specificities of the presented work is to consider real-world data, kindly provided by the Web Hiring Agency Qapa on the one hand, and the Association Bernard Gregory on the other hand. We have thus faced the challenges of heterogeneous job sectors (blue and white collars), and the noise of textual documents.
Our contribution is the design and implementation of a language model, driven by the collaborative data and supporting the cold start recommendation. This language model, essentially aimed to define a relevant metric on the space of job openings, is implemented as a neural network and trained according to two loss functions. The former one aims to preserve the local structure of collaborative information, drawing on non-linear dimension reduction approaches. The latter one, inspired by Siamese networks, aims to reproduce the similarities induced from the collaborative data. The scaling up of the approach and its performance are based on specific samplings of the so-called positive and negative pairs of offers (deemed similar or dissimilar).
The merit of the proposed approach is demonstrated empirically on the real-world data ; for the sake of comparison with the state of the art, we also considered the public domain benchmark CiteULike, aimed at recommending scientific articles to scientists.
We also demonstrated the potential of our approach by participating into the international challenge RecSys 2017, involving over one million of job seekers and job openings. Our final rank was 15th out of 100 participants.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France