RECTO : REcommandation diminuant la Congestion par Transport Optimal

G. Bied[1,2], E. Perennes[1], S. Nathan[2], V. Naya[2], P. Caillou[2], B. Crepon[1], C. Gaillac[3], M. Sebag[2]

[1] CREST, ENSAE
[2] LISN, Université Paris-Saclay
[3] Oxford University

bied@lri.fr

Résumé

La recommandation de biens rivaux (sur le marché du travail ou des rencontres) fait face au danger majeur de la congestion. Pour concevoir un système de recommandation évitant la congestion, une approche possible repose sur le transport optimal, cherchant un appariement global entre l'ensemble des utilisateurs et l'ensemble des items selon un coût de transport à définir. L'originalité de l'approche RECTO *(REcommandation diminuant la Congestion par Transport Optimal) est de construire le coût de transport en fonction du score de recommandation, et de définir la politique de recommandation en fonction du plan de transport entre utilisateurs et items. Une validation expérimentale comparative, menée sur une base de données publique relative au marché matrimonial et une base de données propriétaire relative au marché du travail, illustre le compromis entre précision et congestion. La discussion porte sur l'intégration de critères individuels et collectifs.*

Mots-clés

Systèmes de recommandation, congestion, transport optimal, appariement.

Abstract

The matching setting, a particular case of recommendation problem, focuses on applications where a so-called item can be attributed to at most one user, with the job market and the matrimonial market as chief examples. In such cases, recommending the items preferred by each user might contribute to a congestion issue as users aiming at the same item cannot be all satisfied. While some state of art approaches proceed by repairing the recommendation policy to account for the congestion issue, other approaches take inspiration from the optimal transport (OT) framework, and aim to map the user population onto the item population in order to minimize some global transportation cost. In OT-based recommendation state-of-art approaches, the collaborative matrix (reporting the user-item matches) is interpreted as if it were the result of an OT plan ; the underlying transport cost is inferred and used e.g. to propose new congestion-avoiding recommendation policies. In this paper, another OT-based recommendation strategy is defined, noting that the collaborative matrix reflecting the individual decisions can hardly be considered as the result of a (centralized) OT plan. Accordingly, the proposed algorithm first learns a recommendation score from the data and then defines a cost matrix, with the transportation cost decreasing depending on the matching relevance. The optimal transport plan is thereafter used for the recommendation. The experimental validation of the approach presents comparative results on benchmark data on the matrimonial market, and proprietary real-world data on the job market, illustrating the trade-off between the recall and the congestion indicators. The discussion focuses on the integration of individual and collective criteria.

Keywords

Recommender systems, Congestion, Optimal Transport, Matching.

1 Introduction

La motivation principale de cet article consiste à s'appuyer sur l'état de l'art des systèmes de recommandation [24] pour assister les services publiques de l'emploi. Si les systèmes de recommandation usuels visent à recommander à chaque utilisateur l'offre d'emploi qui leur est la plus désirable, il peut sembler peu approprié de recommander la même offre à de nombreux demandeurs d'emploi : cela induirait un phénomène de congestion au niveau de la population et une satisfaction individuelle finale médiocre. Plus généralement, dans des domaines tels que les marchés du travail ou matrimoniaux, correspondant à une configuration de *recommandation réciproque* [18], une politique de recommandation adéquate devrait prendre en compte les populations de demandeurs d'emploi et d'offres d'emploi dans leur globalité, et connecter les deux populations de manière à éviter la congestion.

S'inspirant de travaux connexes sur les systèmes de recommandation [16, 17, 5] et en économétrie [7, 10], cet article étudie le couplage du transport optimal [9, 21] avec les systèmes de recommandation. L'approche proposée, appelée *REcommandation diminuant la Congestion par Transport Optimal* (RECTO), apprend des correspondances entre les populations d'utilisateurs (demandeurs d'emploi) et

d'items (offres d'emploi), visant à trouver un compromis entre l'intérêt des articles recommandés et une diversité suffisante des recommandations au niveau de la population (par opposition à la sérendipité en recommandation [15], qui vise la diversité des items recommandés au niveau individuel). Les questions scientifiques abordées dans l'article concernent donc : i) la définition d'un indicateur mesurant la qualité de recommandations en termes de congestion ; ii) la mise au point d'un algorithme visant à limiter cette congestion ; iii) l'évaluation du compromis entre l'indicateur de performance de recommandation usuel qu'est le recall et la congestion.

La contribution de l'article est triple. Tout d'abord, l'évitement de la congestion est formalisé dans le cadre du transport optimal (section 3). Deuxièmement, l'algorithme RECTO proposé pour résoudre ce problème est agnostique quant à la distribution des données (par opposition aux conditions requises dans [16, 17, 10, 7]) et moins exigeant computationnellement que les approches d'optimisation combinatoire, par exemple [26]. Troisièmement, les mérites de RECTO sont démontrés expérimentalement sur i) un benchmark public dans le domaine du marché matrimonial, utilisé pour l'évaluation comparative avec [16] ; ii) un vaste jeu de données propriétaire relatif au marché du travail [1], et fournissent des leçons inattendues sur les interactions des indicateurs de recall et de congestion.

2 Notations et État de l'art

Cette section présente le problème de recommandation réciproque, renvoyant le lecteur à [18] pour un état de l'art plus complet, et discute quelques travaux connexes. Le cadre du transport optimal [9, 21] est ensuite brièvement introduit.

Notations. Soit n (respectivement m) le nombre d'utilisateurs (resp. d'items), avec x_i (resp. y_j) la description du i-ème utilisateur (resp. du j-ème item). La matrice collaborative booléenne $M_{i,j}$ est telle que $M_{i,j} = 1$ si et seulement si l'utilisateur i a sélectionné l'item j.

Position du problème. Un système de recommandation apprend généralement une fonction de score ϕ telle que la matrice définie à partir de $\phi_{i,j} = \phi(x_i, y_j)$ soit proche de la matrice collaborative M (en termes d'erreur quadratique moyenne ou de divergence de Kullback-Leibler), éventuellement pénalisée par un terme de régularisation [1]. Sans perte de généralité, on suppose par la suite que les items recommandés au i-ème utilisateur sont ordonnés par ordre croissant de $\phi_{i,j}$.

Dans les problèmes de recommandation réciproque [18], l'item j est soumis à une contrainte de capacité n_j : seuls les n_j meilleurs utilisateurs sélectionnant cet item peuvent être servis. De nouveaux objectifs d'optimisation et des algorithmes spécifiques doivent être définis pour tenir compte de ces contraintes.

Travaux connexes. Une première approche de recommandation réciproque, [12], apprend une fonction de score ϕ comme solution d'un problème d'optimisation contraint. [26] considère la recommandation réciproque comme un problème d'optimisation multi-objectif (NP-difficile), introduisant la satisfaction des contraintes de capacité comme un objectif supplémentaire ; l'optimisation est réalisée en utilisant une approche gloutonne. Dans [2], un modèle auxiliaire prédit la popularité attendue d'un item ; la recommandation est réparée au niveau individuel, en décalant vers le haut ou vers le bas les items recommandés à un utilisateur donné en fonction de leur popularité globale prévue. Dans le contexte de sites de rencontre, [6], s'inspirant de modèles économiques décentralisés, construisent une fonction d'utilité reflétant l'intérêt de x_i pour y_j et réciproquement ; ils utilisent une approche de transport optimal (voir ci-dessous) pour calculer les recommandations finale à partir du score d'utilité.

Transport optimal computationnel. Le transport optimal (OT) vise à faire correspondre une certaine distribution (continue ou discrète) μ à une autre distribution ν. Dans la suite de l'article, μ (respectivement ν) désigne la distribution discrète uniforme sur l'ensemble des n utilisateurs (resp. sur l'ensemble des m items). En notant $\Gamma(\mu, \nu)$ l'ensemble des mesures telles que leurs marginales par rapport au premier et deuxième argument sont respectivement μ et ν, et en définissant $C_{i,j}$ comme le coût de la correspondance de i à j, le problème du transport optimal est de trouver une distribution jointe $\gamma^* \in \Gamma(\mu, \nu)$ telle que :

$$\gamma^*(C) \in \arg\min_{\gamma \in \Gamma(\mu,\nu)} \sum_{i=1}^{n} \sum_{j=1}^{m} \gamma_{i,j} C_{i,j} \tag{1}$$

Une relaxation de ce problème d'optimisation, proposée par [9], y ajoute un terme entropique :

$$\gamma^*(C, \varepsilon) \in \arg\min_{\gamma \in \Gamma(\mu,\nu)} \sum_{i=1}^{n} \sum_{j=1}^{m} \gamma_{i,j} \left(C_{i,j} + \varepsilon \log(\gamma_{i,j})\right) \tag{2}$$

avec ε le poids donné à la régularisation. La solution optimale de l'équation (2) prend la forme $\gamma_{i,j} = \alpha_i \exp\left(-C_{i,j}/\varepsilon\right) \beta_j$, où α et β reflètent les contraintes sur les marginales de γ.

Discussion Les principales approches de recommandation basées sur l'OT supposent que la matrice collaborative observée M est générée comme optimum d'un plan d'OT basé sur un coût d'appariement C à estimer [7, 11, 16, 17] : elles apprennent C à partir des données d'entraînement et utilisent le modèle de coût estimé pour construire des appariements sur de nouvelles données.

Dans le contexte du marché du travail, il est cependant discutable de considérer que les appariements réels (*i.e.* la matrice collaborative observée M) doivent être vus comme la solution d'un plan de transport optimal : par construction, M est le résultat d'un processus décentralisé tandis que la solution d'OT résulte d'un processus centralisé (les deux ne coincidant que sous de fortes hypothèses). En conséquence, l'approche proposée est structurée en deux phases : l'apprentissage de la fonction de coût d'appariement C à partir de M (sans supposer que M soit une solution d'un problème d'OT), et l'utilisation de C dans un problème d'OT.

1. Fourni par *Pôle emploi*.

3 Présentation de RECTO

Soit $\phi_{i,j} \in \mathbb{R}$ le score de recommandation de l'item j pour l'utilisateur i, et définissons l'indicateur booléen $\mathbb{1}_{i \to j,k,\phi}$ comme valant 1 si j figure parmi les k meilleures recommandations pour i d'après ϕ. L'indice ϕ sera omis par la suite lorsque le contexte ne présente pas d'ambiguïté.

Critères de performance. En plus de l'indicateur standard qu'est le Recall@k, qui mesure la fraction d'utilisateurs pour lesquels l'item préféré est classé parmi les k meilleures recommandations,

$$\text{Recall@}k(\phi) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} M_{i,j}.\mathbb{1}_{i \to j,k} \tag{3}$$

nous définissons la notion de part de marché de l'item $MS_\ell(j)$, correspondant à la fraction d'utilisateurs i pour lesquels j figure parmi les ℓ meilleurs articles recommandés à i (avec $\ell < m$) :

$$MS_\ell(j) = \frac{1}{n \times \ell} \sum_{i=1}^{n} \mathbb{1}_{i \to j,\ell}$$

Nous définissons la congestion comme moins l'entropie des parts de marché : plus les recommandations sont réparties uniformément sur tous items, plus leurs parts de marché sont similaires, moins la congestion est importante.

$$\text{Congestion@}\ell(\phi) := \sum_{j=1}^{m} MS_\ell(j) \log\left(MS_\ell(j)\right) \tag{4}$$

Cet indicateur de congestion est normalisé sur $[-1, 0]$ en le divisant par $\log(m)$; l'évitement parfait de la congestion est obtenu pour des parts de marché égales des items, avec -1 comme valeur correspondante de l'indicateur.

Utilisation du Transport Optimal. Comme indiqué précédemment, le transport optimal s'applique sur une matrice de coûts $C_{i,j}$ dépendant du score de recommandation appris $\phi_{i,j}$. L'apprentissage de bout en bout de $C_{i,j}$ sera considéré dans des travaux ultérieurs. Nous considérons ici $C_{i,j} = g(\phi_{i,j})$, où g est une fonction scalaire monotone, considérée comme un hyperparamètre de l'approche, telle que le coût $C_{i,j}$ de transporter i vers j augmente avec $\phi_{i,j}$ (c'est-à-dire que la pertinence de la recommandation de j à j diminue). $\phi_{i,j}$ est plafonné au score du millième élément recommandé à chaque i par ϕ, noté $\phi_i^{(1000)}$ ($\phi_{i,j} \leftarrow \max(\phi_{i,j}, \phi_i^{(1000)})$ dans la suite).
Quatre fonctions g ont été considérées, respectivement linéaires, exponentielles de $\phi_{i,j}$, basées sur le rang, ou de type NDCG [3].
Pour une comparaison équitable des résultats obtenus avec le même poids de régularisation entropique ε, les $C_{i,j}$ sont normalisés de sorte que $\sum_{i,j} C_{i,j} = 1$.

L'algorithme RECTO**.** RECTO est un processus en 2 étapes : i) apprentissage d'une fonction de score ϕ ; ii) résolution du problème de transport optimal défini à partir de $C_{ij} = g(\phi_{i,j})$.

RECTO **: 1. Apprentissage de** ϕ**.** Les deux approches d'apprentissage de ϕ considérées sont XGBOOST et des réseaux de neurones (NN). XGBOOST est un système de recommandation basé sur des arbres boostés utilisé par de nombreux algorithmes de l'état de l'art [24], qui peut être efficacement entraîné en sous-échantillonnant agressivement les paires négatives (i, j), au détriment d'un passage à l'échelle plus limité dû à l'utilisation de variables croisées telles que la distance. NN utilise un réseau de neurones profond, dont l'architecture est adaptée aux spécificités du domaine (par exemple, en considérant des sous-modules dédiés aux informations géographiques ou aux compétences). Les descriptions de l'utilisateur x_i et de l'item y_j sont respectivement projetés dans des espaces latents, notés $z_{x,i}$ et $z_{y,j}$, et leur adéquation $\phi_{i,j}$ est paramétrée comme $z_{x,i}^T A z_{y,j}$ avec A une matrice [4]. Les projections et la matrice A sont apprises de manière end-to-end en utilisant une perte "triplet" [25]. Formellement, en supposant que chaque utilisateur de l'entraînement i est associé à un seul item j, on peut réordonner les colonnes de la matrice de telle sorte que l'utilisateur i soit associé à l'item i, et l'objectif d'apprentissage est alors :

$$\text{Minimiser} \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{m} (\phi_{i,i} - \phi_{i,j} + \eta)+ \tag{5}$$

avec $A+ = \max(0, A)$ et $\eta > 0$. En pratique, un échantillonnage négatif est utilisé pour faire face au nombre de paires négatives. Les hyperparamètres de XGBOOST et NN sont détaillés en annexe B.

RECTO **: 2. Transport optimal.** En fonction du poids de régularisation ε et de la fonction g (avec $C_{ij} = g(\phi_{i,j})$), la distribution discrète γ est entraînée en optimisant l'équation (2). Remarquons que l'extension de l'approche au cas général de recommandation réciproque (*e.g.* où plusieurs postes peuvent être ouverts pour l'offre d'emploi j) est immédiate en rendant ν_j proportionnelle à la contrainte de capacité de l'item j.
Enfin, pour déterminer la recommandation finale, RECTO procède de manière déterministe, en classant les j articles recommandés à l'utilisateur i par ordre décroissant par rapport à $\gamma_{i,j}$.

4 Validation expérimentale

Cette section présente la validation expérimentale de l'approche, réalisée sur deux jeux de données : des données publiques sur le marché matrimonial (notées MAR), utilisées par [16], pour une comparaison avec l'état de l'art ; et un ensemble de données propriétaires (notées JOB) fournies par *Pôle emploi*.
Le premier objectif des expériences est d'évaluer l'efficacité de l'approche proposée en termes de compromis entre le recall et la congestion. Le second objectif est d'étudier comment les résultats dépendent des hyperparamètres de l'approche - ϕ appris avec XGBOOST ou NN ; poids de régularisation entropique ε variant entre $10^{-2}, \ldots, 10^{2}$; coût de transport C_{ij} défini comme $g(\phi_{ij})$ avec g variant entre {Id, Exp, Ndcg, Rank}.

Sept indicateurs de performance sont associés à chaque hyperparamètre : recall@k avec $k = 1,\ 10,\ 100$, congestion@k avec $k = 1,\ 10$, et coverage@k avec $k = 1,\ 10$, indiquant la fraction d'éléments impliqués dans la recommandation top-k d'au moins un utilisateur. D'autres indicateurs de performance sont présentés et discutés dans l'annexe C.

4.1 Données MAR

Description des données Les données comprennent 2 475 hommes et 2 475 femmes, répartis en 50 groupes. Chaque individu est décrit par 11 caractéristiques principalement ordinales. L'appariement 1 à 1 est décrit au niveau individuel et les données comprennent également la matrice collaborative $M_{c,c'}$, qui indique la fraction d'appariements entre les hommes du groupe c et les femmes du groupe c'.

Benchmarks. Les résultats de référence sur MAR sont ceux de RIOT [16], utilisant une factorisation basée sur SVD et itemKNN [8]. Les indicateurs de performance comprennent le RMSE et le MAE entre la matrice collaborative M au niveau des groupes et la matrice de recommandation estimée, mesurée à l'aide d'une validation croisée à 5 plis. RECTO est également évalué au niveau individuel, à l'aide des indicateurs de performance définis en section 4.

Resultats. Les tableaux 1 et 2 affichent respectivement les résultats obtenus sur MAR au niveau du groupe [2] et au niveau individuel.

Au niveau des groupes, γ^{XGB} obtient de légèrement meilleurs résultats (de manière statistiquement significative) par rapport à RIOT en termes de RMSE (8,89± 0,11 contre 8,98± 0,17) et de MAE (5,80± 0,13 contre 5,79± 0,12). γ^{NN} est également légèrement meilleur que RIOT. Les autres algorithmes de référence (aléatoire, PMF, SVD et itemKNN) sont aussi dominés.

Au niveau individuel, XGBOOST est significativement plus performant que NN en termes de recall et de congestion pour toutes les valeurs de k considérées.

γ^{XGB} améliore la congestion au détriment du recall : l'amélioration de la congestion (de -.84 à -.98) est obtenue en diminuant le recall@10 (de 28.4% à 23.7% pour la combinaison d'hyperparamètres $g = Id, \varepsilon = 10^{-2}$).

Pour γ^{NN}, la congestion peut être considérablement amélioré (de -.84 à -.98, pour $g = Id, \varepsilon = 10^{-2}$) tout en préservant le recall@10 (environ 15.4%).

Les indicateurs de recall et de congestion ne sont toutefois pas nécessairement antagonistes dans le problème de MAR : par construction, la matrice collaborative recherchée est une permutation. La principale difficulté de ce problème de recommandation semble donc provenir de la petite taille du jeu de données et de la description peu détaillée des individus.

2. La différence avec [16] s'explique par le fait qu'un bug a été trouvé (et corrigé) dans le code public pour RIOT et les autres algorithmes utilisés par [16], au niveau de la division de l'erreur par le nombre de plis dans chaque itération. Le classement final des algorithmes n'est pas affecté par la correction du bug.

4.2 Données JOB

Description des données L'ensemble d'entraînement comprend environ 1 650 000 demandeurs d'emploi, 477 000 offres d'emploi et 43 000 appariements (contrats signés) en Ile de France au cours de la période février-octobre 2018. La description x_i (respectivement y_j) d'un demandeur d'emploi (resp. d'une offre d'emploi) se trouve dans $\mathbb{R}^{448}$ (resp. $\mathbb{R}^{582}$). La fonction ϕ est apprise sur l'ensemble d'apprentissage ; le plan de transport optimal γ est calculé sur l'ensemble de test restreint au secteur d'emploi de la logistique pour des raisons de passage à l'échelle : il comprend 110 000 demandeurs d'emploi, 14 200 annonces d'emploi et 450 embauches en nov. 2018.

Résultats. Le tableau 3 résume les résultats, avec trois enseignements principaux. Premièrement, NN est dominé par XGBOOST sur les trois indicateurs de performance : recall, coverage et congestion ; en outre, le moindre recall de NN(4 % de perte de recall@100) s'accompagne d'un coverage beaucoup plus faible (7 % de perte de coverage@1). Cette contre-performance est imputée à l'architecture du réseau neuronal (des expériences en cours suggèrent que l'ajout d'un second réseau n'utilisant que le Top 1000 de NN en entrée permet à NN de rattraper son retard). Deuxièmement, la coverage (@1 et @10) augmente de façon monotone et le recall (@1, @10, @100) diminue de façon monotone lorsque ε passe de 1 à 0,01, ce qui laisse peu d'espoir de pouvoir combiner un bon coverage avec un recall décent. Il est plus encourageant de constater que la congestion@1 peut être améliorée de manière significative (de -0,62 à -0,78) au prix d'une perte de recall modérée (le recall@10 passe de 62% à 56%) pour $g = Id, \varepsilon = 0,1$.

L'option $g = ndcg$ a peu d'effets (légèrement positifs) pour $\varepsilon = 1$ et des effets fortement négatifs pour $\varepsilon = .1$ ou $.01$.

Il est surprenant de constater qu'en diminuant ε, on obtient une meilleure congestion (plus faible) au prix d'un moins bon recall. Cet effet est inattendu car plus ε est élevé, plus le plan de transport γ est uniforme (toutes choses étant égales par ailleurs) et plus la congestion semblerait devoir être faible. Une interprétation possible (étayée par des expériences complémentaires présentées dans l'annexe A) est que tous les indicateurs de performance dépendent de l'ordre induit par γ, par opposition aux valeurs réelles de $\gamma_{i,j}$. Bien que la variance de $\gamma_{i,j}$ diminue à mesure que ε augmente, le phénomène du "winner takes all" persiste, c'est-à-dire que les premières recommandations de tous les utilisateurs ne couvrent qu'un maigre 13% à 20% des offres.

Les temps de calcul (tableau 4) pour XGBOOST (environ 2 heures) et NN (environ 30 mn) restent relativement limités. Le temps associé au calcul du transport optimal augmente à mesure que ε diminue, jusqu'à environ 10 mn pour $\varepsilon = .01$ (voir aussi [22]). La majorité du temps de calcul est consacrée au calcul des recommandations avec XGBOOST et γ^{XGB}, du fait de la nécessité de calculer les ϕ_{ij} pour toutes les paires $i - j$.

TABLE 1 – Résultats sur MAR au niveau des groupes ; moyenne et écart-type de la RMSE et de la MAE par rapport à la matrice collaborative M. Les résultats pour RECTO correspondent à $g = Id+$, $\varepsilon = 1$.

	Random	PMF	SVD	itemKNN	RIOT	γ^{NN}	γ^{XGB}
RMSE	10.71± 0.13	446.6± 9.86	441.4± 11.2	9.36± 0.12	9.12± 0.12	8.98± 0.17	8.89± 0.11
MAE	7.22± 0.06	251.3± 6.00	249.2± 5.71	6.30± 0.03	5.98± 0.10	5.80± 0.13	5.79± 0.12

TABLE 2 – Résultats sur MAR au niveau individuel : Recall, Coverage et Congestion.

	Algorithm	Recall (%)		Coverage (%)		Congestion	
		@1	@10	@1	@10	@1	@10
	ϕ Random	0.16	2.27	63.32	100	-0.90	-0.98
RECTO-XGB	ϕ XGBOOST	7.93	27.88	48.55	98.69	-0.84	-0.94
RECTO-XGB	γ^{XGB},$g = Id+$,$\varepsilon = 1.0$	8.05	28.41	49.77	99.18	-0.85	-0.95
RECTO-XGB	γ^{XGB},$g = Id+$,$\varepsilon = 0.1$	8.01	27.02	72.73	100	-0.93	-0.95
RECTO-XGB	γ^{XGB},$g = Id+$,$\varepsilon = 0.01$	6.47	23.77	96.05	100	-0.98	-0.84
RECTO-XGB	γ^{XGB},$g = ndcg$,$\varepsilon = 1.0$	7.93	28.2	48.55	99.02	-0.84	-0.95
RECTO-XGB	γ^{XGB},$g = ndcg$,$\varepsilon = 0.1$	8.10	25.72	59.42	100	-0.89	-0.93
RECTO-XGB	γ^{XGB},$g = ndcg$,$\varepsilon = 0.01$	6.06	19.49	94.26	100	-0.98	-0.73
RECTO-NN	ϕ NN	3.82	15.50	46.27	98.00	-0.83	-0.93
RECTO-NN	γ^{NN},$g = Id+$,$\varepsilon = 1.0$	2.84	14.32	38.86	92.47	-0.80	-0.90
RECTO-NN	γ^{NN},$g = Id+$,$\varepsilon = 0.1$	3.94	15.46	70.12	100	-0.92	-0.98
RECTO-NN	γ^{NN},$g = Id+$,$\varepsilon = 0.01$	3.78	15.46	93.48	100	-0.98	-0.95
RECTO-NN	γ^{NN},$g = ndcg$,$\varepsilon = 1.0$	3.82	15.63	46.27	98.73	-0.83	-0.94
RECTO-NN	γ^{NN},$g = ndcg$,$\varepsilon = 0.1$	4.23	13.87	57.99	99.91	-0.88	-0.93
RECTO-NN	γ^{NN},$g = ndcg$,$\varepsilon = 0.01$	2.89	11.60	93.44	100	-0.98	-0.72

TABLE 3 – Résultats sur JOB : Recall, Coverage et Congestion.

	Algorithm	Recall (%)			Coverage (%)		Congestion	
		@1	@10	@100	@1	@10	@1	@10
	ϕRandom	0	0.21	0.65	99.95	100	-0.99	-0.99
RECTO-XGB	ϕ XGB	9.62	31.40	61.59	12.94	25.16	-0.62	-0.64
RECTO-XGB	γ^{XGB},$g = Id+$,$\varepsilon = 1.0$	4.81	21.99	57.87	21.61	31.76	-0.74	-0.75
RECTO-XGB	γ^{XGB},$g = Id+$,$\varepsilon = 0.1$	2.18	15.31	56.01	27.54	41.24	-0.78	-0.81
RECTO-XGB	γ^{XGB},$g = Id+$,$\varepsilon = 0.01$	4.37	20.45	43.21	46.75	57.61	-0.85	-0.79
RECTO-XGB	γ^{XGB},$g = ndcg$,$\varepsilon = 1.0$	9.62	31.61	62.36	12.96	26.14	-0.62	-0.67
RECTO-XGB	γ^{XGB},$g = ndcg$,$\varepsilon = 0.1$	8.97	25.38	46.06	14.69	30.84	-0.67	-0.74
RECTO-XGB	γ^{XGB},$g = ndcg$,$\varepsilon = 0.01$	5.03	14.00	18.81	36.81	57.52	-0.82	-0.81
RECTO-NN	ϕ NN	5.68	28.66	57.98	6.02	17.78	-0.46	-0.49
RECTO-NN	γ^{NN},$g = Id+$,$\varepsilon = 1.0$	6.78	26.14	60.39	11.99	26.30	-0.62	-0.65
RECTO-NN	γ^{NN},$g = Id+$,$\varepsilon = 0.1$	2.40	19.03	50.43	28.23	40.16	-0.80	-0.79
RECTO-NN	γ^{NN},$g = Id+$,$\varepsilon = 0.01$	3.93	16.30	27.89	53.38	62.35	-0.83	-0.70
RECTO-NN	γ^{NN},$g = ndcg$,$\varepsilon = 1.0$	5.68	27.46	59.08	6.02	19.75	-0.46	-0.55
RECTO-NN	γ^{NN},$g = ndcg$,$\varepsilon = 0.1$	5.25	23.3	49.01	8.85	26.40	-0.53	-0.65
RECTO-NN	γ^{NN},$g = ndcg$,$\varepsilon = 0.01$	1.53	12.36	24.28	35.41	51.56	-0.81	-0.81

TABLE 4 – Temps de calcul en secondes sur JOB (moyenne sur toutes les options g). NN a été entraîné sur un serveur équipé de 2 processeurs Intel Xeon Silver 4214 2,2 GHz, de 192Go de RAM et d'un GPU Tesla T4. XGBOOST est entraîné sur un serveur DELL PowerEdge R640 avec 2X Intel Xeon Gold 6130 2.10GHz CPUs (2×16 cores) et 384Go RAM. Le plan de transport optimal est calculé sur le DELL avec les mêmes ressources que pour XGBOOST.

Temps de calcul	ϕ		γ^{XGB}			γ^{NN}		
	XGBoost	NN	$\varepsilon = 0.01$	$\varepsilon = 0.1$	$\varepsilon = 1$	$\varepsilon = 0.01$	$\varepsilon = 0.1$	$\varepsilon = 1$
Total	104,340	4,104	104,778	104,394	104,385	4,611	4,156	4,148
(incl. Appr./OT)	(7,454/−)	(2,039/−)	(7,454/438)	(7,454/54)	(7,454/45)	(2,039/507)	(2,039/52)	(2,039/44)

5 Conclusion et Perspectives

Dans la lignée de l'IA éthique [14], cet article vise à prévenir les effets indésirables des systèmes de recommandation de bien rivaux, notamment dans le domaine du marché du travail. En effet, si certaines offres d'emploi sont recommandées à de nombreux demandeurs d'emploi, un phénomène de congestion est observé au niveau global, entraînant un gaspillage de temps et d'autres effets préjudiciables pour les demandeurs d'emploi comme pour les recruteurs.

L'approche proposée s'inspire du transport optimal, avec l'idée de "transporter" globalement la population des demandeurs d'emploi sur la population des offres d'emploi, en visant un recall décent avec une faible congestion. La question clé devient alors la définition du coût de transport. Dans cet article, le coût de transport est basé sur un score de recommandation appris de façon classique (sans prendre en compte la congestion). Les leçons surprenantes tirées de l'application de l'approche sur un vaste jeu de données issu du monde réel sont que le coût de transport et la régularisation du transport (utilisée pour garantir le passage à l'échelle de l'OT) interagissent de manière subtile. En particulier, une forte régularisation (qui devrait normalement donner un plan de transport uniforme) dégrade considérablement le recall, *tout en n'améliorant pas la congestion*.

Ce travail ouvre deux perspectives principales. Du côté algorithmique, des travaux futurs se pencheront sur l'apprentissage de bout en bout du plan de recommandation, en tenant compte à la fois du recall et du coverage. Un objectif intermédiaire est d'apprendre la fonction g utilisée pour définir les coûts de transport à partir de la fonction de score.

Une autre perspective, dans le cadre du marché du travail ou des rencontres matrimoniales, consisterait à estimer les caractéristiques structurelles du comportement individuel, telles que l'aversion au risque ou à la concurrence. Ces caractéristiques pourraient à leur tour être incorporées aux recommandations, permettant de prendre en compte le niveau de congestion acceptable pour différents types d'individus dans la définition d'une stratégie globale évitant la congestion.

Plus fondamentalement, la construction d'un "système de recommandation d'emploi équitable" devrait être considérée comme l'apprentissage d'un modèle prescriptif (*suivez cette politique pour atteindre les objectifs souhaités*), plutôt que d'un modèle prédictif (*suivez cette politique car elle estime avec précision les préférences des utilisateurs*) [13, 20, 23]. Les mérites et les limites d'un tel modèle prescriptif nécessiteront des expériences fondées sur la méthodologie des essais contrôlés randomisés [19].

Remerciements

Cette recherche a été soutenue par l'institut convergences DATAIA dans le cadre du "Programme d'Investissement d'Avenir" (ANR-17-CONV-0003) géré par l'Institut Polytechnique de Paris et l'Université Paris-Saclay.

Références

[1] Charu C Aggarwal. *Recommender systems*, volume 1. Springer, 2016.

[2] Fedor Borisyuk, Liang Zhang, and Krishnaram Kenthapadi. Lijar : A system for job application redistribution towards efficient career marketplace. In *Proceedings of the 23rd ACM SIGKDD*, pages 1397–1406. ACM, 2017.

[3] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96, 2005.

[4] Gal Chechik, Uri Shalit, Varun Sharma, and Samy Bengio. An online algorithm for large scale image similarity learning. *Advances in Neural Information Processing Systems*, 22 :306–314, 2009.

[5] Kuan-Ming Chen, Yu-Wei Hsieh, and Ming-Jen Lin. Prediction and congestion in two-sided markets : Economist versus machine matchmakers. *SSRN Electronic Journal*, 2019.

[6] Kuan-Ming Chen, Yu-Wei Hsieh, and Ming-Jen Lin. Reducing recommendation inequality via two-sided matching : A field experiment of online dating. *SSRN Electronic Journal*, 01 2020.

[7] Pierre-André Chiappori and Bernard Salanié. The econometrics of matching models. *Journal of Economic Literature*, 54(3) :832–61, 2016.

[8] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, page 3946, New York, NY, USA, 2010. Association for Computing Machinery.

[9] Marco Cuturi. Sinkhorn distances : Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, page 22922300, 2013.

[10] Alfred Galichon. *Optimal transport methods in economics*. Princeton University Press, 2018.

[11] Alfred Galichon and Bernard Salanié. Cupids invisible hand : Social surplus and identification in matching models. *Available at SSRN 1804623*, 2020.

[12] Stanislao Gualdi, Matús Medo, and Yi-Cheng Zhang. Crowd avoidance and diversity in socio-economic systems and recommendation. *CoRR*, abs/1301.1887, 2013.

[13] Guido W Imbens and Donald B Rubin. *Causal inference in statistics, social, and biomedical sciences*. Cambridge University Press, 2015.

[14] Jon Kleinberg, Jens Ludwig, Sendhil Mullainathan, and Ashesh Rambachan. Algorithmic fairness. In *Aea papers and proceedings*, volume 108, pages 22–27, 2018.

[15] Matevž Kunaver and Tomaž Požrl. Diversity in recommender systems–a survey. *Knowledge-Based Systems*, 123 :154–162, 2017.

[16] Ruilin Li, Xiaojing Ye, Haomin Zhou, and Hongyuan Zha. Learning to match via inverse optimal transport. *J. Mach. Learn. Res.*, 20(80) :1–37, 2019.

[17] Ruishan Liu, Akshay Balsubramani, and James Zou. Learning transport cost from subset correspondence. *arXiv preprint arXiv :1909.13203*, 2019.

[18] Ivan Palomares, Carlos Porcel, Luiz Pizzato, Ido Guy, and Enrique Herrera-Viedma. Reciprocal recommender systems : Analysis of state-of-art literature, challenges and opportunities towards social recommendation. *arXiv preprint arXiv :2007.16120*, 2020.

[19] W. Pearce and S. Raman. The new randomised controlled trials (RCT) movement in public policy : challenges of epistemic governance. *Policy Sci*, 47 :387402, 2014.

[20] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference : foundations and learning algorithms*. MIT press, 2017.

[21] Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends® in Machine Learning*, 11(5-6) :355–607, 2019.

[22] Bernhard Schmitzer. Stabilized sparse scaling algorithms for entropy regularized transport problems. *arXiv preprint arXiv :1610.06519*, 2019.

[23] Flavian Vasile and Stephen Bonner. Causal embeddings for recommendation : An extended abstract. In *IJCAI-19*, pages 6236–6240. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[24] Maksims Volkovs, Guang Wei Yu, and Tomi Poutanen. Content-based neighbor models for cold start in recommender systems. In *Proceedings of the Recommender Systems Challenge 2017 - RecSys Challenge 17*. ACM Press, 2017.

[25] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *J. Mach. Learn. Res.*, 10(2), 2009.

[26] Bin Xia, Junjie Yin, Jian Xu, and Yun Li. We-rec : A fairness-aware reciprocal recommendation based on walrasian equilibrium. *Knowledge-Based Systems*, 182, 08 2019.

Appendices

Annexe A : Une régularisation entropique plus élevée peut ne pas réduire la congestion

D'après [21] (prop. 4.1), lorsque le poids ε du terme de régularisation entropique tend vers ∞, la solution γ du problème de transport optimal régularisé tend vers une distribution uniforme. Lorsque $\varepsilon \to 0$, en revanche, la solution converge vers le plan de transport optimal avec entropie maximale. De manière informelle, augmenter ε conduit à des solutions γ moins parcimonieuse.

Cependant, de manière inattendue, l'utilisation de γ pour un processus de recommandation par classement n'implique pas nécessairement qu'un γ plus uniforme entraîne moins de congestion.

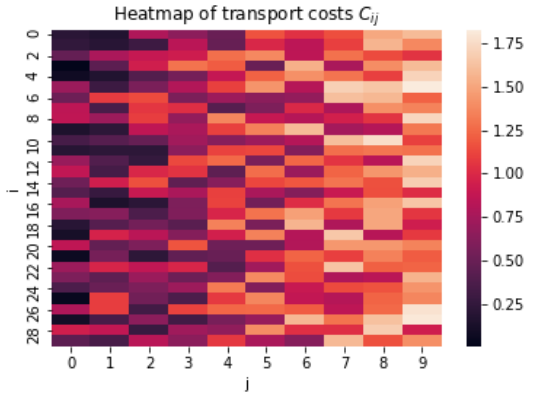
Ce phénomène peut être analysé en simulation. 1 000 matrices de coût C de taille $n = 30$, $m = 10$ sont générées indépendamment, avec $C_{ij} \sim \mathcal{U}(\frac{j}{m}, \frac{j}{m} + 1)$ (les items étant ordonnés par attractivité croissante). Les plans de transport γ avec marges uniformes par rapport aux utilisateurs et aux items sont ensuite calculés à l'aide de l'algorithme de Sinkhorn et un poids de régularisation entropique $\varepsilon = 100$ et $\varepsilon = 0,01$. La moyenne et l'écart type sur les 1 000 exécutions de la congestion obtenue après les classements issu ces plans indiquent que la congestion est significativement plus élevée pour la valeur de ε la plus élevée :

ε	moyenne de congestion@1	écart-type
100	-0.940521	0.029445
0.01	-0.996059	0.003586

Les figures 1, 2, 3 et 4 illustrent ce phénomène sur une seule simulation représentative. Les valeurs de γ_{ij} sont plus uniformes lorsque $\varepsilon = 100$ que lorsque $\varepsilon = 0,01$, mais le classement par ligne conduit à une répartition plus inégale des recommandations vers les différentes offres.

Dans l'ensemble, une régularisation entropique plus élevée a un impact indéterminé sur la congestion et peut même l'augmenter en pratique. Le choix de ε doit donc être basé sur un ensemble de validation, ainsi que sur des critères numériques pour la convergence de l'algorithme de Sinkhorn. On peut noter que l'utilisation de valeurs extrêmement faibles de ε avec une implémentation naïve de l'algorithme de Sinkhorn peut avoir des conséquences néfastes sur la stabilité numérique ainsi que sur la vitesse de convergence, bien que des alternatives aient été développées, par exemple dans [22].

FIGURE 1 – Coûts bruts



Annexe B : Hyperparamètres

Cette annexe détaille les hyperparamètres utilisés pour entraîner XGBOOST et NN sur les deux jeux de données.

XGBOOST

Sur MAR, XGBOOST est utilisé avec ses paramètres par défaut, à l'exception du nombre d'arbres, fixé à 200. Une perte logistique est utilisée et le taux d'échantillonnage négatif est fixé à 50 (Tableau 5).

TABLE 5 – XGBOOST Hyperparamètres pour MAR

Nombre d'arbres	200
Fonction de perte	Logistique
Ratio d'échantillonnage négatif	50

Sur JOB, XGBOOST est utilisé avec les hyperparamètres indiqués dans le tableau 6. Les autres hyperparamètres sont fixés à leur valeur par défaut.

TABLE 6 – XGBOOST Hyperparamètres pour JOB

col_sample_bytree	0.6
eta	0.075
gamma	0.85
max_depth	12
min_child_weight	1
subsample	0.9
Nombre d'arbres	400
Fonction de perte	Logistique
Ratio pour l'échantillonnage négatif	50

NN

Le paramètre de marge η dans la fonction de perte est fixé à 1 dans toutes les expériences.
Sur MAR, NN est utilisé avec les hyperparamètres du Tableau 7. Dans chaque batch, 10 paires négatives sont sélectionnées uniformément pour chaque positive.
Sur JOB, le modèle NN utilise une architecture définie pour le marché de l'emploi, avec trois modules neuronaux. Le premier module "géographique" prend les coordonnées latitude et longitude standardisées d'une commune, et produit une représentation de dimension 50 en utilisant deux couches cachées de taille 100 et des fonctions d'activation *tanh*. Le module est entraîné pendant 100 *epochs* avec l'algorithme d'optimisation Adam et un taux d'apprentissage initial de 10^{-4}. Les exemples négatifs sont choisis en prenant des items plus éloignés (géographiquement) que l'item positif. Le deuxième module "compétences" prend les compétences standardisées et les réduit à une représentation de dimension 100 avec une couche cachée de taille 200 et une fonction d'activation RELU. La matrice de similarité est diagonale, et le module est entraîné pendant 100 *epochs* avec l'algorithme d'optimisation Adam et un taux d'apprentissage initial 10^{-4}. Le troisième module "autre" prend les autres variables, avec une couche cachée de taille 500 et produit une représentation de dimension 200. Le module est entraîné pendant 100 *epochs* avec l'algorithme d'optimisation Adam et un taux d'apprentissage initial 10^{-4}. L'architecture d'ensemble est initialisée en utilisant les poids pré-entraînés des trois modules précédents, et la matrice de similarité A est contrainte d'être diagonale par blocs. Le module est ensuite entraîné pendant 35 *epochs* avec l'algorithme d'optimisation Adam et un taux d'apprentissage initial 10^{-4}, en utilisant des exemples négatifs choisis uniformément. Les autres hyperparamètres sont détaillés dans la Table 6.

TABLE 7 – NN : Hyperparamètres pour MAR

Couche 1	*tanh*, dim. 300
Embedding	*tanh*, dim. 300
Algorithme d'apprentissage	Adam
Taux d'apprentissage	0.001
Epochs	300
Taille de *batch*	64
Ratio d'échantillogange négatif (par *epoch*)	10

Annexe C : Résultats complets

La figure 5 présente les résultats de toutes les méthodes dans le plan 2D recall@10, congestion@10. La figure 6 présente des courbes de Lorenz pour illustrer la qualité de recommandation. Enfin, le tableau 4 présente les temps de calcul pour chaque méthode. Les temps d'entraînement pour XGBOOST et NN sont limités à respectivement 2 heures et 30 minutes. Le temps du transport optimal augmente avec la diminution de ε jusqu'à environ 10 minutes pour $\varepsilon = 0,01$. Le coût le plus important vient du calcul des recommandations avec XGBOOST et γ^{XGB} en raison de la nécessité de calculer des variables d'adéquation jointe pour toutes les paires (utilisateur, item).

FIGURE 2 – $\varepsilon = 100$

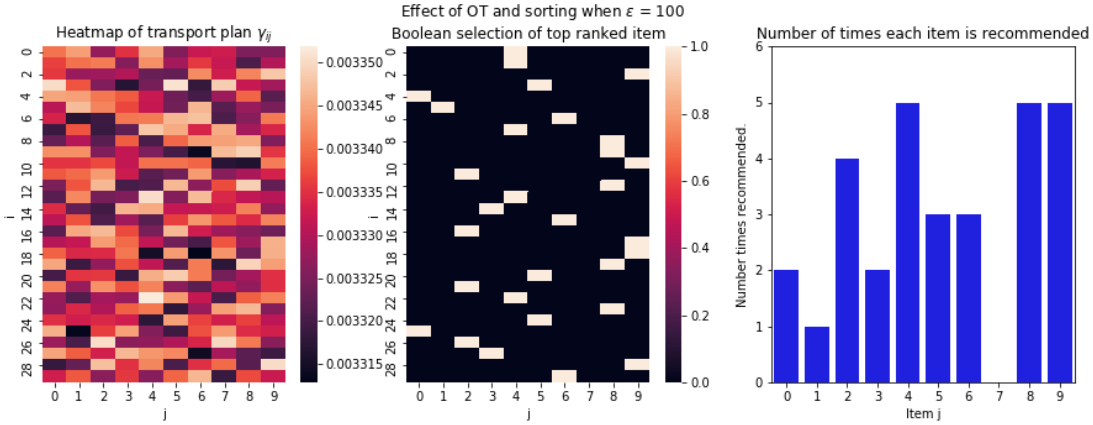


FIGURE 3 – $\varepsilon = 0.1$

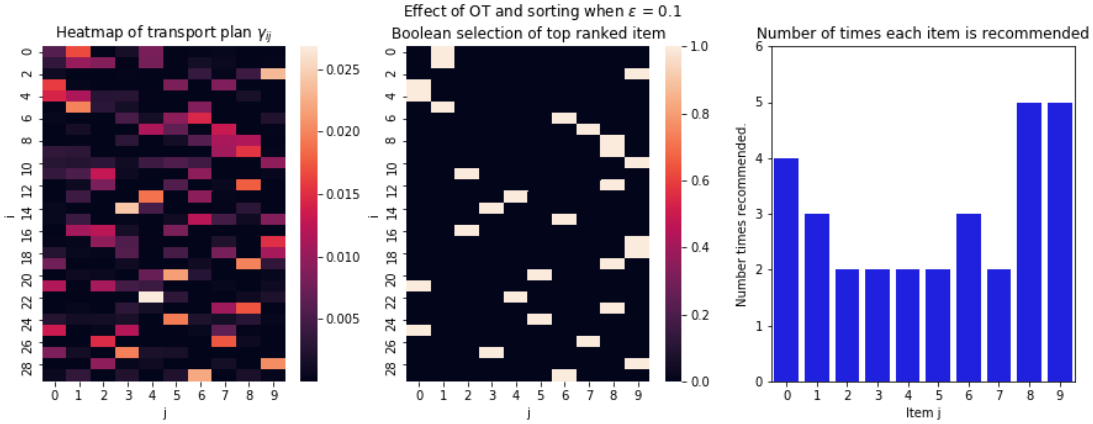


FIGURE 4 – $\varepsilon = 0.01$

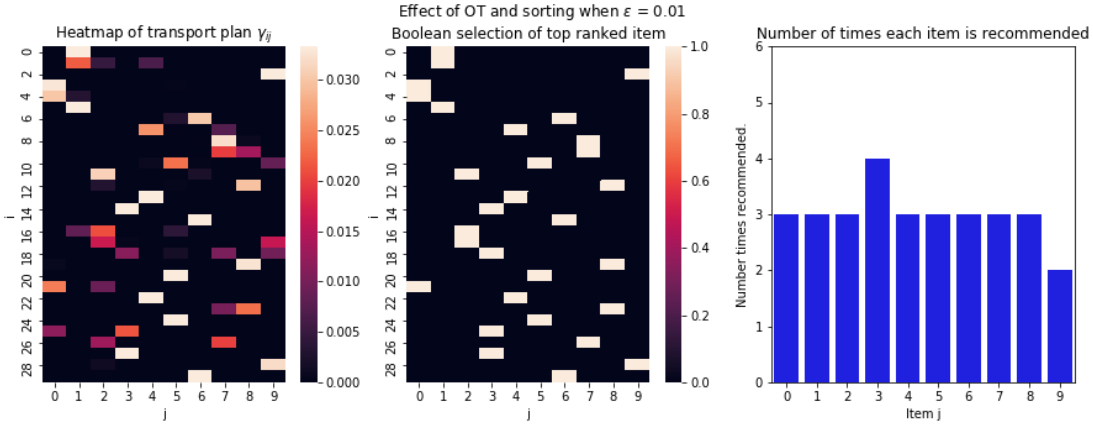


FIGURE 5 – Front de Pareto congestion (-Congestion@10) - précision de la recommandation (Recall@10), ensemble de données JOB

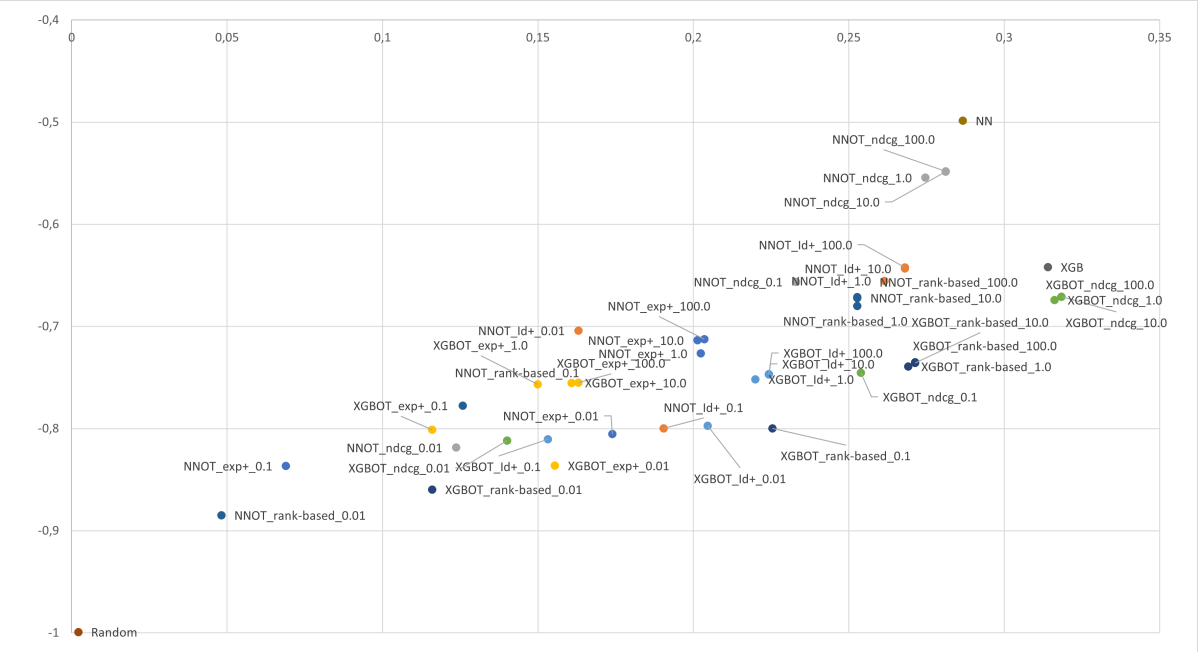
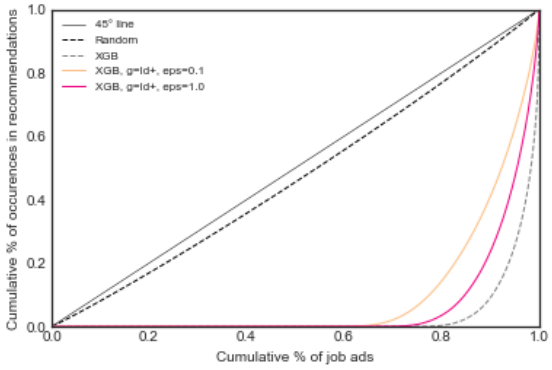
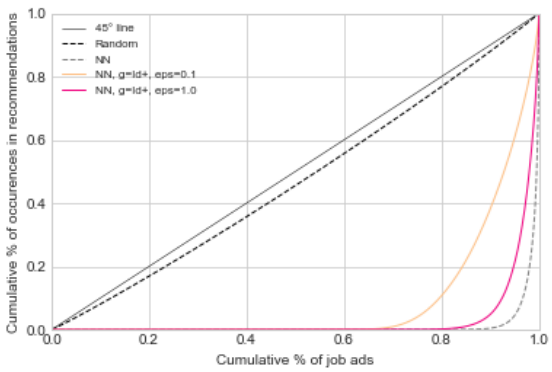


FIGURE 6 – Courbes de Lorenz calculées sur les recommandations du Top10



(a) XGB sur le jeu de données JOB



(b) NN sur le jeu de données JOB

TABLE 8 – Hyperparamètres - NN (JOB)

Sous-module géographique	
Couche 1	*tanh*, dim. 100
Couche 2	*tanh*, dim. 100
Embedding	*tanh*, dim. 50
Algorithme d'apprentissage	Adam
Taux d'apprentissage	0.0001
Epochs	100
Taille de *batch*	32
Sous-module compétences	
Couche 1	ReLU, dim. 200
Embedding	*tanh*, size = 100
Algorithme d'apprentissage	Adam
Taux d'apprentissage	0.0001
Epochs	100
Taille de *batch*	32
Module "général"	
Couche 1	ReLU, dim. 500
Embedding	*tanh*, dim. 200
Algorithme d'apprentissage	Adam
Taux d'apprentissage	0.0001
Epochs	100
Taille de *batch*	256
Apprentissage à chaud	
Structure bloc-diagonale	Oui
Epochs	10 / 25
Algorithme d'apprentissage	Adam
Taux d'apprentissage	0.0001 / 0.00001
Taille de *batch*	256