RankMerging: Apprentissage supervisé de classements pour la prédiction de liens dans les grands réseaux sociaux

Lionel Tabourier*,**, Anne-Sophie Libert***, Renaud Lambiotte***

*Sorbonne Universités, UPMC Univ. Paris 06, UMR 7606, LIP6
**CNRS, UMR 7606, LIP6, Paris, France
***naXys, Université de Namur, Namur, Belgique

Résumé. Trouver les liens manquants dans un grand réseau social est une tâche difficile, car ces réseaux sont peu denses, et les liens peuvent correspondre à des environnements structurels variés. Dans cet article, nous décrivons *RankMerging*, une méthode d'apprentissage supervisé simple pour combiner l'information obtenue par différentes méthodes de classement. Afin d'illustrer son intérêt, nous l'appliquons à un réseau d'utilisateurs de téléphones portables, pour montrer comment un opérateur peut détecter des liens entre les clients de ses concurrents. Nous montrons que *RankMerging* surpasse les méthodes à disposition pour prédire un nombre variable de liens dans un grand graphe épars.

1 Introduction et contexte

Le problème de la prédiction de liens tel qu'il est formulé dans l'article de Liben-Nowell et al. (2007) peut être compris comme une tâche de classification binaire. Des outils classiques d'apprentissage tels que les arbres de classification, les SVM ou les réseaux de neurones ont été utilisés pour le résoudre sur des réseaux biologiques et de collaboration (Pujari et al. (2012)). Cependant, ces méthodes ne permettent pas à l'utilisateur de faire varier le nombre de prédictions selon ses besoins. Pour ce faire, il est possible de calculer un score pour chaque paire de nœuds, corrélé à la probabilité d'existence d'un lien entre ces nœuds, on obtient alors un classement, et l'utilisateur effectue la prédiction en sélectionnant les T paires les mieux classées. Le score peut être basé sur la structure connue du réseau, mais également sur d'autres sources d'information : par exemple les attributs des nœuds, la dynamique des contacts ou la localisation géographique (Scellato et al. (2011)). Pour combiner les informations capturées par différents scores, on utilise des méthodologies d'apprentissage de classements. Parmi les méthodes à disposition, certaines sont non-supervisées et peuvent être vues comme des méthodes de consensus, telles que celles décrites dans Dwork et al. (2001). Il existe également des méthodes supervisées : une solution consiste à se ramener à un problème de classification en effectuant une *transformation deux-à-deux*, plutôt que de considérer des éléments à ordonner, on examine des couples d'éléments dont on cherche à dire lequel doit être classé audessus de l'autre (Herbrich et al. (1999)). Malheureusement, cette méthode n'est pas adaptée à de grands réseaux où le nombre d'éléments à classer est élevé. Plus généralement, la plupart des méthodes d'apprentissage de classements ont été créées pour des tâches de recherche d'information, où l'on souhaite une précision élevée sur un petit nombre d'éléments (e.g., Burges

et al. (2011)). Nous souhaitons ici au contraire pouvoir fixer le nombre de prédictions, quitte à perdre en précision, pour prédire des liens non-observés dans de grands réseaux sociaux, et nous définissons dans ce but une méthodologie simple mais efficace d'apprentissage supervisé.

Données. La validité de cette méthode est testée sur un jeu de données d'appels téléphoniques anonymisés, correspondant à un mois de communications entre environ un million de clients d'un opérateur européen. Le réseau est représenté par un graphe pondéré dont les sommets sont les utilisateurs et les appels des liens. Après prétraitement des données, celui-ci compte 1.131.049 sommets, 795.865 liens et 10.934.277 appels. Le nombre d'appels entre i et j est le poids $w(i,j)$ du lien, et nous appelons activité totale W d'un nœud la somme des poids des liens qui lui sont adjacents. Sachant qu'un opérateur connaît les interactions entre ses propres clients (nœuds A) et les clients de ses concurrents (nœuds B), on simule la situation où il cherche à découvrir les liens existants entre les clients de ses concurrents ($B-B$). Nous construisons un graphe d'apprentissage avec les seuls sommets A en séparant cet ensemble en deux sous-ensembles A_1 et A_2. Pendant l'étape d'apprentissage, les liens $A_2 - A_2$ sont supposés inconnus et on cherche à les découvrir à l'aide des interactions $A_1 - A_1$ et $A_1 - A_2$. Au cours de la phase de test, toutes les interactions $A - A$ ainsi que les $A - B$ sont supposées connues, et nous cherchons à découvrir les liens $B - B$. Les utilisateurs sont affectés aléatoirement aux ensembles A_1, A_2 et B selon les proportions 50, 25, 25%. On obtient alors un réseau d'apprentissage $\mathcal{G}_{learn}$ et un réseau de test $\mathcal{G}_{test}$ [1].

2 Classements non-supervisés

Scores de classement. Le but de ce travail est de montrer comment notre méthode permet de combiner des informations issues de différentes sources, nous ne présentons ici que quelques caractéristiques structurelles permettant d'associer à chaque paire de nœuds un score à partir duquel est construit un classement. On trouve dans la littérature beaucoup d'indices, nous en avons choisi quelques classiques et proposons de les généraliser à des réseaux pondérés. Certaines de ces caractéristiques sont dites *locales* car elles ne considèrent que des paires de nœuds à distance au plus 2. On appelle $\mathcal{N}(i)$ l'ensemble des voisins du sommet i :

- nombre de voisins communs (*CN*) : $s_{CN_w}(i,j) = \sum_{k \in \mathcal{N}(i) \cap \mathcal{N}(j)} w(i,k).w(j,k)$
- indice d'Adamic-Adar (*AA*) : $s_{AA_w}(i,j) = \sum_{k \in \mathcal{N}(i) \cap \mathcal{N}(j)} \frac{1}{log(W(k))}$
- indice de Jaccard (*Jacc*) : $s_{Jacc_w}(i,j) = \sum_{k \in \mathcal{N}(i) \cap \mathcal{N}(j)} (w(i,k) + w(j,k)) \,/\, (W(i) + W(j))$.

D'autres caractéristiques sont dites *globales* car elles sont calculées sur l'ensemble de la structure du réseau et permettent de classer des paires de nœuds distantes :

- indice de Katz (*Katz*) : calculé à l'aide du nombre de chemins de longueur l de i à j (au sens d'un multigraphe pour un réseau pondéré), noté $\nu_{ij}(l)$, selon l'expression : $s_{Katz}(i,j) = \sum_{l=1}^{\infty} \beta^l \nu_{ij}(l)$, où β est un paramètre ajustable.
- indice *Random Walk with Restart* (*RWR*), dérivé de l'algorithme *PageRank* : considérant un marcheur aléatoire partant d'un sommet i, allant de k à k' avec une probabilité

1. $\mathcal{G}_{learn}$ compte 48.911 sommets, 597.538 arêtes, il y a 49.731 liens $A_2 - A_2$ à découvrir ; $\mathcal{G}_{test}$ compte 1.131.049 sommets, 746.202 liens, et il y a 49.663 liens $B - B$ à découvrir.

$p.w(k,k')/W(k)$ et revenant en i avec une probabilité $1-p$, l'indice est la probabilité pour que ce marcheur soit en j dans l'état stationnaire de la marche.

- attachement préférentiel (*PA*), basé sur l'observation dans les réseaux sociaux que les nœuds de fort degré tendent à créer plus de nouveaux liens : $s_{PA_w}(i,j) = W(i).W(j)$ [2].

Enfin nous utilisons une méthode pour agréger des classements dans le but d'améliorer la qualité de la prédiction. Il s'agit de la *méthode de Borda*, initialement définie pour obtenir un consensus dans un système de vote. On affecte à chaque paire un score correspondant à la somme sur l'ensemble des classements des nombres de paires moins bien classées, soit : $s_B(i,j) = \sum_{\kappa=1}^{\alpha} |r_\kappa| - r_\kappa(i,j)$, où $|r_\kappa|$ est le nombre d'éléments classés dans r_κ [3].

Résultats. Certaines métriques usuelles d'évaluation des problèmes de classification, comme la courbe ROC, ne sont pas adaptées au problème, en raison de la forte asymétrie des classes. En effet, les graphes étant peu denses, on s'attend à ce que le taux de faux positif soit élevé dans une large gamme de prédictions et rende l'observation de la courbe ROC peu instructive. Comme nous souhaitons pouvoir ajuster le nombre de prédictions pour trouver un bon compromis entre précision (**Pr**) et rappel (**Rc**), nous visualisons les résultats à l'aide du **F**-score et des courbes **Pr**-**Rc**.

Nous traçons sur la Figure 1 les résultats obtenus sur le graphe d'apprentissage $\mathcal{G}_{learn}$ pour prédire les liens $A_2 - A_2$ en utilisant certains des scores définis précédemment (pas la totalité pour une question de lisibilité). On voit que l'allure des courbes de **F**-score varient significativement d'un indice à un autre, ce qui indique que chaque score permet de détecter des liens différents, ce dont nous chercherons à tirer parti dans *RankMerging*. Comme on peut s'y attendre pour une méthode de consensus, la méthode de Borda améliore les performances de la classification, en particulier la précision sur la prédiction des paires les mieux classées. Étant donnée la difficulté de la tâche, la précision est peu élevée en moyenne : lorsque **Rc** est plus grand que 0.06, **Pr** est inférieur à 0.3 pour toutes les méthodes. Nous n'utilisons que des scores structurels, rendant improbable la prédiction de liens entre paires de sommets éloignées dans le graphe, le rappel est donc limité à des valeurs relativement faibles, car augmenter le nombre de prédictions diminuerait drastiquement la précision.

3 Méthode *RankMerging*

Les différentes méthodes de classements précédemment citées ne sont pas sensibles aux mêmes types d'information structurelle. Nous décrivons une méthode générique d'apprentissage supervisé qui permet d'agréger les classements en tirant partie de la complémentarité des sources d'information. Celle-ci ne nécessite pas qu'une paire soit bien classée selon chaque critère, comme pour une règle de consensus, mais qu'elle soit bien classée selon au moins un. La procédure est dénommée *RankMerging*, une implémentation et un guide utilisateur sont à disposition sur `http://lioneltabourier.fr/program.html`.

2. *Katz* et *RWR* sont calculés à l'aide de sommes infinies, que nous approximons à l'aide des quatre premiers termes afin de réduire le coût du calcul. La forte asymétrie des classes tend à diminuer les performances de l'indice *PA*, nous limitons la prédiction avec cet indice aux paires de sommets distantes d'au plus 3.

3. Pour que cette méthode ne biaise pas en faveur des prédicteurs qui classent un grand nombre d'éléments, on considère qu'une paire classée dans r_κ mais pas dans $r_{\kappa'}$ est aussi classée dans $r_{\kappa'}$, avec un rang équivalent à toutes les autres paires non-classées et en-dessous de toute paire classée. Pour plus de détails, voir Dwork et al. (2001).

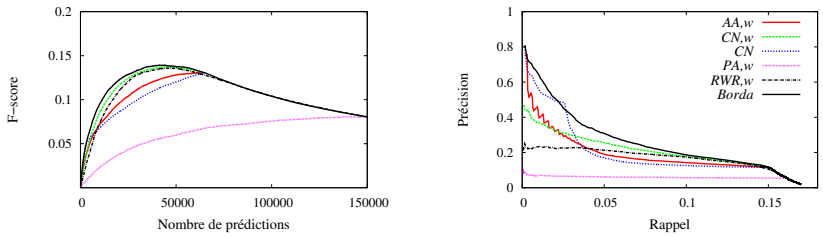

FIG. 1 – *Résultats obtenus sur le graphe d'apprentissage pour différents indices structurels. Gauche : **F**-score en fonction du nombre de prédictions. Droite : **Pr** en fonction de **Rc**.*

Principe de la méthode. Nous disposons d'α classements : $r_1, ..., r_\alpha$, obtenus par des méthodes non-supervisées, qui sont les entrées de l'algorithme. Durant la phase d'apprentissage, nous savons pour chaque paire si le lien existe ou non, c'est-à-dire s'il s'agit d'un vrai ou d'un faux positif (*tp* ou *fp*). Notre but est de combiner les classements pour maximiser le nombre de *tp* à nombre de prédictions T fixé. Soit ϕ_i, le curseur utilisé pour parcourir le classement r_i : les liens prédits sur $\mathcal{G}_{learn}$ sont situés entre les rangs 1 et ϕ_i de chaque classement. Nous parcourons tous les classements à la fois, et notre but est de calculer les valeurs optimales de $\phi_1, ..., \phi_\alpha$ pour que le nombre total de *tp* soit maximum à T fixé. Notons qu'il faut prendre en compte les liens déjà vus pour prédire les suivants, car chaque paire ne peut apparaître qu'une fois dans le classement final. Cela rend le problème d'optimisation difficile à résoudre exactement, nous définissons alors l'heuristique qui suit.

L'idée centrale est de déterminer à chaque pas de l'algorithme le classement qui prédit le nombre le plus élevé de *tp* dans les prochaines étapes. Dans ce but nous définissons une *fenêtre* $\mathcal{W}_i$ pour chaque classement : c'est l'ensemble des g liens non-prédits classés à partir du rang ϕ_i dans r_i [4]. On appelle *qualité* χ_i le nombre de tp dans $\mathcal{W}_i$. À chaque étape, le classement r_i dont la fenêtre a la qualité la plus élevée est sélectionné (en cas d'égalité, on choisit aléatoirement), et on ajoute alors la paire classée au rang ϕ_i de r_i au classement de sortie de la phase d'apprentissage (r_M^L). On met à jour ϕ_i et le curseur de fin de la fenêtre, de manière à ce que chaque $\mathcal{W}_i$ contienne toujours exactement g paires, puis on met à jour les qualités χ_i. Au cours du processus, on enregistre à chaque pas les valeurs des curseurs $\phi_1 ... \phi_\alpha$, qui indiquent la contribution de chaque classement au classement agrégé. Cette procédure est itérée jusqu'à ce que le classement agrégé r_M^L contienne le nombre de paires T fixé par l'utilisateur. Cet algorithme présente l'intérêt majeur de ne parcourir qu'une seule fois chaque classement, soit une complexité en $O(\alpha N)$ pour α classements de taille N (on note que $N \leq T$).

La phase de test consiste à agréger les classements obtenus sur le réseau $\mathcal{G}_{test}$ en utilisant les paramètres $\phi_1 ... \phi_\alpha$ appris durant la première phase. L'implémentation pratique est simple : à chaque étape on regarde quel classement a été choisi à l'étape correspondante de l'apprentissage, et on sélectionne la paire la mieux classée du classement correspondant pour le graphe de test. Si celle-ci n'est pas déjà dans le classement agrégé de la phase de test (r_M^T), elle y est ajoutée, sinon on recherche la paire suivante du classement r_i, et ainsi de suite jusqu'à ce qu'on ait trouvé une paire qui ne soit pas dans r_M^T [5].

4. g est l'unique paramètre de *RankMerging* fixé par l'utilisateur.
5. Notons que les nombres de paires classées peuvent varier entre $\mathcal{G}_{learn}$ et $\mathcal{G}_{test}$: respectivement n_L et n_T.

Protocole expérimental et résultats. Nous évaluons les performances de *RankMerging* en les comparant à des techniques existantes. En premier lieu, nous confrontons les résultats à la méthode non-supervisée de Borda, vue précédemment. Nous comparons aussi à des méthodes de classification supervisées, même si celles-ci ne sont pas conçues pour varier le nombre T de prédictions possibles [6]. Nous utilisons les implémentations proposées dans la boîte à outils Python *scikit learn* (`scikit-learn.org`) des méthodes suivantes : les k-plus proches voisins (*NN*), les arbres de classification (*CT*) et *AdaBoost* (*AB*), en faisant varier les paramètres de manière à obtenir plusieurs points dans l'espace **Pr**-**Rc** [7].

Suivant la description de la méthode faite précédemment, nous mesurons les ϕ_i obtenus sur $\mathcal{G}_{learn}$ pour découvrir les liens $A_2 - A_2$, puis nous les utilisons pour agréger les classements obtenus sur $\mathcal{G}_{test}$ pour prédire les liens $B - B$, en utilisant le facteur d'échelle $f \approx 1.5$. La sélection des caractéristiques pertinentes n'est pas problématique ici. En effet, l'utilisateur peut agréger autant de classements qu'il le souhaite car, d'une part, l'addition d'un nouveau classement a un faible coût computationnel, et d'autre part, la méthode est conçue pour qu'un classement n'apportant pas d'information nouvelle soit simplement ignoré pendant l'agrégation. Le paramètre g de l'algorithme est ici fixé par une simple extrapolation : on mesure la valeur de g qui permet de maximiser la qualité de l'agrégation sur $\mathcal{G}_{learn}$, et on emploie la même pour l'agrégation sur $\mathcal{G}_{test}$ (ici $g = 200$).

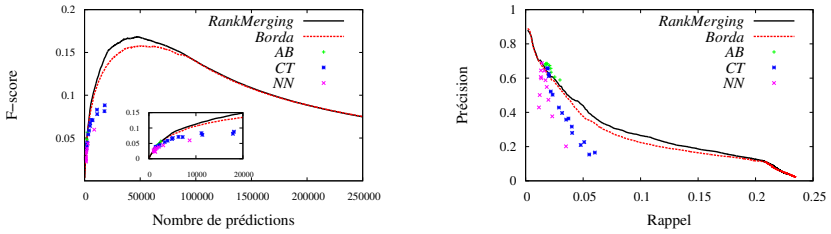

FIG. 2 – *Résultats obtenus sur le graphe de test ($g = 200$) comparé à plusieurs benchmarks. Gauche : **F**-score en fonction du nombre de prédictions. Droite : **Pr** en fonction de **Rc**.*

Sur la Figure 2, nous traçons le **F**-score et la courbe précision-rappel obtenus pour *RankMerging* ($g = 200$), en agrégeant les classements des indices suivants : AA_w, CN_w, CN, $Jacc_w$, $Katz_w$ ($\beta = 0.01$), PA_w, RWR_w ($p = 0.8$) et la méthode de Borda appliquée à ces sept indices. *RankMerging* permet d'améliorer les prédictions : la mesure de l'aire sous la courbe **Pr**-**Rc** indique une amélioration de 8.3% par rapport à la méthode de Borda [8]. On pouvait s'y attendre, dans la mesure où *RankMerging* est une méthode supervisée et utilise l'information contenue dans le consensus de Borda. En fait, la méthode est conçue pour que *n'importe quel*

Nous devons alors prendre en compte un facteur d'échelle $f = n_T/n_L$: pour un nombre donné de prédictions T sur le graphe d'apprentissage, la prédiction sur le graphe de test sera optimale avec les mêmes proportions de chaque classement pour $[T.f]$ prédictions. Autrement dit, à l'étape n du processus d'agrégation sur les données de test, on utilise les valeurs des $\phi_1 ... \phi_\alpha$ enregistrées pendant l'étape $s = [n/f]$ de la phase d'apprentissage.

6. Il serait plus satisfaisant d'utiliser des méthodes supervisées de classement, telles que *RankSVM*, mais comme dit en introduction, cela n'est pas envisageable pour les tailles de classements que nous considérons.

7. Ces paramètres sont la valeur de k pour *NN*, la taille des feuilles pour *CT*, le nombre d'arbres pour *AB*.

8. Pour les valeurs de précision élevées, il n'y a en revanche pas de gain substantiel, on l'utilisera donc plutôt lorsque l'on souhaite faire varier la précision et le rappel à des valeurs intermédiaires.

classement non-supervisé puisse être agrégé sans perte de performance. Nous vérifions cela en pratique en retirant un à un les classements et en constatant que l'amélioration ne fait que décroître, et ce quel que soit l'ordre dans lequel on retire les différents classements [9]. En ce qui concerne les méthodes de classifications supervisées, nous pouvons constater que leurs performances sont élevées uniquement pour un faible nombre de prédictions (inférieur à 2000), mais ces méthodes n'étant pas conçues pour faire varier le nombre de prédictions, elles produisent de très faibles performances en dehors de leur domaine optimal.

4 Bilan et perspectives

Nous avons présenté *RankMerging*, une méthodologie d'apprentissage supervisé pour agréger des classements obtenus par des méthodes non-supervisés dans le but d'améliorer la prédiction de liens. Cette méthode simple à mettre en œuvre a un coût computationnel linéaire, et permet de multiplier les sources d'information sans perte de performance, elle est donc appropriée pour la prédiction de liens dans les grands réseaux sociaux. Ici, nous n'avons utilisé que la structure du graphe comme source d'information. Une suite logique à cette étude serait alors de considérer d'autres sources, telles que le profil des utilisateurs (âge, lieu de travail...) ou la dynamique des interactions.

Références

Burges, C. et al. (2011). Learning to rank using an ensemble of lambda-gradient models. *JMLR - Proc. Track 14*.

Dwork, C. et al. (2001). Rank aggregation methods for the web. In *WWW'01*. ACM.

Herbrich, R. et al. (1999). Large margin rank boundaries for ordinal regression. *NIPS'99*.

Liben-Nowell, D. et al. (2007). The link-prediction problem for social networks. *JASIST 58*(7).

Pujari, M. et al. (2012). Supervised rank aggregation approach for link prediction in complex networks. In *WWW'12 Companion*. ACM.

Scellato, S. et al. (2011). Exploiting place features in link prediction on location-based social networks. In *SIGKDD'11*. ACM.

Summary

Uncovering missing links in social networks is a difficult task because of their sparsity, and because links may represent different types of relationships, characterized by different structural patterns. In this paper, we define a simple supervised learning-to-rank framework, called *RankMerging*, which aims at combining information provided by various unsupervised rankings. As an illustration, we apply the method to the case of a cell phone service provider, which aims at discovering links among contractors of its competitors. We show that our method substantially improves the performance of the available classification methods.

9. Ces résultats sont disponibles dans la version longue de l'article, *RankMerging : Learning to rank in large-scale social networks*, *DyNakII (PKDD'14 workshop)*.