Filtrage et Recommandation sur les Réseaux Sociaux

Mohammed Ryadh Dahimene

le cnam

ÉCOLE DOCTORALE INFORMATIQUE, TÉLÉCOMMUNICATION ET ÉLECTRONIQUE (PARIS)

ÉQUIPES ISID/VERTIGO – LABORATOIRE CEDRIC

THÈSE présentée par :

Mohamed Ryadh DAHIMENE

soutenue le : **8 Décembre 2014**

pour obtenir le grade de : **Docteur du Conservatoire National des Arts et Métiers**

Discipline/ Spécialité : Informatique

Filtrage et Recommandation sur les Réseaux Sociaux

THÈSE dirigée par :

M. DU MOUZA Cédric	MCF-HDR, CNAM
Mme. CONSTANTIN Camelia	MCF, UPMC

RAPPORTEURS :

M. DEFUDE Bruno	Professeur, TELECOM SudParis
Mme. VOISARD Agnès	Professeur, Freie Universität Berlin

JURY :

M. DEFUDE Bruno	Professeur, TELECOM SudParis
Mme. VOISARD Agnès	Professeur, Freie Universität Berlin
M. CAUTIS Bogdan	Professeur, Université Paris-Sud
Mme. COMYN-WATTIAU Isabelle	Professeur, CNAM
M. VODISLAV Dan	Professeur, Université de Cergy-Pontoise
M. DU MOUZA Cédric	MCF-HDR, CNAM
Mme. CONSTANTIN Camelia	MCF, UPMC

Conservatoire National des Arts et Métiers
École Doctorale Informatique, Télécommunication et Électronique (Paris)
Laboratoire CEDRIC – Équipes ISID/VERTIGO

THÈSE

présentée par

Mohamed Ryadh Dahimene

pour obtenir le grade de
Docteur du Conservatoire National des Arts et Métiers

Discipline/Spécialité : Informatique

Filtrage et Recommandation sur les Réseaux Sociaux

Devant le jury composé de :

Monsieur	Bruno Defude	Rapporteurs
Madame	Agnès Voisard	
Monsieur	Bogdan Cautis	Examinateurs
Madame	Isabelle Comyn-Wattiau	
Monsieur	Dan Vodislav	
Monsieur	Cédric Du Mouza	Directeur de thèse
Madame	Camelia Constantin	Encadrante de thèse

Paris, le 8 Décembre 2014

Je dédie ce travail à la mémoire de trois hommes,
à mon grand-père Rachid Benhaddad
à Karim Benabadji
et à Michel Scholl

Remerciements

Je tiens d'abord à remercier Michel Scholl [1] de m'avoir accepté en thèse un beau jour de Mai 2010 et d'avoir mis ce travail sur les rails.

Aussi, je remercie profondément Cédric et Camelia de m'avoir accompagné durant cette aventure en guidant mes pas, en m'accordant leur confiance et en orientant notre collaboration durant ces années. Sans leur implication, ce travail n'aurait pas vu le jour.

Ma gratitude s'adresse également à Monsieur Bruno DEFUDE ainsi qu'à Madame Agnès VOISARD pour avoir accepté de rapporter cette thèse. Je remercie aussi Monsieur Bogdan CAUTIS, Madame Isabelle COMYN-WATTIAU et Monsieur Dan VODISLAV de bien vouloir participer à mon jury.

J'adresse mes remerciements aussi à tous les membres des équipes ISID et VERTIGO qui m'ont accueilli et avec qui j'ai eu le plaisir d'échanger pendant ma thèse sans oublier aussi les membres du LIP6. Merci à Bernd AMANN, Jacky AKOKA, Michel CRUCIANU, Hammou FADILI, Marin FERECATU, Fayçal HAMDI, Nadira LAMMARI, Elisabeth METAIS, Nicolas PRAT, Christophe PICOULEAU, Philippe RIGAUX, Samira SI-SAID CHERIFI et Nicolas TRAVERS.

Je remercie aussi très chaleureusement mes collègues du CNAM mais aussi d'ailleurs pour avoir été là pendant ces années. Merci à Amina, Andrés, Anh, Damien, Feten, Houda, Lydia, Nabil, Nelly, Pascal, Quentin, Rodney, Sarah et enfin Zeinab pour son aide et ses conseils.

Les mots ne suffisent pas pour remercier mes parents, ma sœur, ma

1. http ://cedric.cnam.fr/∼scholl/hommage

grand-mère et toute ma famille pour leur soutien et pour avoir toujours cru en moi. Ce travail est aussi le fruit de leur patience. Merci.

Enfin, merci aussi à Darine et un clin d'œil particulier aux amis Ahmed, Chakib, Chemsou, Elkindi, Mebarek, Malek et Salim et aux "Saint-Jacquois" : Clau, Gabi, Lee, Manu, Vic et Paul.

Résumé

Ces dernières années, le contenu disponible sur le Web a augmenté de manière considérable dans ce qu'on appelle communément le Web social. Pour l'utilisateur moyen, il devient de plus en plus difficile de recevoir du contenu de qualité sans se voir rapidement submergé par le flot incessant de publications. Pour les fournisseurs de service, le passage à l'échelle reste problématique. L'objectif de cette thèse est d'aboutir à une meilleure expérience utilisateur à travers la mise en place de systèmes de filtrage et de recommandation. Le filtrage consiste à offrir la possibilité à un utilisateur de ne recevoir qu'un sous ensemble des publications des comptes auxquels il est abonné. Tandis que la recommandation permet la découverte d'information à travers la suggestion de comptes à suivre sur des sujets donnés. Nous avons élaboré MicroFilter un système de filtrage passant à l'échelle capable de gérer des flux issus du Web ainsi que RecLand, un système de recommandation qui tire parti de la topologie du réseau ainsi que du contenu afin de générer des recommandations pertinentes.

Mots-clés : Réseaux Sociaux, Filtrage, Recommandation, Indexation, Microblogging

Abstract

In the last years, the amount of available data on the social Web has exploded. For the average user, it became hard to find quality content without being overwhelmed with publications. For service providers, the scalability of such services became a challenging task. The aim of this thesis is to achieve a better user experience by offering the filtering and recommendation features. Filtering consists to provide for a given user, the ability of receiving only a subset of the publications from the direct network. Where recommendation allows content discovery by suggesting relevant content producers on given topics. We developed MicroFilter, a scalable filtering system able to handle Web-like data flows and RecLand, a recommender system that takes advantage of the network topology as well as the content in order to provide relevant recommendations.

Keywords : Social Networks, Filtering, Recommendation, Indexing, Micro-Blogging

Table des matières

1 Introduction **1**
- 1.1 Contexte . . . 1
- 1.2 Notre Problématique . . . 6
- 1.3 Contributions . . . 6
- 1.4 Organisation de la thèse . . . 8

2 État de l'art **11**
- 2.1 Introduction . . . 11
- 2.2 Un Web Social . . . 12
  - 2.2.1 Caractérisation du phénomène . . . 13
  - 2.2.2 Comportement des utilisateurs . . . 17
  - 2.2.3 La détection d'événements . . . 18
- 2.3 Le Filtrage . . . 19
  - 2.3.1 Stratégies de filtrage . . . 19
  - 2.3.2 Indexation . . . 20
  - 2.3.3 Systèmes de publication/souscription . . . 21
- 2.4 Les systèmes de recommandation . . . 21
  - 2.4.1 Définition . . . 21
  - 2.4.2 Les systèmes basés sur le contenu . . . 22
  - 2.4.3 Le filtrage collaboratif . . . 23
  - 2.4.4 Les systèmes hybrides . . . 24
  - 2.4.5 Recommandations sociales . . . 25
- 2.5 Mesures sur les graphes sociaux . . . 27
- 2.6 Conclusion . . . 29

3 Données issues des réseaux sociaux **31**
- 3.1 Introduction . . . 31
- 3.2 Le Micro-Blogging . . . 32
- 3.3 Modèle de données . . . 33

3.3.1 Exemple de graphe social . . . 34
3.3.2 Le système de micro-blogging . . . 34
3.3.3 Les filtres . . . 35
3.3.4 La notification . . . 35
3.4 L'acquisition de données . . . 36
3.4.1 Le *crawling*, approche et limitations . . . 36
3.4.2 Constitution du jeu de données . . . 39
3.5 Jeu de données pour le Filtrage . . . 40
3.6 Jeu de données pour la recommandation . . . 40
3.7 Conclusion . . . 44

4 Filtrage et indexation **47**
4.1 Introduction . . . 47
4.2 L'indexation des filtres . . . 48
4.3 Structures et modèle analytique . . . 50
4.3.1 Le PFT-index . . . 50
4.3.2 Le PTF-index . . . 51
4.3.3 Le TPF-index . . . 53
4.4 Expérimentations . . . 55
4.4.1 Espace mémoire . . . 56
4.4.2 Temps d'indexation . . . 58
4.4.3 Temps de recherche (matching) . . . 59
4.4.4 Efficacité du filtrage . . . 61
4.5 L'évolution du graphe social . . . 62
4.6 Une structure hybride . . . 63
4.6.1 Occupation mémoire de la structure hybride . . . 65
4.6.2 Temps de matching pour la structure hybride . . . 65
4.7 Conclusion . . . 68

5 Recommandation de comptes **69**
5.1 Introduction . . . 69
5.2 Motivation . . . 70
5.3 Recommandation . . . 70
5.3.1 Le score de recommandation . . . 73
5.3.2 La convergence du calcul des scores . . . 77
5.4 Une estimation efficace des recommandations . . . 79
5.4.1 Vue d'ensemble de l'approche basée sur les *landmarks* 79
5.4.2 Pré-traitement . . . 81
5.4.3 Approximation rapide des recommandations . . . 83
5.5 Expérimentations . . . 86

5.5.1 Implantation 86
5.5.2 Qualité des recommandations 87
5.5.3 Validation par les utilisateurs 91
5.5.4 Approximation du calcul de recommandations 93
5.6 Conclusion 97

6 Conclusion **99**
6.1 Contributions 99
6.2 Perspectives 101

A Captures d'écran de MicroFilter **105**

B Captures d'écran de RecLand **107**

Table des figures

1.1 Nombre d'utilisateurs actifs sur *Facebook* et *Twitter* entre 2010 et le second trimestre 2014 3
1.2 Architecture centralisée de *Twitter* 4
1.3 L'illustration sur la page d'accueil de *Twitter.com* lors des interruptions de service (connue sous le nom de Twitter Fail Whale) 5

3.1 Un graphe social étiqueté (*LSG*) 36
3.2 Visualisation d'un extrait du graphe social obtenu par *crawling* 38
3.3 Twitter400k Degré entrant/sortant 42
3.4 Visualisation d'un graphe social de 10.000 nœuds 43
3.5 Distribution des arcs par *topic* 44

4.1 Le PFT-index 50
4.2 Le PTF-index 52
4.3 Le TPF-index 54
4.4 Espace mémoire / τ 57
4.5 Espace mémoire / N 57
4.6 Temps de matching / τ 60
4.7 Temps de matching / N 61
4.8 Utilisateurs suivant leur nombre de *followers* 64
4.9 Utilisateurs suivant leur nombre moyens de *tweets* 64
4.10 Espace mémoire occupé par rapport à N et au seuil choisi . . 65
4.11 Temps de matching pour la structure hybride par rapport au seuil popularité 66

5.1 Exemple de *LSG* dans un contexte de recommandation 72
5.2 Exemple de recommandation par *landmarks* 80
5.3 Rappel à N 88
5.4 Précision VS rappel 89

5.5 Rappel suivant la stratégie de suppression d'arêtes 90
5.6 Rappel suivant la popularité des topics 91
5.7 Extrait de l'interface du système d'évaluation pour le *topic* `Technology` 92
5.8 Scores de pertinence obtenus par la validation par les utilisateurs 92

A.1 MicroFilter : Vue d'ensemble 105
A.2 MicroFilter : Sélection d'un compte 106
A.3 MicroFilter : Paramètres 106
A.4 MicroFilter : Flux 106

B.1 RecLand : Vue d'ensemble 107
B.2 RecLand : Sélection d'un compte 108
B.3 RecLand : Paramètres 108
B.4 RecLand : Affichage des résultats 109
B.5 RecLand : Top-k générés 109
B.6 RecLand : Affichage des *Tweets* 110

Liste des tableaux

2.1 Dates de lancement et nombre d'utilisateurs des principaux réseaux sociaux 13
2.2 Propriétés topologiques de jeu *Twitter* [Myers et al., 2014] . . 15
2.3 Conventions de notation sur *Twitter* 17

3.1 Description du jeu de données 39
3.2 Exemple de données du LSG pour le filtrage 40
3.3 Propretés topologique du graphe Twitter400k 41

4.1 Table des notations 49
4.2 Tailles des index 56
4.3 Temps d'indexation (en s) 58
4.4 Temps de matching 59
4.5 Nombre de *tweets* remis 62
4.6 Temps moyens d'insertion/suppression (en Nano-Secondes) . 62
4.7 Nombre d'utilisateurs pour différentes valeurs de seuil de popularité 67
4.8 Efficacité des structures d'indexation 67

5.1 Notations 71
5.2 Algorithmes de sélection de *Landmarks* 94
5.3 Temps moyen de sélection et de calcul pour un *landmark* par stratégie de sélection 95
5.4 Comparaison des stratégies de sélection de *landmarks* 96

Chapitre 1

Introduction

1.1 Contexte

Nous vivons à l'ère de l'information. L'avènement d'Internet et des nouvelles technologies de l'information et de la communication a provoqué une véritable rupture qui nous destine à un monde de plus en plus inter-connecté. Il est estimé que la quantité de données produite chaque jour équivaut à 2,5 trillions d'octets (10^{18}) et que plus de 90% de la totalité des données disponibles actuellement ont été produites lors des deux seules dernières années [IBM]. Le 22 Août 2014, le terme "Mégadonnées", équivalent Français du terme *big data*, a fait officiellement son entrée au journal officiel pour caractériser des *"données structurées ou non dont le très grand volume requiert des outils d'analyse adaptés"* [JORF]

Ce phénomène s'est vu amplifié une première fois au début des années 2000 avec l'arrivée du Web 2.0. Cette étape a permis aux utilisateurs de plus en plus nombreux d'accéder plus facilement à la publication de contenu via la démocratisation des blogs et autres systèmes de collaboration tels que les forums de discussion ou les Wikis. Cette transformation a été supportée par une transition technologique qui a vu les pages Web passer d'un format statique (HTML), à un format interactif grâce aux langages de programmation Web dit "dynamiques" tel que PHP, JSP ou ASP.

Peu après, la deuxième accélération fut entamée avec l'adoption à grande

échelle des réseaux sociaux. Ces applications ont depuis leurs débuts eu pour vocation de transférer les interactions sociales du monde réel vers Internet. Axés au départ sur des communautés bien identifiés, tels que les amateurs de musique (ex. *MySpace*[1]) ou des fonctionnalités précises tel que le partage de fichiers multimédia (ex. *YouTube*[2], *Instagram*[3]), ces réseaux ont très vite absorbé la plupart des interactions entre Internautes avec des offres de plus en plus généralistes (ex. *Facebook*[4], *Google+*[5]) englobant la plupart des interactions allant du partage de contenu à la communication.

Fort de ses 1,31 milliard d'utilisateurs actifs (utilisant le réseau au moins une fois par mois), *Facebook*, sans doute le réseau social le plus connu, enregistre des taux de croissance vertigineux. Il est estimé que plus de 829 millions d'utilisateurs utilisent le réseau quotidiennement [Facebook] avec plus de 78% de connexions sur dispositif mobile (Smartphones, tablettes). Cette rapide adoption de la part des Internautes (voir Figure 1.1) s'est vue accompagnée par une déferlante sans commune mesure de données générées par les utilisateurs (*User Generated Content*). Un utilisateur de *Facebook* a en moyenne 130 connexions "d'amitié" et produit une moyenne de 36 publications par mois ce qui a pour résultat d'exposer chaque utilisateur à plus de 1500 nouvelles publications à chaque connexion sur le réseau [DMRa]. En Septembre 2011, le nombre total de photos stockées sur les serveurs de *Facebook* dépassait les 140 milliards, plus de dix mille fois plus que la collection nationale de la bibliothèque du congrès Américain et environ 4% du nombre total de photos jamais prises (estimé à 3,5 billions (10^{12}) en 2011) [Bergen].

Parmi les plateformes sociales ayant affiché les plus fortes croissances, un modèle particulier a émergé ces dernières années : le **Micro-Blogging**. Le micro-blogging tire son nom de sa principale caractéristique, la taille des publications (appelée *tweets*) y est limitée à 140 caractères. Ce modèle simpliste a séduit les Internautes qui l'ont très vite adopté comme moyen pour transmettre et commenter l'actualité. Les utilisateurs peuvent s'abonner à d'autres utilisateurs pour ainsi recevoir leurs publications, ce qui résulte en la présence d'un graphe de relations de **suivi** (*following*).

1. http://www.myspace.com
2. http://www.youtube.com
3. http://www.instagram.com
4. http://www.facebook.com
5. http://plus.google.com

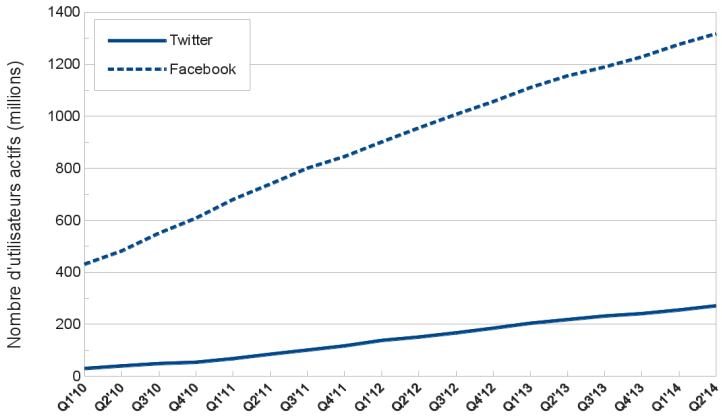

Figure 1.1 – Nombre d'utilisateurs actifs sur *Facebook* et *Twitter* entre 2010 et le second trimestre 2014

Twitter [6], le réseau social leader en matière de micro-blogging, s'est imposé en quelques années comme une source incontournable d'information, en plus de sa fonction de communication. En moins de 8 ans d'existence, *Twitter* à vu sa base d'utilisateurs actifs atteindre les 271 millions et le nombre d'utilisateurs ayant publié au moins un *tweet* dépasser les 570 millions. Ces utilisateurs sont responsables de la production de plus de 500 millions de *tweets* par jour soit une moyenne de 6000 *tweets* par seconde (TPS) avec des pics pouvant atteindre les 143.000 TPS [7]. Chaque semaine, *Twitter* enregistre un million de nouveau comptes crées. En contraste, ce chiffre était de 1000 nouveau comptes par semaine en 2008. Chaque utilisateur de *Twitter* est suivi en moyenne par 208 comptes et en suit à son tour 108 [DMRb]. Il existe cependant une grande disparité sur le réseau avec des comptes très suivis comme le président des États-Unis Barack Obama avec presque 45 millions de suiveurs (followers) ou la chanteuse Katy Perry avec plus 55 millions de followers (chiffres relevés en Août 2014). Cette disparité s'observe aussi en ce qui concerne les fréquences de publication où certains comptes sont très prolifiques comme par exemple Fox News (média Américain) avec ses 135 *tweets* en moyenne par jour ou certains comptes robots (*bots*) qui fonctionnent comme des agrégateurs automatiques de contenu et culminent à plus de 150 *tweets* par jour [Cheng and Evans].

6. http://www.twitter.com
7. https://blog.twitter.com/2013/new-tweets-per-second-record-and-how

Ces taux de croissance et d'utilisation enregistrés sur les plateformes sociales représentent un défi permanent autant pour les fournisseurs de service que pour les utilisateurs finaux. Pour le premiers, il s'agit d'être capable de dimensionner le service afin de faire face au flux sans cesse croissant de *tweets* à transmette. Pour diverses raisons (sécurité, présence de publicité, politiques de contrôle) les réseaux sociaux fonctionnent sur une architecture centralisée. Les publications émises par les utilisateurs sont d'abord transmises au système central qui par la suite s'occupe de les orienter et de notifier les différents utilisateurs destinataires (voir Figure 1.2).

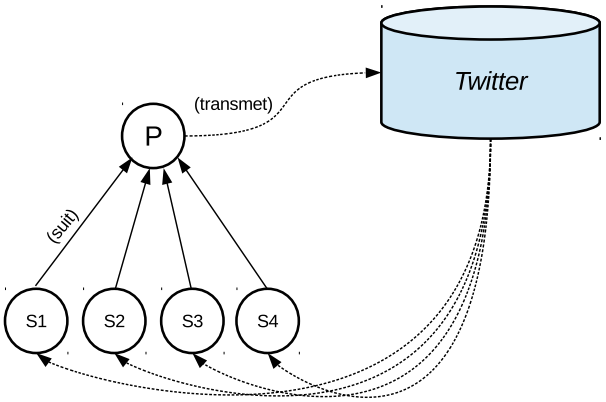

Figure 1.2 – Architecture centralisée de *Twitter*

Avec 200 millions de nouveaux *tweets* par jour en Juillet 2011 et chaque *tweet* devant atteindre les *followers* du compte le publiant, *Twitter* a du transmettre plus de 350 milliard de *tweets* par jour sur son réseau [8]. Cette surcharge de trafic couplée à l'hétérogénéité des tailles de comptes (en nombre de *followers*) avec certains comptes très suivis pose un réel challenge de passage à l'échelle. Les premières années d'exploitation de *Twitter* ont été marquées par de fréquentes interruptions de service dues au fait que la pla-

8. http://latimesblogs.latimes.com/technology/2011/07/twitter-delivers-350-billion-tweets-a-day.html

teforme n'arrivait pas à suivre le flux de *tweets* incessant. L'image affichée sur le page d'accueil de *twitter.com* pendant ces interruptions en est même devenue une icône du Web 2.0 (Figure 1.3)

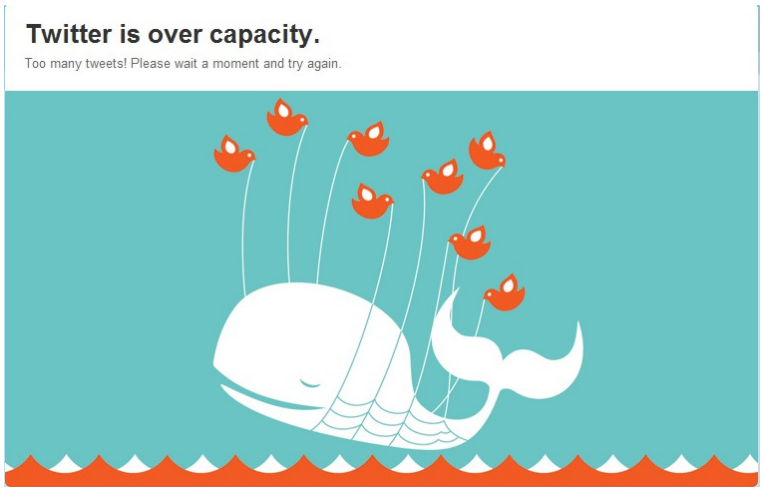

Figure 1.3 – L'illustration sur la page d'accueil de *Twitter.com* lors des interruptions de service (connue sous le nom de Twitter Fail Whale)

Du point de vue des utilisateurs du service, la quantité phénoménale d'information publiée par les comptes suivis a pour effet de diluer l'information de qualité dans les flux sans cesse grandissants de *tweets* reçus. Malgré la présence d'un système de classement basique (voir Section 3.2), il devient vite difficile de distinguer le contenu se rapprochant des centres d'intérêt de l'utilisateur au milieu du flot global d'informations.

Dans un sondage de 2013 mené en Angleterre sur 587 participants et portant sur les réseaux sociaux et la surcharge d'information [Bontcheva et al., 2013], 33,9% des sondés ont répondu positivement à la question : Diriez-vous que vous recevez trop de publications ? et 70,4% trouvent difficile la tache qui consiste à localiser les publications de qualité ou ayant un quelconque intérêt au milieu des autres. Aussi, 66,3% des utilisateurs interrogés ont eu à un moment donné la sensation de ne plus pouvoir suivre ce qu'il se passe sur le réseau car il reçoivent trop de publications. Il en résulte un comportement particulier étudié par [Kwak et al., 2011, Kivran-Swaine et al., 2011] où les utilisateurs ont tendance à suivre un compte temporairement puis à se désabonner rapidement car vite submergés d'informations ne répondant

pas à leurs critères de qualité. Le même sondage de 2013 rapporte que 64,9% des participants se sont déjà désabonné d'un compte car il publiait trop et 44,1% car il ne publiait pas assez de contenu jugé intéressant.

1.2 Notre Problématique

Nous sommes partis du constat que les utilisateurs des réseaux sociaux se retrouvent confrontés en permanence à un véritable déluge d'informations amplifié par le phénomène du Web social ces dernières années. Nous nous intéressons au défi de fournir à l'utilisateur une expérience de qualité sur le Web social. Afin de parvenir à nos fins, nous explorons deux approches complémentaires.

Dans un premier temps, il s'agit de fournir un mécanisme de filtrage dans un contexte de Micro-Blogging. En effet, les plateformes tels que *Twitter* fonctionnent suivant le paradigme du Tout-ou-Rien (*all-or-nothing*) : si un utilisateur A suit un autre utilisateur B sur le réseau, A va recevoir **toutes** les publications de l'utilisateur B. Notre défi consiste à fournir la fonctionnalité de filtrage qui permettrait à A de ne plus recevoir qu'un sous-ensemble des publications générées par B et qui correspondent à son centre d'intérêt.

En second lieu, il s'agit de s'attaquer au problème de la découverte d'information sur le réseau en fournissant des recommandation de comptes à suivre. La tâche consiste en l'occurrence à fournir à A une liste triée par pertinence de comptes à suive par rapport à un certain centre d'intérêt.

L'obstacle majeur à considérer lors de la mise en place des fonctions de filtrage et de recommandation sur les réseaux sociaux est celui du passage à l'échelle. En effet, la taille des graphes sociaux existants couplée à leurs taux de croissance vertigineux rendent la tâche complexe. Il nous faut donc fournir ces fonctionnalités tout en s'avisant à proposer des solutions capables de s'adapter à des tailles et des taux de croissance semblables à ceux présentés en section 1.1.

1.3 Contributions

Dans cette thèse, en réponse à la quantité croissante d'informations à laquelle les utilisateurs des réseaux sociaux se trouvent exposés, nous intro-

duisons des solutions visant à améliorer l'expérience utilisateur axées sur deux stratégies :

— Le filtrage efficace des publications reçues à travers des structures d'indexation spécialement conçues.

— La recommandation intelligente de sources de contenu pertinentes.

Le principal défi à relever concernant ces deux tâches consiste à fournir un service de qualité employable sur les systèmes existants. Avec des chiffres sans cesse croissants, le passage à l'échelle doit constituer une préoccupation permanente qui doit être prise en compte dès la conception et jusqu'à l'implémentation des solutions de filtrage ou de recommandation sur les réseaux sociaux.

Afin d'aborder le problème, nous avons d'abord menée une étude de l'existant présentant tout d'abord une série d'études sur la nature des données sur plateformes sociales. Cette caractérisation nous a permis d'approfondir notre connaissance du domaine. Cet état de l'art traite par la suite des différentes approches adoptées en matière de filtrage et de recommandation sur les réseaux sociaux et positionne notre travail vis-à-vis des travaux cités.

Nous nous sommes ensuite intéressés aux données sociales ainsi qu'à leur acquisition. Nous avons proposé le *LSG* (*Labelled Social Graph*) qui est un modèle de représentation de données spécifique à nos cas d'utilisation et qui capture les intérêts des utilisateurs sous forme de graphe social étiqueté. Un travail d'acquisition et de transformation a ensuite été effectué pour aboutir à des jeux de données représentatifs que nous présentons et détaillons.

Nous avons élaboré par la suite diverses stratégies de filtrage de données basées sur des structures de listes inverses. Ces structures exploitent les caractéristiques intrinsèques aux données issues du monde du Micro-Blogging. Nous avons par la suite proposé un modèle analytique pour envisager leur évolution en temps et en mémoire. Après avoir testé le comportement de ces structures en situation de passage à l'échelle, nous avons proposé une approche hybride qui apporte un compromis et permet d'aboutir à un filtrage efficace.

Par la suite, basé sur le modèle élaboré à l'origine dans un contexte de filtrage, nous avons proposé une stratégie de recommandation de comptes à suivre sur les plateformes de Micro-Blogging. Tirant parti des caractéristiques intrinsèques au graphe d'intérêt, nous avons proposé un score de recommandation doté de trois composantes (*i*) La topologie du graphe : le score favorise les comptes "proches" par rapport à ceux distant dans le graphe (*ii*) La sémantique : le score considère la proximité sémantique sur les chemins entre le nœud (compte) et ses recommandations (*iii*) L'autorité : le score mesure l'influence des comptes à recommander sur un sujet donné et ce de manière locale ainsi que globale. Afin de pouvoir fonctionner sur des graphes de taille importante, nous avons proposé une approche par *landmarks* (comptes élus par le système pour pré-calculer et stocker certains scores) qui fournit des approximations de scores tout en réduisant considérablement les temps nécessaires aux calculs des recommandations. Nous avons ensuite mené une validation expérimentale du système de recommandation proposé face à deux approches connues. Les expériences menées concernent la pertinence des recommandations générés mais aussi du gain induit par les *landmarks* et ainsi la capacité du système à passer à l'échelle. Une validation par des utilisateurs a aussi été menée pour tester la pertinence de notre approche par rapport aux algorithmes concurrents.

1.4 Organisation de la thèse

Cette thèse est organisée comme suit :

Après ce chapitre d'introduction, nous présentons en chapitre 2 l'état de l'art du domaine ciblé. Ce chapitre détaille les différentes avancées récentes en commençant d'abord par la caractérisation du Web "social" qui permet d'acquérir une compréhension plus précise sur la nature et les spécificités du phénomène. Nous y présentons par la suite les différentes approches qui ont été élaborés afin d'introduire la notion de filtrage d'information pour les utilisateurs des réseaux sociaux avec une mise en avant de leur points forts ainsi que de leur limitations. Par la suite, cette étude de l'existant se concentre sur le domaine de la recommandation en présentant d'abord les grandes approches adoptés pour la recommandation de contenu pour se recentrer par la suite sur les solutions proposées dans le contexte des réseaux sociaux.

Le chapitre 3 a pour but de détailler les caractéristiques des données

issues des plateformes sociales de type Micro-Blogging et de présenter le modèle de données adopté par notre approche afin fournir les fonctionnalités de filtrage et de recommandation. Nous y présentons aussi les étapes qui nous ont permises de constituer un jeu de données représentatif et présenterons une caractérisation de ces données.

Dans le chapitre 4, nous présentons en détail notre solution de filtrage spécifique au Mirco-Blogging nommée MicroFilter. Nous détaillerons les différentes structures d'index élaborées ainsi qu'un modèle analytique de coût en temps et en mémoire pour chacune des structures. Nous présenterons aussi les résultats expérimentaux sur le comportement de nos index sur un jeu de données conséquent puis nous présenterons une structure hybride qui pressente un compromis en tirant avantage d'aspects complémentaires de deux des structures précédemment testées.

Par la suite, le Chapitre 5 présente RecLand notre système de recommandation pour les réseaux sociaux. Ce chapitre s'ouvre sur la présentation du score de recommandation proposé qui prend en considération les aspects topographiques, sémantiques ainsi que l'autorité des utilisateurs recommandés. Puis nous détaillons les algorithmes de notre approche par *landmarks* qui permettent une approximation des scores de recommandation. Cette approche apporte un gain important en temps de calcul afin de permettre à notre système d'être exploité sur des graphes sociaux de tailles similaires à celles observées en production. Finalement, nous présenterons les résultats expérimentaux de notre système de recommandation ainsi que le résultat d'une validation par les utilisateurs de la pertinence des recommandations générées par notre système.

Enfin, le chapitre 6 clôt le document en résumant le travail réalisé au cours de cette thèse et en présentant les perspectives de recherche.

Chapitre 2

État de l'art

"There's too many men
Too many people
Making too many problems
And not much love to go round
Can't you see
This is a ***land of confusion****"*

— Genesis, *Land of confusion*, 1986

2.1 Introduction

Le travail présenté dans cette thèse s'articule autour des données générées par les utilisateurs des réseaux sociaux. Il est donc primordial d'abord pour nous de comprendre les modes de fonctionnement ainsi que les différentes utilisations que font les Internautes de ces plateformes sociales. La première partie de ce chapitre est dédiée à la caractérisation du Web social, nous y présentons les travaux qui s'intéressent à la nature de ces données ainsi qu'au comportement des utilisateurs sur ces plateformes.
Par la suite, nous nous intéressons aux travaux récents dans le contexte du filtrage sur les réseaux sociaux. Nous exposerons les différentes approches ainsi que les parallèles possibles avec d'autres types d'interactions déjà identifiées et étudiées telles que les systèmes de publication/souscription. La troisième partie va présenter les solutions existantes dans le domaine des systèmes de recommandation. Elle s'ouvre sur les grandes approches adoptées dans ce contexte puis se recentre sur les données sociales. Plusieurs solutions existent

pour générer des recommandations sur les réseaux sociaux, nous exposerons leurs avantages ainsi que leurs limitations. Enfin, nous présenterons quelques approches utiles pour accélérer certains calculs sur les grands graphes, approches dont nous nous sommes inspirés pour adresser le défi du passage à l'échelle.

2.2 Un Web Social

Autour de l'an 2000, la première forme se rapprochant des réseaux sociaux à apparaitre sur le Web fut les *blogs*[1]. Les *blogs* sont des pages Web personnelles, souvent éditées par un seul utilisateur, qui contiennent des publications datées et organisées chronologiquement de plusieurs types : articles divers, opinions politiques, activités quotidiennes, photos, créations artistiques, etc. L'avènement des blogs marqua le début de la déferlante de données générées par les utilisateurs (*User Generated Content* ou *UGC*). Le réseau à proprement dit était alors déduit via les interactions entre utilisateurs (*blogueurs*) qui peuvent référencer d'autres blogs ou bien laisser des commentaires sur les publications de leur choix. Dès 2003, Kumar & al. s'intéressent à la caractérisation du phénomène en étudiant l'ensemble des blogs (appelé *blogosphère*) ainsi que leur évolution [Kumar et al., 2003]. En étudiant les quelques 750.000 liens existants entre plus de 25.000 blogs, les auteurs définissent une structure qu'il nomment le "*Time Graph*" et qui capture les interactions entre les blogs du corpus ainsi que la temporalité de ces liens sous forme de graphe. L'étude montre entre autre que le monde du blogging a vécu une transformation avec pic de croissance et une structuration en communautés vers la fin de l'année 2001, croissance qui n'a pas cessé depuis et qui a évolué vers d'autres types d'interactions.

Les principaux réseaux sociaux vont faire leur apparition entre les années 2003 et 2006 (voir tableau 2.1). Il s'agit alors de créer et d'entretenir des liens sociaux entre les différents utilisateurs. Les différentes plateformes sociales en ligne sont souvent appelés "Médias Sociaux". [Kaplan and Haenlein, 2010] définissent les médias sociaux comme un ensemble d'applications Internet qui permettent aux utilisateurs de produire et d'échanger du contenu (*UGC*), supporté technologiquement par les outils du Web 2.0.

1. Le terme blog est un anglicisme provenant de la contraction de "Web Log", littéralement "Registre Web"

Réseau social	Date de mise en ligne	# d'utilisateurs (en millions)
Friendster	Mars 2002	8.2 (2010)
LinkedIn	Mai 2003	300 (2014)
Hi5	Juillet 2003	100 (2011)
MySpace	Août 2003	36 (2013)
Facebook	Février 2004	1310 (2014)
Twitter	Mars 2006	271 (2014)
Google+	Juin 2011	540 (2013)

Table 2.1 – Dates de lancement et nombre d'utilisateurs des principaux réseaux sociaux

2.2.1 Caractérisation du phénomène

[Java et al., 2007] présente l'une des première études à s'être intéressé aux médias sociaux, plus particulièrement à la plateforme de micro-blogging *Twitter* et ce dès sa première année d'existence. Parmi les résultats de l'étude sur la communauté de moins de 100.000 utilisateurs qui constituait le *Twitter* de l'époque, les auteurs ont démontré que *Twitter* constituait un réseau du type "*Small World Network*". Ce type particulier de réseau est caractérisé par une grande inter-connectivité entre ses utilisateurs. Ce phénomène de "petit monde" a été popularisée par Stanley Milgram et son concept de "six degrés de séparation" [Milgram, 1967] qui stipule que n'importe quelle paire d'individus choisie au hasard aux État-Unis peut être relié par une chaine de six relations en moyenne [2]. Dans le monde virtuel, cette valeur moyenne a été observée et s'avère être d'autant plus petite sur les réseaux sociaux : elle n'est que de 3.74 pour *Facebook* et de 3.05 pour *Twitter* [3] [Backstrom et al., 2012, Myers et al., 2014]. Cette même étude a aussi mis en avant l'aspect communautaire de ce genre de réseaux. En identifiant les grandes communautés thématiques présentes dans *Twitter,* il apparait que les utilisateurs ayant des intérêts communs ont tendance à se regrouper, on parle alors de phénomène de localité. Cette localité peut être thématique (c.à.d les utilisateurs ayant les mêmes centres d'intérêt) mais aussi géographique (utilisateurs se connectant avec d'autres comptes provenant de la même ré-

2. En 1929, l'écrivain Hongrois Frigyes Karinth fut le premier à émettre le postulat des "six degrés de séparation" dans une nouvelle intitulée *Láncszemek* (*Chaînes*)

3. La définition de degré de séparation compte le nombre d'intermédiaires entre deux nœuds du graphe social, elle est égale à distance moyenne entre deux nœuds (nombre moyen d'arcs) moins un.

gion) comme démontré par [Gonzalez et al., 2011].

Étant donné la nature stratégique des données issues des médias sociaux, le rôle commercial de ces plateformes ainsi que la difficulté inhérente à la manipulation de grands volumes de données, très peu d'équipes de recherche ont eu accès à des jeux de données réalistes et de tailles conséquentes [Kwak et al., 2010, Backstrom et al., 2012, Myers et al., 2014].

La nature privée des données sur *Facebook* fait que ce dernier est un des réseaux les plus fermés. Les paramètres de confidentialité choisis par les utilisateurs font que peu d'études existent sur ce réseau en comparaison au monde du micro-blogging où les publications sont le plus souvent publiques. Dans [Backstrom et al., 2012], les auteurs présentent les résultat de leur caractérisation du réseau social *facebook* se basant principalement sur l'algorithme *HyperANF* [Boldi et al., 2011] qui permet une approximation efficace de la fonction de voisinage dans un graphe [4]. Cette analyse sur le plus grand réseau social jamais étudié ($\approx$ 69 milliards de liens d'amitié entre plus de 721 millions d'utilisateurs) a pu démontrer la nature de "petit monde" du réseau avec une valeur de degré de séparation moyen de 3.74 mais aussi que le principal effet de localité existant sur le graphe était celui du à la localité géographique partitionnant le graphe en sous-communautés.

Grâce à une batterie de serveurs autorisés à récolter des données par *Twitter*, Kwak & al. ont pu étudier le comportement des utilisateurs et apporter des éléments de réponse à la question : *Twitter* est-il un réseau social, ou un média d'information ? les résultats de leurs expériences sur 41 millions d'utilisateurs liés par plus de 1,47 milliard de relations de suivi montrent que le réseau présente un compromis entre son aspect social et son rôle dans la diffusion d'information. Il montre aussi que l'application directe de l'algorithme PageRank [Brin and Page, 1998] donne le même résultat que le classement des utilisateurs par nombre de suiveurs (*followers*). PageRank, l'algorithme originel du moteur de recherche *Google*, classe les pages Web dans un graphe hypertexte (graphe des pages Web se référençant) en fonction du nombre de liens référençant la page pondéré par la popularité de celle-ci.

Cette étude à été complétée trois ans plus tard par une équipe au sein

4. La fonction voisinage $N_G(t)$ dans un graphe G retourne pour chaque $t \in \mathbb{N}$ le nombre de paires de nœuds $\langle x, y \rangle$ telles que y est atteignable à partir de x en moins de t sauts

même de *Twitter* [Myers et al., 2014]. Elle retrouve les même grandes lignes sur un graphe plus pertinent car exempt d'éventuels biais liés aux stratégies d'acquisition de données (*crawling*). Nous présentons les principales mesures calculés par cette étude dans le tableau 2.2.

Mesure	Valeur	
Nombre total de nœuds	∼175 Millions	
Nombre total de liens de suivi	∼20 Milliards	
Distance moyenne entre deux nœuds	4.05	
Degré sortant moyen	470	
Degré entrant moyen	339	
Local Clustering Coefficient	mut-deg=5	0.23
	mut-deg=20	0.19
	mut-deg=100	0.14
Degré entrant maximum	14,7 Millions	
Degré sortant maximum	755.000	
% Liens mutuels	42%	

Table 2.2 – Propriétés topologiques de jeu *Twitter* [Myers et al., 2014]

Ces études ont permis de mettre en avant la nature hybride des systèmes de Micro-Blogging. En effet, les valeurs élevées de degré sortant moyen [5] et de degré sortant maximum révèlent que les utilisateurs considèrent les comptes suivis plutôt comme des sources d'informations et non des relations d'amitié dérogeant ainsi à la règle sur les réseaux dit sociaux. Il a été démontré en sciences sociales qu'un individu moyen ne peut entretenir plus de 150 relations sociales stables [Dunbar, 1992].

Le *local clustering coefficient* (traduit parfois par "coefficient d'agglomération" [6]) est une mesure qui évalue si deux nœuds A, B du graphe associés chacun à un même nœud C sont connectés entre eux. Cette mesure a été introduite par [Watts and Strogatz, 1998] pour caractériser les réseaux du type petit monde. Une forte valeur de LCC pour un nœud indique que son voisinage est dense (proche de former une clique). La moyenne du LCC sur

5. Le degré sortant (resp. degré entrant) désigne le nombre de comptes qu'un utilisateur suit (resp. le nombre d'utilisateurs qui suivent un compte) dans le graphe social

6. http://fr.wikipedia.org/wiki/Analyse_des_réseaux_sociaux

tout le graphe permet caractériser le réseau. Sur les réseaux sociaux cette mesure peut être interprétée comme la fraction d'utilisateurs dont les amis sont eux même amis entre eux [Myers et al., 2014]

Les valeurs de LCC présentées dans le tableau 2.2 ont été calculées sur le graphe dit "mutuel". Il s'agit du graphe qui ne contient que les relations du type A suit B et B suit A. Pour une valeur de degré mutuel [7] égale à 5 le LCC est égal à 0,23. Pour cette même valeur de degré [Ugander et al., 2011] rapporte que le LCC moyen sur *Facebook* est de 0,4 ce qui confirme la nature plus "sociale" de *Facebook*. Pour un degré mutuel égal à 100, on retrouve un LCC de 0,14 sur *Twitter* comme sur *Facebook* c.à.d que pour un utilisateur moyen, 14% de ses voisins dans le graphe sont à leur tour liés.

Le principe d'homophilie, terme emprunté à la sociologie qui stipule que les gens avec des goût et des intérêts similaires ont tendance à se connecter, se traduit dans les réseaux sociaux par le fait que des comptes à faible distance sur le graphe sont plus semblables que des comptes éloignés [8]. [Golder and Yardi, 2010] présente une étude qui s'intéresse à la motivation qui mène les utilisateurs des plateformes de micro-blogging a créer de nouveaux liens. Il apparait que la transitivité (le fait qu'un nœud soit suivi par un compte déjà suivi) et la mutualité sont parmi les facteurs de motivation les plus importants considérés par les utilisateurs pour former de nouveaux liens. Ce résultat sur la transitivité a été validé aussi par [Romero and Kleinberg, 2010] qui montre les utilisateurs créent souvent de nouveaux liens sur le réseau en court-circuitant des chemins de longueur 2 (se lient avec des amis d'amis).

Au niveau du contenu généré par les utilisateurs des systèmes de micro-blogging, la limitation de la taille des publications à 140 caractères pousse les utilisateurs à adopter diverses conventions de langage. [Laboreiro et al., 2010, Han and Baldwin, 2011] ont analysé l'aspect lexical d'un corpus de *tweets* et relèvent des similitudes avec le langage communément appelé "langage SMS". Les principales conventions de notation utilisés dans *Twitter* sont résumés dans le tableau 2.3.

7. Le nombre d'arcs réciproques pour nœud ($c.à.d$ $A \rightarrow B$ et $B \rightarrow A$)
8. http://lafeuille.blog.lemonde.fr/2011/12/29/homophilie-et-proximite-dans-les-reseaux-sociaux-de-lecteurs/

Notation	Signification
@	Sert à citer un autre utilisateur en précédant son pseudonyme par un @
#	Avant un des mots du *tweet*, permet de désigner ce mot en tant que mot-clé pour catégoriser la publication, les mots précédés du symbole dièse sont appelés "*hashtags*"
RT	Désigne un *retweet* ou le fait de re-publier un *tweet* reçu
#FF	Désigne le "Follow Friday", chaque Vendredi les utilisateurs publient des listes de comptes qu'ils jugent intéressants à suivre

Table 2.3 – Conventions de notation sur *Twitter*

2.2.2 Comportement des utilisateurs

L'ensemble de notations adoptées par les utilisateurs enrichissent la sémantique associée aux *tweets*. Ces conventions créent aussi de manière tacite un graphe de relations implicites entre les utilisateurs. On peut distinguer par exemple le graphe des @citations où les nœuds représentent les utilisateurs et les arcs la présence d'une citation entre deux nœuds ou bien le graphe des *retweets* ou un arc représente le *retweet* d'un utilisateur d'un *tweet* publié par un autre compte.

Cette sémantique peut être exploitée afin de comprendre le comportement des utilisateurs et de calculer diverses métriques sur celui-ci. [Welch et al., 2011] a passé en revue les différents liens pouvant exister entre les utilisateurs, entre les utilisateurs et les *tweets* ainsi que les liens inter-*tweets*. Les auteurs ont mis en avant le fait que le lien de *retweet* est un indicateur d'intérêt entre deux comptes plus fort que le lien de suivi (*following*). De plus, ce lien présente l'avantage de pouvoir être facilement associé à un centre d'intérêt précis (*topic*). Ce *topic* pouvant être extrait en analysant simplement les *retweets*.

Dans [Cha et al., 2010], les auteurs comparent deux méthodes pour le calcul de l'influence des utilisateurs sur le réseau se basant sur le degré entrant et les @citations. Ils ont observé que le degré entrant est un bon

indicateur en ce qui concerne la popularité d'un compte mais qu'il échoue à caractériser son influence sur le réseau (calculée en terme de *retweets* ou de @-citation). Aussi, en étudiant les comptes les plus influents sur le réseau, les auteurs ont pu montrer que l'influence sur le réseau s'obtient par un effort de cohérence de la part des comptes influents en limitant par exemple leurs publications à un sujet précis (*topic*). Une autre approche dans la mesure de l'influence des utilisateurs consiste à la calculer sur l'ensemble du réseau et d'obtenir ainsi les top-utilisateurs sur tout le réseau [Pal and Counts, 2011]. Dans [Vosecky et al., 2012], les auteurs utilisent aussi les liens implicites combinés à une analyse du contenu des *tweets* afin de filtrer et classer les publications en fonction de leur qualité.

[Achananuparp et al., 2012] propose trois modèles différents pour capturer le comportement des utilisateurs sur les réseaux sociaux. Les auteurs ont montré que ces modèles comportementaux pouvaient être employés pour détecter des événements de manière efficace en contraste avec les méthodes basés sur l'analyse du contenu des publications.

2.2.3 La détection d'événements

L'aspect événementiel des réseaux sociaux et notamment des plateformes de micro-blogging est incontestable. Des pics d'activité sont souvent relevés au moment d'événements importants. Les méta-données associés à un *tweet* contiennent un champ sur la position géographique de l'émetteur au moment du *tweet* ce qui permet, selon la nature de l'événement, de le localiser géographiquement. Les pics d'activité peuvent alors concerner une région précise (ex. les catastrophes naturelles) ou bien être généralisés pour des événements d'ampleur mondiale (ex. nouvel an).

Une étude de 2010 menée par Tumasjan & al. sur *Twitter* pendant les élections fédérales en Allemagne révèle que la plateforme est utilisé activement par les Internautes pour exposer et débattre de leurs opinions politiques. L'analyse de plus de 100.000 *tweets* a permis de montrer un fort rapprochement avec les sondages effectués par des organismes spécialisés en cette période ainsi qu'une forte corrélation avec le résultat final du scrutin [Tumasjan et al., 2010].

Dans [Sakaki et al., 2010], les auteurs tirent parti de l'aspect temps-réel des plateformes de micro-blogging. Ils présentent un algorithme pour le monitorage de *Twitter* et la détection des tremblements de terres au Japon.

Grâce a la forte adoption de *Twitter* au Japon, le réseau agit comme un capteur qui permet de détecter avec succès 96% des tremblements de terre de magnitude supérieure à 3, de localiser approximativement l'épicentre et d'émettre des bulletins d'alerte plus rapidement que les voies traditionnelles.

2.3 Le Filtrage

La nature hybride des plateformes de micro-blogging liée aux flux sans cesse croissants de données s'y écoulant introduit le besoin de mécanismes de filtrage efficace pour les utilisateurs. Pour éviter le phénomène de saturation (*flooding*), certains travaux se sont concentré sur la présentation des données aux utilisateurs des plateformes de micro-blogging afin d'améliorer la lisibilité des flux reçus.

2.3.1 Stratégies de filtrage

Certains travaux se sont inspirés du domaine de la recherche d'information. [Sriram et al., 2010] présente une approche de *clustering* et de classification des *tweets* basée sur l'établissement au préalable d'un profil utilisateur. Cette technique s'appuie sur un classificateur qui s'occupe de router les nouvelles publications vers des classes préétablies. Cette classification peut se baser aussi sur les sujets d'actualité abordés dans le réseau (*hot trends*). Dans [Sankaranarayanan et al., 2009] par exemple, les auteurs décrivent une méthode qui permet l'extraction de concepts à partir des *tweets*, et l'utilisent pour détecter des sujets d'actualité (*news*) depuis *Twitter*. Des travaux tels que [Weng et al., 2010, Bakshy et al., 2011] s'appuient plutôt sur l'influence globale des utilisateurs sur le réseau, calculée via différentes techniques, pour classer les *tweets* et appliquer des techniques de classement. Dans [Uysal and Croft, 2011], les auteurs utilisent le comportement de *retweet* comme indicateur d'intérêt et s'en servent pour faire remonter les publications les plus *retweetés* dans les résultats.

Une autre technique de filtrage est présentée dans [Haghani et al., 2010] où les auteurs appliquent des algorithmes de *top-k* sur des flux Web 2.0 tout en considérant l'aspect temps-réel de ces données ainsi que leur fraicheur via une fenêtre temporelle. [Albakour et al., 2013] propose une technique de filtrage avec réécriture de requêtes qui prend en considération la taille réduite des *tweets* ainsi que le cycle de vie limité et l'aspect spontané de certaines requêtes (ex. lors d'un événement particulier).

Plus généralement, le problème de la propagation de données sur le Web 2.0 avec l'existence d'un graphe sous-jacent a déjà été abordé par différents travaux. [Silberstein et al., 2010] considère des flux à haute fréquence de mise à jour. Les auteurs rapportent le problème à un problème de calcul de vues sur des flux de données et proposent une méthode pour matérialiser les événements du flux de manière sélective afin d'améliorer l'aptitude au passage à l'échelle. Il s'agit d'adopter une approche hybride en distinguant les comptes à forte fréquence de publication des comptes à plus faible fréquence. *Twitter* utilise une technique similaire pour créer les vues pour ses utilisateurs (le *timeline Twitter* [9]).

2.3.2 Indexation

Quelques travaux récents s'attaquent au problème du passage à l'échelle pour l'indexation et la recherche d'information dans des flux issus de réseaux sociaux. [Busch et al., 2012] présente *EarlyBird*, le système d'indexation en temps-réel déployé par *Twitter* pour fournir la fonction de recherche. *EarlyBird* se base sur une structure de listes inverses qui lie les requêtes (formulées par les utilisateurs sous forme de mots-clés) à des listes de *tweets* fonctionnant en ajout-seulement (*append-only*). Ces listes, maintenues par le système en ordre chronologique ascendant, permettent de récupérer efficacement les *tweets* les plus récents pour une requête donnée avec des temps de latence de moins de 200 millisecondes pour l'indexation ainsi que pour recherche. Le défi étant de répondre efficacement aux requêtes de recherche tout en réduisant au maximum le temps nécessaire à nouveau *tweet* pour être indexé.

Dans *TweetIndex* [Chen et al., 2011], les auteurs proposent une approche intéressante en affectant une priorité d'indexation plus élevée à un nouveau *tweet* si ce dernier présente une forte probabilité d'être associé à une requête de recherche. L'indexation des *tweets* ayant une faible probabilité d'être recherchés est quant à elle retardée.

[Wu et al., 2013], présente une approche similaire. *LSII*, la structure proposée, implémente une cascade à multi-niveaux d'index de tailles croissantes afin d'insérer les nouveaux *tweet* dans les plus petits index. Par la suite, les *tweets* plus anciens sont déplacés vers des index plus grands (donc moins efficaces) par lots.

9. blog.evanweaver.com/2010/08/12/distributed-systems-primer-update/

2.3.3 Systèmes de publication/souscription

Des parallèles existent entre le domaine du filtrage de flux sociaux et les systèmes de publication/souscription (*pub/sub*). Différentes structures d'indexation ont été proposées pour gérer des flux de nouvelles publications et vérifier efficacement la satisfaction de souscriptions. *Le Subscribe* [Pereira et al., 2000] fut l'un des premiers systèmes *pub/sub* à traiter des données hautement dynamiques tels que les données du Web. Les auteurs présentent une structure pour évaluer la correspondance des souscriptions en comptant le nombre de prédicats contenus.

Dans [Broder et al., 2011], les auteurs supposent un système du type *pub/sub* ou les producteurs de publications ainsi que leur souscripteurs sont liés dans un graphe logique dirigé (comme dans les réseaux sociaux) régit par des contraintes (sous forme de prédicats). Il proposent ainsi des algorithmes pour l'évaluation efficace de contraintes dans un tel graphe (ex. la propagation d'une publication dans un réseau social du type *Facebook* ou les utilisateurs peuvent avoir différents paramètres de confidentialité). Dans [Hmedeh et al., 2012] , les auteurs proposent des schémas d'indexation sur des systèmes *pub/sub* dans un contexte RSS afin de satisfaire un grand nombre de requêtes utilisateur le tout en prêtant une attention particulière au passage à l'échelle.

2.4 Les systèmes de recommandation

2.4.1 Définition

Face aux flux importants de données circulant sur le Web, une des approches adoptées pour améliorer l'expérience utilisateur consiste à fournir des recommandations de contenu pertinent. D'après Chris Anderson dans "*The Long Tail*", les bouleversements qu'a subi le Web et la masse de données qui constituent Internet font que "*nous quittons progressivement l'âge de l'information pour rentrer dans l'âge de la recommandation*" [Anderson, 2006].

La tâche d'un système de recommandation consiste à accomplir un filtrage d'information afin de suggérer à un utilisateur des articles à acheter (ex. *e-commerce*) ou bien d'autres utilisateurs avec qui interagir/se connecter (ex. réseaux sociaux) [Ricci et al., 2011]. Selon leur mode de fonctionnement, les systèmes de recommandation peuvent être *personnalisés*

ou *non-personnalisés*. Les systèmes de recommandation *non-personnalisés* fournissent aux utilisateurs des suggestions mais ne prennent pas en considération les préférences de ces utilisateurs. Une illustration d'un tel système est par exemple la recommandation des articles les plus populaires sur une boutique en ligne ce mois-ci ou bien la liste de chansons qui font le plus de passages radio sur une période donnée (*top-50*).

Du fait de la richesse de la sémantique associée aux données sociales, nous nous intéressons plus particulièrement aux systèmes de recommandation *personnalisés*. Ces systèmes fonctionnent en utilisant les caractéristiques des utilisateurs, leurs préférences, profils ou bien leur interactions passées afin de fournir des suggestions de contenu pertinent. Ces suggestions peuvent aussi être vues d'un autre angle comme étant des prédictions sur les interactions futures de ces mêmes utilisateurs. Trois grandes approches de systèmes de recommandation *personnalisés* existent : Les systèmes basés sur le contenu, les systèmes basés sur le filtrage collaboratif et les systèmes hybrides.

2.4.2 Les systèmes basés sur le contenu

Les systèmes de recommandation basés sur le contenu (*content-based*) fonctionnent en analysant les caractéristiques des objets à recommander (produits, etc.) puis en les regroupant. Par la suite, le système va suggérer aux utilisateurs ayant acheté/consommé un produit quelconque par le passé, les objets/produits estimés similaires [Ricci et al., 2011].
L'architecture générale d'un système de recommandation basé sur le contenu s'articule autour de 3 modules principaux :

— L'analyseur de contenu – Selon la nature des données à recommander (texte, éléments multimédia, pages Web, produits commerciaux, etc.) une étape de pré-traitement est nécessaire afin de décrire les objets à recommander et d'en extraire les caractéristiques. Le module d'analyse de contenu est responsable de produire une description structurée de ces objets. Cette description va servir d'élément d'entrée aux autres modules.

— Le module d'apprentissage de profils – Ce module est responsable de l'analyse des interactions passées de l'utilisateur sur les objets du système. En utilisant des méthodes empruntées au monde de l'apprentissage, ce module construit une description des préférences

des utilisateurs.

— Le module de filtrage – A partir des profils utilisateurs et des descriptions des objets à recommander, ce module construit des listes de suggestions à présenter aux utilisateurs.

[Diederich and Iofciu, 2006] présente un exemple de système de recommandation basé sur le contenu. Les profils utilisateurs y sont déduits à travers les étiquettes (*tags*) issues de l'activité d'annotation effectuée par les utilisateurs sur le système. Ce système se base par la suite sur ces profils pour générer des recommandations de collaborateurs potentiels.

2.4.3 Le filtrage collaboratif

La deuxième grande famille de systèmes de recommandation est basée sur l'hypothèse que les utilisateurs qui ont aimé des articles similaires par le passé ont un goût similaire et vont donc apprécier les mêmes articles dans le futur. Un des exemples les plus connus d'un tel système a été popularisé par le site de commerce en ligne *Amazon.com* et son algorithme de *Item-to-item Collaborative Filtering* qui se traduit sur le site par la fonctionnalité "Les gens qui ont acheté le produit x ont aussi acheté le produit y" [Linden et al., 2003].

L'avantage principal de cette approche est qu'elle ne nécessite pas de description précise des objets à recommander. Les recommandations étant basées sur l'ensemble des interactions des utilisateurs avec les objets/produits, cette méthode permet de recommander des objets complexes sans avoir à les analyser. La plupart des services de recommandation de musique en ligne fonctionnent sur ce mode (ex. *last.fm* [10]) car les fichiers multimédia sont difficiles à analyser. Pour pouvoir fonctionner le système a besoin de collecter des données sur les utilisateurs et leurs préférences, cette collecte peut se faire de deux façons :

— Collecte Explicite – Dans ce cas, les utilisateurs sont sollicités pour émettre leurs avis sur des produits/objets. Il peuvent le faire via un système de notation (ex. une grille de 5 étoiles, un questionnaire de satisfaction), ou bien en publiant leurs avis sur un élément donné

10. http://www.last.fm/

(ex. La fonction *"J'aime"* sur le réseau social *Facebook* permet aux utilisateurs d'exprimer leur intérêt pour un élément donné).

— Collecte Implicite – La collecte implicite s'intéresse aux interactions des utilisateurs sur le système. Les exemples de cette collecte incluent la surveillance du nombre de visites sur une page, le nombre de vues sur une vidéo, le temps passé sur une section donné ou de l'historique des achats sur une plateforme de *e-commerce*.

[Das et al., 2007] décrit la plateforme de recommandation utilisé par *Google News*. L'approche basée sur le filtrage collaboratif a permis au système d'être indépendant vis-à-vis du contenu des publications suggérées et à la technique d'être adaptée à d'autres applications ou de gérer d'autre langues à moindre coût.

2.4.4 Les systèmes hybrides

Chacune des approches de recommandation présentées présente des avantages ainsi que des inconvénients. Par exemple, les systèmes basés sur le contenu ont besoin d'un riche historique d'interactions pour pouvoir fonctionner ; le système ne pourra pas fournir recommandations de qualité pour un utilisateur fraichement inscrit. D'un autre côté, les systèmes du type filtrage collaboratif ont besoin de l'existence d'une large base d'interactions sur l'ensemble du catalogue d'objets/produits du système afin de pouvoir calculer des rapprochements entre les utilisateurs.

Une solution consiste à proposer des systèmes hybrides, qui tirent parti des avantages des deux approches citées précédemment. Cette solution permet de combler les lacunes de l'une des approches sur des cas d'utilisation précis. Ces approches hybrides peuvent être implémentées de diverses façons [Adomavicius and Tuzhilin, 2005] :

— En combinant les résultats produits par chacune des deux approches exécutées indépendamment ;

— En sélectionnant un aspect précis d'une des approches et en l'intégrant à l'autre (compléter les approches) ;

— En unifiant les deux approches dans un modèle global.

Un grand nombre de systèmes de recommandation actuellement en exploitation fonctionne sur un modèle hybride. Parmi les systèmes hybrides les plus connus, le système de recommandation mis en place par le géant Américain de la vidéo à la demande sur Internet *Netflix* [Amatriain, 2013]. Après avoir proposé un prix d'un million de Dollars en 2006 aux travaux obtenant les meilleures recommandations sur les données issues de la plateforme, l'entreprise a intégré les propositions les plus pertinentes dans sa version du système de recommandation mis en production. Il est reporté que plus des deux tiers du total des vidéos consommées sur cette plateforme sont issues des seules recommandations présentées aux utilisateurs ce qui reflète l'importance de la capacité à fournir des recommandation de qualité.

2.4.5 Recommandations sociales

Les données issues des réseaux sociaux représentent une véritable mine d'information pour un éventuel système de recommandation. Fonctionnant sur l'adage *"Dis moi qui tu fréquentes, je te dirai qui tu es"*, nous pouvons identifier un nouveau type de systèmes de recommandation basé sur la présence d'une *communauté* d'utilisateurs liés par des liens sociaux [Ricci et al., 2011]. Sur les plateformes sociales, ces systèmes de recommandation permettent de recommander tout un ensemble d'informations. Les exemples incluent des utilisateurs à suivre, des publications précises, des éléments multimédia, des groupes (sous-communautés) à intégrer etc.

La principale caractéristique des réseaux sociaux étant l'existence d'un graphe de relations sociales, cette donnée est l'information principale au centre des diverses stratégies de recommandation. [Liben-Nowell and Kleinberg, 2003] présente une étude qui compare diverses méthodes de prédiction de liens. La prédiction de liens consiste à analyser l'état du réseau à un instant t afin d'anticiper la création de nouveaux liens à un moment $t + 1$. Ces techniques sont souvent utilisées dans le contexte de la recommandation. Les méthodes présentées par Liben-Nowell & Kleinberg se basent sur les propriétés topologiques des réseaux pour le calcul des prédictions.

Parmi les mesures présentés, la mesure de *Katz* présente des résultats pertinents une fois exploitée sur des graphes issus du Web. Cette mesure, issue du monde de la sociologie [Katz, 1953b], se base sur la connectivité entre deux nœuds du graphe social. Plus le nombre de chemins entre deux nœuds est élevé et plus la longueur des ces chemins est courte, plus le score de *Katz* entre ces deux nœuds sera élevé. Concrètement, le score de *Katz*

entre un nœud u et un nœud v s'exprime comme suit :

$$katz_\beta(u,v) = \sum_{l=1}^{\infty} \beta^l \times |P_{u,v}^{\langle l \rangle}| = \sum_{p \in P_{u,v}} \beta^{|p|}$$

Où β représente un facteur de décroissance ($\beta \in [0,1]$), $P_{u,v}$ représente l'ensemble de tous les chemins existant entre u et v et $P_{u,v}^{\langle l \rangle} \subseteq P_{u,v}$ l'ensemble de tous les chemins de longueur égale à l existant entre u et v. Le facteur β est utilisé pour donner plus d'importance aux chemins courts (c.à.d aux nœuds proches dans le graphe) exploitant ainsi le phénomène de localité et d'homophilie.

Une autre mesure topologique est présentée dans [Budalakoti and Bekkerman, 2012], les auteurs proposent de combiner deux scores de classement de comptes basés sur deux sources différentes issues du même réseau (les invitations sur le réseau social, ainsi que le graphe social proprement dit). Ces données sont ensuite utilisées pour produire la liste triée des comptes les plus influents sur le réseau.

Certains travaux appliquent des méthodes de filtrage collaboratif ainsi que des méthodes basés sur le contenu dans un contexte social. [Chen et al., 2012] introduit une méthode pour classer des *Tweets* qui exploite les profils de préférences utilisateur, une mesure d'autorité du compte publiant le *tweet* ainsi que la qualité de la publication. [Hannon et al., 2010] passe en revue une série de méthodes d'extraction de profils utilisateur sur le réseau *Twitter*. Il y est testé des méthodes basées sur le contenu publié par l'utilisateur, celui des comptes qu'il suit ainsi que celui de ses *suiveurs*. Un classement basé sur le score de TF-IDF est ensuite employé pour trouver les utilisateurs similaires. Une méthode similaire de recommandation est adoptée par [Pennacchiotti et al., 2012].

Dans [Diaz-Aviles et al., 2012] les auteurs décrivent une approche qui garde trace des interactions passées par les utilisateurs pour le calcul en temps-réel des recommandations. Les techniques présentées par [Pennacchiotti et al., 2012, Diaz-Aviles et al., 2012] et [Chen et al., 2012] fournissent des recommandations au niveau de granularité du *tweet*. Le passage à l'échelle est alors problématique étant donné le nombre important de *tweets*, l'aspect temps-réel de la plateforme et les fréquences élevées de publication.

Les approches citées précédemment ne prennent pas en considération

la topologie du graphe dans le calcul des recommandations. [Weng et al., 2010] présente une adaptation de l'algorithme *PageRank* au monde du micro-blogging appelée *TwitterRank*. Cette approche prend en compte la structure des liens du graphe ainsi que l'autorité des utilisateurs sur des sujets (*topics*) donnés. Les *topics* utilisés par *TwitterRank* sont obtenus par l'application de la méthode LDA (Allocation de Dirichlet Latente) qui est une technique probabiliste permettant la caractérisation du contenu des utilisateurs. [Liang et al., 2012] propose une méthode basée sur le contenu afin de fournir des recommandations de sujets (*topics*) en exploitant les liens implicites issus des conventions adoptées sur les plateformes de micro-blogging (voir tableau 2.3). [Kywe et al., 2012] propose une technique de recommandation de *hashtags* personnalisée. Certains *hashtags* ayant un cycle de vie extrêmement court, les auteurs proposent de combiner le contenu des *tweets* avec une approche de filtrage collaboratif sur une fenêtre temporelle d'un mois afin de recommander des *hashtags* pertinents aux utilisateurs.

Une approche présentée par [Chaoji et al., 2012] vise à maximiser la découverte de nouveau contenu lors de la tâche de recommandation. Ce problème s'apparente à un problème d'optimisation multi-objectif *NP-difficile*. Les auteurs proposent une approximation qui atteint un degré de propagation de contenu important. Cependant l'application de cette méthode sur de grands graphes demeure problématique.

Dans [Gupta et al., 2013], les auteurs présentent le système de recommandation mis en production par *Twitter* pour sa fonction "*Qui suivre ?*" (*Who to follow ?*). Il se base sur le déploiement de l'algorithme SALSA [Lempel and Moran, 2001] dans un environnement centralisé. SALSA fonctionne en créant un graphe biparti avec d'un côté le cercle de confiance d'un utilisateur (pouvant être calculé par exemple comme l'ensemble les comptes avec qui l'utilisateur interagit le plus ou bien à partir d'un algorithme de marche aléatoire) et de l'autre coté les comptes les plus suivis par ce cercle de confiance, considéré comme des autorités. Cette approche ne prend pas en considération le sujet (*topic*) sur lequel les comptes recommandés peuvent être une autorité.

2.5 Mesures sur les graphes sociaux

Les calculs de mesures sur les réseaux sociaux peuvent vite devenir problématiques. En effet, la taille des graphes disponibles sur le Web couplée à la complexité des algorithmes de calcul sur les graphes rendent n'importe quelle tâche de mesure difficile. Dans cette section nous présentons des techniques d'approximation utilisées pour mesurer de manière approchée les longueurs de plus courts chemins sur les graphes issus du Web. Nous nous sommes inspiré de ces techniques pour proposer la stratégie de recommandation présentée en chapitre 5.

Pour accélérer les calculs de recommandation, nous pré-calculons les scores de recommandation pour un ensemble de nœuds élus appelés *landmarks*. Le calcul basé sur les *landmarks* est une technique connue qui fonctionne sur le paradigme de *"Diviser pour mieux régner"*. Cette technique est utilisée traditionnellement pour l'estimation du calcul de la longueur du plus court chemin entre deux nœuds sur un graphe. L'idée consiste à sélectionner un sous-ensemble de nœuds L pour lequel le système stocke et maintient les distances vers d'autres nœuds. La distance $d(u, v)$ entre deux nœuds u et v est par la suite estimée en calculant le minimum de $d(u, l) + d(l, v)$, où l représente un *landmark* ($l \in L$) [Thorup and Zwick, 2001]. Le choix des *landmarks* étant crucial pour une estimation précise, diverses stratégies de sélection peuvent être employées (voir chapitre 5)

[Sarma et al., 2010] a choisi de pré-calculer les opérations liées au calcul de la longueur du plus court chemin dans des structures appelées "*Sketches*". Ces structures sont sollicitées au moment de la requête pour fournir une estimation de la longueur de plus court chemin. Cette technique permet le passage à l'échelle sur des graphes de tailles similaires aux graphes issus du Web social. [Gubichev et al., 2010] propose une extension de l'algorithme basé sur la structure de *Sketch* proposé par [Sarma et al., 2010] afin de répondre aux requêtes de calcul de la longueur du plus court chemin en permettant aussi la récupération du chemin en question ce qui est utilisé afin améliorer la précision générale du système.

[Tretyakov et al., 2011] propose d'utiliser une structure arborescente pour le stockage des données liées à l'estimation des plus courts chemins. L'avantage de l'utilisation de cette structure réside dans le fait qu'elle supporte aisément les mises à jours liées à l'évolution de la topologie du graphe. Les auteurs introduisent aussi une stratégie de sélection de *landmarks*. Cette stratégie a pour but de maximiser la zone de couverture dans laquelle le système peut estimer les plus courts chemins dans le graphe.

[Potamias et al., 2009] contient une étude sur l'impact des stratégies de sélection de *landmarks* sur la précision des estimations de longueurs de plus courts chemins. Les auteurs ont prouvé que le problème de l'optimisation de la zone de couverture des *landmarks* dans un graphe est un problème *NP-difficile* et ils ont démontré expérimentalement qu'un choix réfléchi des stratégies de sélection de *landmarks* permettait d'améliorer sensiblement la précision du système et ainsi d'obtenir de meilleurs résultats.

2.6 Conclusion

Dans ce chapitre, nous avons présenté l'état de l'art des travaux qui s'intéressent aux données générées par les utilisateurs des médias sociaux. Nous avons passé en revue :

- Des études qui se sont intéressés à la caractérisation du phénomène du Web social
- Des travaux proposés afin d'améliorer la lisibilité des flux à travers des solutions de filtrage
- Des systèmes de recommandations qui permettent de présenter du contenu pertinent aux utilisateurs
- Les approches d'approximation par *landmarks* afin d'effectuer des mesures de plus courts chemins sur de grand graphes

Ce travail nous permet de situer notre contribution qui vise à améliorer l'expérience des utilisateurs sur les plateformes de micro-blogging à travers l'introduction de fonctions efficaces pour filtrage et la recommandation.

Chapitre 3

Données issues des réseaux sociaux

"Walked out this morning
Don't believe what I saw
*A **hundred billion bottles***
Washed up on the shore"

— The Police, *Message in a bottle*, 1979

3.1 Introduction

L'avènement des médias sociaux a marqué l'explosion du volume de données générés par les utilisateurs (*UGC, User Generated Content*) disponibles sur le Web. Dans le chapitre précédent nous avons mis en avant l'effort de recherche qui a été effectué sur ce type de données et notamment sur les données issues des plateformes du type micro-blogging. Nous avons observé qu'un nombre important d'études portent sur le mircro-blogging, ceci est dû principalement à la nature publique des informations publiées sur ces plateformes.

Dans ce chapitre, nous présentons le modèle de données que nous avons proposé afin modéliser les plateformes du type micro-blogging. Nous exposons ensuite les différentes étapes qui nous ont menés à la constitution d'un jeu de données réaliste sur lequel nous avons pu tester nos propositions

dans un contexte de filtrage et de recommandation d'information. Finalement, nous présenterons les caractéristiques des jeux de données obtenus en contraste avec les caractéristiques des données réelles issues de ces plateformes que nous avons présenté en chapitre 2.

3.2 Le Micro-Blogging

Dans un système de micro-blogging, un utilisateur, via son compte, s'abonne aux mises à jour d'autres comptes sur la plateforme. On dit alors qu'un compte "*suit*" un autre compte. Chaque publication (désignée par *tweet* dans ce qui suit) émise par un compte donné est transmise à tous les comptes des utilisateurs qui suivent le compte émetteur. Un large graphe social de relations de suivi existe donc entre les différents comptes sur le système.

Afin de concevoir des structures capables de gérer les logiques de filtrage et de recommandation proposées, nous avons dû prêter une attention particulière à certaines caractéristiques des réseaux sociaux et des plateformes du type micro-blogging en particulier. Ces caractéristiques, décrites en chapitre d'introduction, se résument en :

— **Publications courtes**
La restriction de la taille des publications (140 caractères) nous pousse à réfléchir à un traitement efficace des documents indexés (*ex.* la taille moyenne d'un *tweet* sur *Twitter* est de 14.7 termes [Foster et al., 2011]). Cette restriction est due historiquement au fait que ces systèmes ont été pensés pour offrir un accès via les réseaux *GSM* via le protocole SMS.

— **Hétérogénéité des comptes en taille**
Les différentes études conduites sur le phénomène du micro-blogging ont révélé une forte hétérogénéité en ce qui concerne la fréquence de publication ainsi que le nombre de *followers* des comptes.

— **La dynamicité du graphe**
Comme observé dans *Twitter* [Kivran-Swaine et al., 2011,Kwak et al., 2011], les utilisateurs s'abonnent et se désabonnent très souvent des comptes suivis. Nos propositions doivent êtres capables de suivre et de gérer efficacement cette évolution.

— **Un Système centralisé**
Dans leur grande majorité, les systèmes de micro-blogging existants reposent sur une architecture purement centralisée. Chaque nouvelle publication est d'abord transmise aux serveurs centraux du système qui, détenant le graph social, s'occupent de la faire suivre aux différents *followers* du compte émetteur. Ceci est dû essentiellement à des raisons de sécurité, de contrôle de confidentialité et autres politiques de placements publicitaires.

— **Conventions/Notations**
En raison de la simplicité du modèle, les utilisateurs des plateformes de micro-blogging ont adopté divers conventions et notations afin d'enrichir les fonctions proposées par ces plateformes (voir section 2.2.1).

— **Les listes *Twitter***
Une fonctionnalité appelée "listes" dans *Twitter* permet de regrouper certains utilisateurs dans un ensemble cohérent pour ainsi accéder à un flux réduit contenant uniquement les publications des utilisateurs groupés. Les utilisateurs peuvent créer leurs propres listes ou bien s'abonner à des listes déjà existantes. Cette fonction simple fournit un mécanisme basique de curation de contenu par une approche de regroupement (*clustering*) de comptes. Certains travaux s'intéressent aux listes pour divers scénarii allant de l'identification de *topics* à la recommandation [Yamaguchi et al., 2012, Velichety and Ram, 2013, Garcia-Silva et al., 2012]. En effet, les utilisateurs souvent groupés ensemble sont souvent associés à une thématique commune. En revanche, cette fonctionnalité ne fait que diviser le flux de données reçues par un compte en "sous-flux" indépendants.

En considérant les particularités des données issues du monde du micro-blogging, nous présentons dans la section suivante le modèle de données proposé pour représenter un tel système.

3.3 Modèle de données

Afin de modéliser un réseau social de type micro-blogging fournissant les fonctions de filtrage et de recommandation, nous commençons par présenter un exemple de graphe social où les utilisateurs ont des intérêts précis.

3.3.1 Exemple de graphe social

Nous supposons un graphe orienté G qui nous servira à illustrer notre modèle de données. Michel, un utilisateur du système, décide de suivre deux comptes très actifs sur le réseau qui sont ceux de *CNN* et *L'AFP* (Agence France Presse). Ces deux comptes sont spécialisés dans l'actualité et publient une quantité importante de publications par jour à propos de divers sujets. AFP étant connu pour produire des *tweets* de qualité en ce qui concerne les flashs d'information, *CNN* suit alors AFP et reçoit tous ses *tweets*. D'un autre côté *AFP* ne couvre pas les sujets IT et Cinéma et se base sur les publications de *CNN* pour ces deux domaines. Remarquant une certaine redondance et pour éviter d'être submergé de *tweets* Michel voudrait donc recevoir les publications qui concernent l'IT et le cinéma seulement de la part de *CNN* et ceux concernant la politique de la part de *AFP*.
Cédric de son côté décide de suivre *AFP* uniquement pour les publication qui traitent de sport. Enfin, Ryadh suit Cédric pour les *tweets* qui concernent l'IT (voir figure 3.1).

Nous nous baserons sur cet exemple pour illustrer notre modèle de données ainsi que nos propositions de filtrage en chapitre 4.

3.3.2 Le système de micro-blogging

Un système de micro-blogging à l'instar de *Twitter* peut être représenté sous forme de graphe *G=(N, E)* où l'ensemble de nœuds N représente les utilisateurs (comptes) du système et l'ensemble des arcs $E \subseteq N \times N$ représente les liens de suivi (*following*). Plus précisément un arc orienté *e=(u, v)* existe entre un nœud u et un nœud v si l'utilisateur représenté par u est notifié à chaque fois que l'utilisateur v publie un *tweet* (u reçoit les mises à jour de v).

Pour un nœud n, on définit $\Gamma^+(n)$ l'ensemble des nœuds suivis par n comme

$$\Gamma^+ : N \to 2^N, \Gamma^+(n) = \{n' | (n, n') \in E\}$$

De manière similaire on définit $\Gamma^{-}(n)$ l'ensemble de nœuds qui suivent n.

Chaque nœud produit des publications ou *tweets*. Une publication est définie comme une séquence de termes $p =< t_0, t_1, t_2, \ldots, t_i >$. On note P l'ensemble des publication et $\mathcal{V}_P$ le vocabulaire de ces publications. A noter que nous ne basons pas sur l'ordre des termes pour la correspondance (*matching*) : Par conséquent nous considérons plutôt l'ensemble des termes d'une publication $p = \{t_0, t_1, t_2, \ldots, t_i\}$ plutôt que la séquence. La *séquence de tweets* d'un nœud n, notée n_s désigne la séquence de *tweets* triée chronologiquement $n_s =< p_0, p_1, p_2, \ldots, p_i >$ publiés par n.

3.3.3 Les filtres

Nous proposons un filtrage basé sur des mots-clefs. Un filtre F dans notre système est représenté comme un ensemble de termes distincts $F = \{t_1, t_2, ..., t_n\}$ où chaque terme t_i appartient au vocabulaire de termes noté $\mathcal{V}_F$. La taille de F, notée $|F|$, est le nombre total de termes distincts contenu dans le filtre. Comme dans [Yan and Garcia-Molina, 1994], nous supposons que $\mathcal{V}_F \subseteq \mathcal{V}_P$. $\mathcal{F}$ représente l'ensemble de filtres, sans contenir le filtre $\perp$ qui satisfait toutes les publications, *c.à.d.*, $\perp = \mathcal{V}_P$. Une fonction *label* associe un filtre à chaque arc du graphe social G :

$$label : E \rightarrow \mathcal{F} \cup \perp$$

Nous appelons le graphe social dont les arcs sont associés à des filtres le *labelled social graph* (*LSG*). A noter que l'utilisateur peut donc exprimer différents intérêts à travers les filtres et ce pour une source précise. Dans l'exemple cité en 3.3.1, l'utilisateur $Michel$ veut récupérer tout les *tweets* de *CNN* qui traitent de *IT*, *politique* et *cinéma* , et depuis *AFP* seulement les tweets de *politique*. on obtient donc : $label(Michel, CNN) = \{IT, politics, movies\}$ et $label(Michel, AFP) = \{politics\}$.

La figure 3.1 présente le *LSG* pour l'exemple présenté.

3.3.4 La notification

Nous supposons une logique disjonctive pour la notification des utilisateurs : L'utilisateur est notifié lorsqu'il suit un compte qui publie un *tweet* qui correspond à au moins un des termes de son filtre. Une extension permettant de gérer la conjonction ainsi que la négation est abordée dans les pistes de recherche future. Formellement, nous définissions une notification

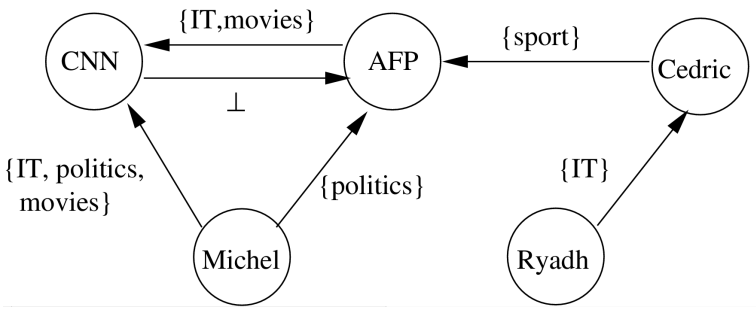

FIGURE 3.1 – Un graphe social étiqueté (LSG)

comme suit :

L'ensemble des nœuds $\mathcal{N}$ à être notifiés pour une nouvelle publication p à un nœud n est :

$$\mathcal{N}(n,p) = \{n' \in N | (n',n) \in E, \exists t \in p, t \in label(n',n)\}$$

Afin d'expérimenter nos approches de filtrage et de recommandation, nous avons du procéder à une étape d'acquisition en vue d'obtenir des jeux de données conformes au modèle proposé. Cette étape est décrite dans les sections qui suivent.

3.4 L'acquisition de données

Une des conclusions issue de notre analyse de l'état de l'art fût que, mis à part pour quelques rares travaux dont les équipes étaient en collaboration directe avec les fournisseurs de service, la collecte de données sur les plateformes sociales demeure problématique. En effet plusieurs facteurs rendent la constitution de jeux de données représentatifs complexe, voire dans certains cas impossible.

3.4.1 Le *crawling*, approche et limitations

La première approche que nous avons adopté dans notre tentative d'acquisition de données consistait à parcourir le graphe social de *Twitter* via

un programme spécifiquement conçu (*crawler*) afin de récolter la structure de graphe ainsi que les données associées aux utilisateurs. *Twitter* fournit en effet une interface publique de programmation (API) pour les applications désirant exploiter l'écosystème du réseau social. Généralement ces applications sont des clients qui permettent de porter le service *Twitter* sur plusieurs plateformes notamment sur les environnement mobiles.

L'API *Twitter* permet d'interroger le service et d'obtenir les description d'utilisateurs, les *tweets* ainsi qu'un ensemble de méta-données associés aux divers éléments récupérés. Le code affiché en 3.1 illustre l'anatomie simplifiée d'un objet *tweet* avec les principales méta-données associés. L'objet *tweet* complet tel que fourni par l'API contient lui plus de 28 méta-données (sans prendre en compte les données associées au compte propriétaire du *tweet*).

```
{
  "coordinates": null,
  "created_at": "Wed Jun 06 20:07:10 +0000 2012",
  "text": "Along with our new #Twitterbird, we've also updated our
      Display Guidelines: https://t.co/EdomjYs",
  "retweet_count": 66,
  "id": 210462857140252672,
  "entities": {
    "urls": [
      {
        "expanded_url": "https://dev.twitter.com/terms/display-
            guidelines",
        "url": "https://t.co/EdomjYs",
      }
    ],
    "hashtags": [
      {
        "text": "Twitterbird",
      }
    ]
  }
}
```

Code 3.1 – Vue simplifiée d'un objet *tweet* au format JSON

Avec l'évolution du service, l'approche commerciale de *Twitter* a changé et l'entreprise a commencé à cannibaliser le marché des clients pour dispositifs mobiles en rachetant les principaux acteurs du marché. Par exemple, le client *Twitter* sous iOS "Tweetie" a été racheté par le réseau social et a fini par devenir le client officiel de la plateforme sur *Iphone* [1] ou bien le logiciel populaire "Tweetdeck" qui fournit une vue en colonnes des flux sociaux a

1. https://blog.twitter.com/2010/twitter-iphone-0

vu son offre être incorporée à *Twitter* et devenir une des fonctionnalités du réseau social [2].

Dès lors, la stratégie de *Twitter* a évolué. Les données étant le bien le plus précieux pour la plateforme sociale, *Twitter* a mis en place une politique de limitation pour les requêtes possibles sur son API. Ces limitations ont très vite rendu impossible la constitution de jeux de données significatifs à partir de l'exploitation de l'API. Nous nous sommes heurtés à ces limitations ce qui nous a poussé à arrêter le *crawling* direct de la plateforme et a envisager d'autres approches.

De plus, le fait de mettre en place une stratégie de *crawling* en suivant les liens entre les utilisateurs récoltés dans un réseau aussi vaste que *Twitter* introduit un biais dans les données collectées et ne permet donc pas la génération d'un jeu de données représentatif. La figure 3.2 illustre un exemple de jeu de données récolté par notre *crawler*. Sur cette illustration nous pouvons observer un phénomène de "constellations" autour des comptes récoltés avec des sauts entre ces constellations qui correspondent aux liens suivis par la stratégie de *crawling*.

3.4.2 Constitution du jeu de données

Après maintes métamorphoses de son API publique, *Twitter* a publié une API nommée "Streaming API" [3]. Cette API est en réalité un flux constant de *tweets* représentant approximativement 1% du flux global circulant sur la plateforme.
En collaboration avec Forth Institute (Heraklion, Crete) [4], nous avons pu obtenir des données *Twitter* sur une période de plus de quatre mois en utilisant cette API ce qui nous a permis de constituer une collection de plus de 170 millions de *tweets*.

En parallèle, nous avons pu obtenir le graphe social représentatif de l'ensemble du réseau de *Twitter* utilisé dans [Kwak et al., 2010]. Dès lors, nous avons pu générer un jeu de données complet (*graphe + tweets*) et ce en fusionnant les tweets obtenus par l'API de streaming avec la structure de graphe social ce qui correspond à "peupler" le graphe avec les *tweets* des

2. https://blog.twitter.com/2011/official-tweetdeck-has-been-acquired-twitter
3. https://dev.twitter.com/streaming/overview
4. Nous remercions Vassilis Christophides pour son aide et son implication

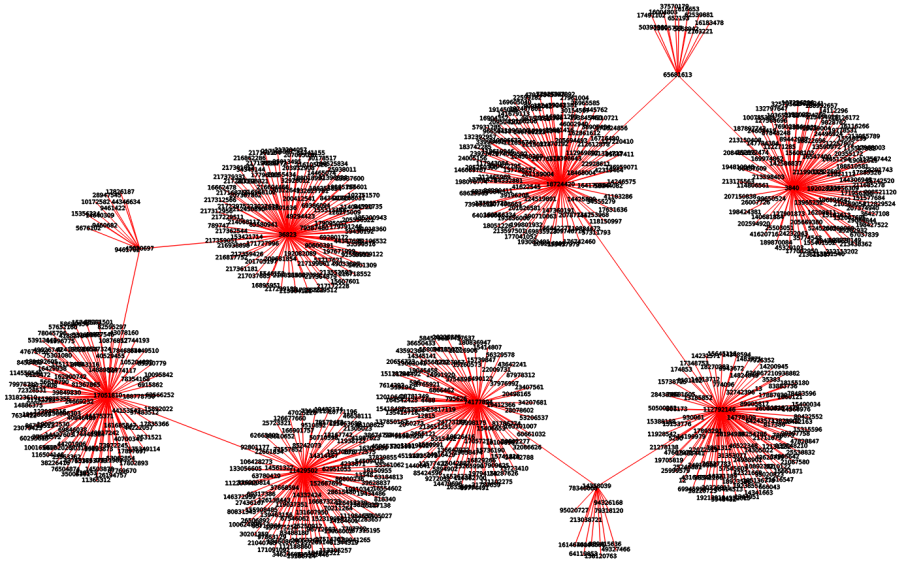

Figure 3.2 – Visualisation d'un extrait du graphe social obtenu par *crawling*

utilisateurs. Le résultat obtenu fut un large jeu de données contenant 2.9 millions d'utilisateurs, 169 millions d'arcs de graphe (relations de suivi) et plus de 33.9 millions de *tweets*.

Enfin, nous avons procédé à une analyse linguistique des publications pour n'en retenir que celles qui étaient rédigées en Anglais ainsi que les comptes associés. Les chiffres du jeu de données obtenu sont résumés dans le tableau 3.1.

Élément	Valeur
Utilisateurs	2.170.784
Tweets	15.717.449
Arcs de graphe	148.508.857

Table 3.1 – Description du jeu de données

Ce jeu de données nous a servi de base pour l'élaboration de jeux de test

plus spécifiques afin de tester nos approches de filtrage et de recommandation.

3.5 Jeu de données pour le Filtrage

Le but de notre système de filtrage est de notifier efficacement les utilisateurs de l'arrivée d'une publication contenant les filtres spécifiés. Afin de générer une structure de *LSG* pour tester notre système de filtrage, nous avons dû simuler des requêtes de filtrage. Pour ce faire, nous avons fait l'hypothèse que la taille moyenne des filtres (nombre de termes distincts d'une étiquette sur un arc du *LSG*) correspond au nombre moyen de termes utilisés sur les recherches dans *Twitter* [Teevan et al., 2011] (voir 4.2).

Après avoir éliminé les *URLs* et autres termes du langage courant (*stop-words*) des *tweets*, nous avons décidé de générer les filtres parmi les termes les plus fréquents et significatifs dans les publications du *publisher* dans un arc. Le tableau 3.2 illustre un échantillon du *labelled social graph* généré dans un contexte de filtrage.

Publisher	Follower	Filtre
12	36255965	conference deadline download
12	36256156	DB conference
12	36256607	software twitter

Table 3.2 – Exemple de données du LSG pour le filtrage

Le graphe social étiqueté obtenu nous a permis de tester le comportement de nos structures de filtrage dans des scénarios de passage à l'échelle. Ces expérimentations sont décrites en détail au chapitre 4. Cependant, dans une logique de recommandation, la génération du jeu de données nécessaire à la validation est plus complexe. Ceci est due à l'étape nécessaire de validation de la qualité des recommandations générées.

3.6 Jeu de données pour la recommandation

Basé sur la structure de graphe social généré précédemment, nous avons dû élaborer un jeu de données spécifique afin de tester notre système de recommandation. La qualité des recommandations générées étant basée en partie sur la sémantique des liens de suivi (*following)*, le défi consiste à étiqueter correctement les arcs du graphe social en fonction des intérêts des utilisateurs.

A partir du graphe social, nous avons d'abord extrait un sous-graphe représentatif de 400.000 nœuds en adoptant l'approche décrite par [Sethu and Chu, 2012]. Le tableau 3.3 décrit les principales propriétés topologiques du sous-graphe obtenu.

Propriété	Valeur
Nombre total de nœuds	399,999
Nombre total d'arcs	2,848,915
Plus court chemin moyen	3.74
Degré sortant moyen	9.12
Degré entrant moyen	7.12
LCC moyen	0.144
Diamètre du graphe	17
Degré entrant maximum	103,153
Degré sortant maximum	65,874

Table 3.3 – Propretés topologique du graphe Twitter400k

La valeur relativement élevée du local clustering coefficient (LCC) couplée à une valeur de plus court chemin moyen de 3,74 confirme le fait que notre sous-graphe présente les caractéristiques d'un réseau du type petit monde (*small world*, voir 2.2.1) avec des valeurs de propriétés topologiques similaires à celles observées pour le graphe réel de *Twitter* [Myers et al., 2014].

La figure 3.3 affiche les distributions de degré sortant/entrant pour tout les nœuds du sous-graphe obtenu. Nous pouvons observer un phénomène de longue traîne qui indique une distribution du type "loi de puissance" (*power law*) caractéristique des réseaux sociaux. Un tout petit nombre des comptes

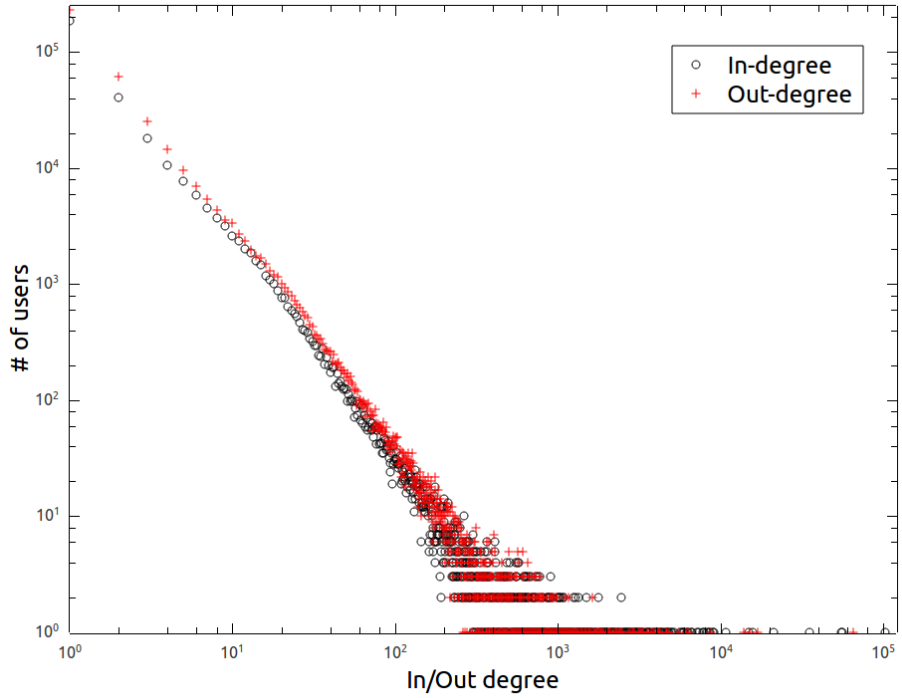

FIGURE 3.3 – Twitter400k Degré entrant/sortant

dispose d'un très grand nombre de *followers* (resp. suit un grand nombre de comptes) pendant que la plupart des comptes font partie de la classe des petits *followers* (resp. suivent peu de comptes). La figure 3.4 présente une visualisation d'un sous graphe social de 10.000 nœuds réalisé en adoptant la même approche (la taille des nœuds sur l'image est proportionnelle à leur nombre de *followers*).

Afin de procéder à l'étiquetage du graphe social obtenu, nous avons utilisé OpenCalais [5] pour caractériser les publications (*tweets*) de chacun des nœuds du jeu de données. OpenCalais est un service en ligne (API) fourni par *Thomson Reuters* qui permet d'extraire automatiquement des informations sémantiques sur des documents textuels. Le résultat de cette étape fût l'étiquetage des comptes avec un *topic* basé sur les *tweets* publiés. L'étiquetage

5. http://www.opencalais.com/

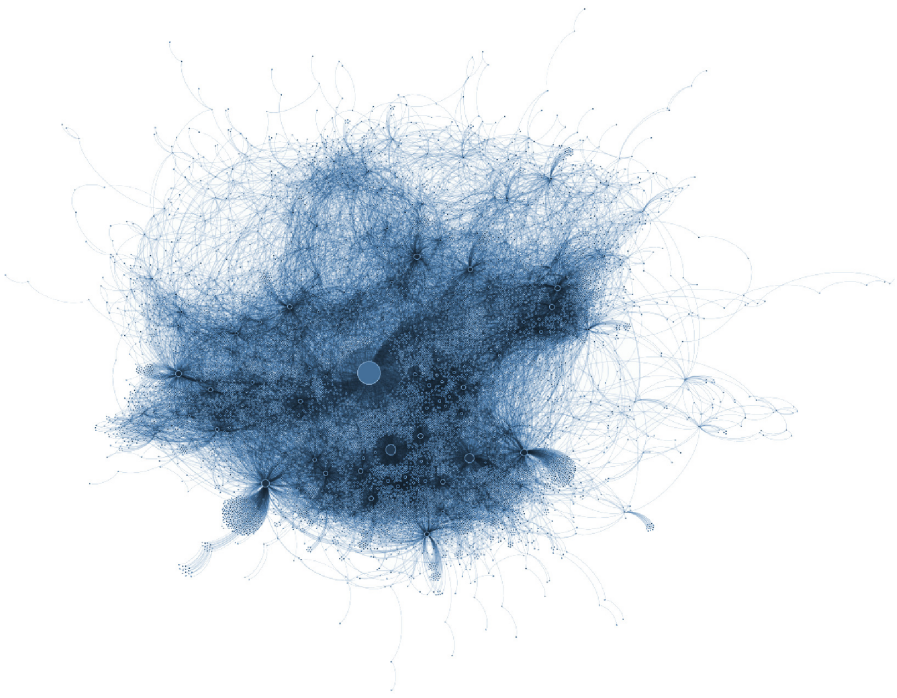

Figure 3.4 – Visualisation d'un graphe social de 10.000 nœuds

s'est fait en utilisant une liste de 18 *topics* standards utilisés habituellement pour classer des articles/documents [6]. Cette approche a permis de catégoriser les deux tiers des comptes sur notre graphe social.

Afin de compléter cette étape, nous avons utilisé une classificateur basé sur la méthode de classification probabiliste naïve bayésienne. Nous nous sommes basés sur un pack logiciel appelé WEKA [7] pour l'étape d'apprentissage et de classification. La précision du classificateur obtenu a été de 0,83. Le résultat de cette étape fut une liste contenant tout les comptes de notre graphe social avec leur profil de publieur (*topics* caractérisants ses publications). Le profil de *follower* quant à lui (profil d'intérêt de chacun des

6. http://www.opencalais.com/documentation/calais-web-service-api/api-metadata-/document-categorization

7. http://www.cs.waikato.ac.nz/ml/weka/

utilisateurs) a été déterminé en calculant le TF (fréquence de termes) sur les profils publieurs des comptes suivis par un compte donné.

Enfin, nous avons sélectionné le *topic* à choisir comme étiquette pour chaque arc comme étant le *topic* dans l'intersection d'un profil *follower* avec le profil publieur correspondant ayant le plus grand score dans le profil *follower*. Le résultat est un graphe social étiqueté sur chaque arc. Nous notons ce jeu de données *"Twitter400k"*.

Notre génération d'étiquettes sur les arcs a engendré la distribution de *topics* décrite dans la figure 3.5 pour les 18 *topics*. Cette distribution biaisée demeure similaire à certaines distributions de termes observées dans un contexte Web comme par exemple sur *Yahoo ! Directory* [Liu et al., 2005].

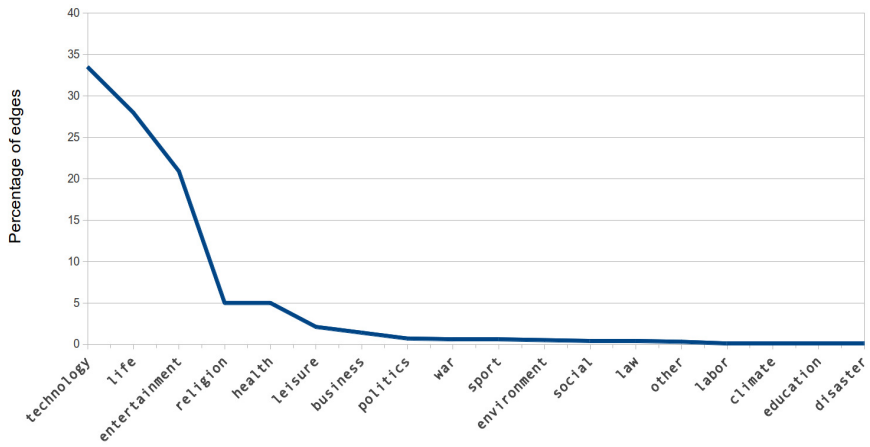

Figure 3.5 – Distribution des arcs par *topic*

Ce jeu de données étiqueté prend en compte les intérêts des utilisateurs du graphe social ainsi que leur profil publieur et nous permet ainsi de valider l'aspect qualitatif du système de recommandation proposé et décrit en chapitre 5.

3.7 Conclusion

Dans ce chapitre nous avons présenté le modèle de données que nous proposons pour représenter un système du type micro-blogging. Ce modèle de graphe étiqueté (LSG) permet de capturer l'intérêt des utilisateurs pour les comptes qu'ils suivent. Par la suite, nous avons décrit les différentes étapes qui nous ont menés à la constitution de jeux de données représentatifs. Nous avons illustré les difficultés liées à la récolte de données ainsi que la différence quand à la constitution d'un jeu de données dans un contexte de filtrage et un contexte de recommandation.

Le chapitre suivant décrit en détail l'approche que nous avons proposée afin de fournir un filtrage efficace des flux sociaux et ainsi améliorer l'expérience des utilisateurs des plateformes de micro-blogging.

Chapitre 4

Filtrage et indexation

4.1 Introduction

Dans le but d'améliorer l'expérience utilisateur ainsi que de réduire le nombre de publications en circulation sur le réseau, nous avons décidé d'introduire la notion de filtrage dans les systèmes de type micro-blogging. Dans le sondage réalisé par [Bontcheva et al., 2013], 55,6% des utilisateurs interrogés on senti le besoin d'un outil qui leur permettrait de filtrer leur flux social et d'en écarter les publications non pertinentes. L'idée principale derrière notre approche est de pouvoir suivre un certain compte du système tout en réduisant le nombre de *tweets* reçus par l'application d'une requête de filtrage. Nous avons appelé notre système de filtrage MicroFilter.

Dans ce chapitre, nous introduisons d'abord la motivation ainsi les différentes structures proposées pour effectuer un filtrage efficace en se basant sur le modèle de données présenté en Chapitre 3. Pour chacune de ces structures, est présentée ensuite une analyse qui permet d'établir un modèle analytique de coût en temps et en mémoire pour les différentes propositions. Par la suite nous présentons le comportement de nos structures observé expérimentalement et qui nous a permis de proposer une structure hybride qui offre un compromis intéressant. Ce chapitre se referme par une conclusion qui résume les différentes contributions de notre approche de filtrage ainsi que certaines pistes d'améliorations possibles.

4.2 L'indexation des filtres

Actuellement, les schémas d'indexation utilisés dans les systèmes de micro-blogging comme *Twitter* permettent de récupérer efficacement pour un utilisateur n l'ensemble $\Gamma^-(n)$ de ses *followers* dans le graphe social G. Ces systèmes reposent essentiellement sur un index exploitant une table de hachage sur l'identifiant du nœud. Cette technique permet de récupérer la liste des suiveurs (*followers*) d'un nœud (voir [Weaver] pour un aperçu de l'architecture de Twitter).

[Teevan et al., 2011] rapporte une taille de 1.64 termes pour les recherches sur le moteur de recherche de *Twitter*. Nous supposons une taille de filtre similaire à cette valeur, ce choix est motivé par :

— (i) Le filtrage correspond à la préférence d'un utilisateur vis-à-vis des publications d'un compte donné. En effet, ce scénario diffère du cas habituel où le filtrage est appliqué au flux global. Nous supposons que les besoins en filtrage sur un compte spécifique sont plus précis.

— (ii) Nous rencontrons souvent dans le monde du micro-blogging des utilisateurs qui détiennent une autorité sur le graphe social sur un sujet (*topic*) précis [Pal and Counts, 2011]. *Twitter* recommande déjà des comptes à suivre sur le réseau pour certains *topics*. Avec notre système, les utilisateurs peuvent mettre en place des filtres pour recevoir uniquement les *tweets* correspondant à ces *topics*.

Étant donné cette taille de filtre sur un système utilisé simultanément par plusieurs centaines de millions d'utilisateurs, le défi consiste à déterminer efficacement l'ensemble $\Gamma^-(n)$ de *followers* destinataires d'une publication donnée en se basant sur la structure de *LSG*. Cette problématique est d'autant plus délicate pour les comptes avec un large nombre de *followers*.

Nous proposons de comparer 3 différentes structures d'indexation qui exploitent un stockage du graphe social sous forme de liste inverse pour gérer le filtrage. Pour satisfaire la notification en temps-réel, en prenant en considération le débit élevé de publications (*ex. Twitter* a relevé des pics à plus de 8800 *tweets* par seconde en 2011 [1] et de plus de 143.000 TPS en 2013 [2]), nous

1. http://yearinreview.twitter.com/en/tps.html
2. https://blog.twitter.com/2013/new-tweets-per-second-record-and-how

nous basons sur des structures qui peuvent être maintenues en mémoire centrale. Cette raison élimine les structures arborescentes (Voir [Hmedeh et al., 2012] pour une comparaison des structures basées sur le hachage et les solutions arborescentes). Nos propositions sont basées sur des structures de listes inverses qui exploitent une factorisation par rapport aux comptes publieurs, aux suiveurs ou bien aux filtres. Ces structures peuvent être facilement déployées comme extension des structures existantes déjà implantés sous forme de listes inverses.

Nos trois variantes de listes inverses exploitent trois facteurs de factorisation différents : les identifiant de *followers*, les identifiants de *publishers* ou les termes utilisés pour le filtrage. Du fait que nous stockons des identifiants de termes et non les termes eux-mêmes (grâce à l'utilisation d'une table de correspondance) nous considérerons dans ce qui suit que toutes les entrées, les identifiant de *followers*, les identifiants de *publishers* ou les identifiants de termes nécessitent le même espace de stockage en mémoire (un entier sur *8-octets* dans notre implantation afin de gérer les identifiant de *Twitter*).
L'ensemble des paramètres qui impactent le temps de construction des index, l'espace mémoire ainsi que les temps de recherche (*matching*) sont résumés dans le tableau 4.1.

N	Le nombre total de comptes
$\Gamma^+(n)$	Comptes suivis par n
$\Gamma^-(n)$	Comptes qui suivent n
φ	Nombre moyen de *followers* pour un utilisateur
τ	Nombre moyen de termes de filtre pour une paire (*publisher*, *follower*)
$\|p\|$	Taille d'une publication (termes distincts) p
(k, β)	Coefficients de Heaps pour les données de type micro-blogging
γ	Constante de la loi de Zipf pour les données de type micro-blogging
$\mathcal{V}_F$	Vocabulaire des filtres
θ_{dir}	Taille d'une entrée de *directory*
θ_{list}	Taille d'une *posting list*
θ_{entry}	Taille d'une entrée d'une *posting list*

Table 4.1 – Table des notations

4.3 Structures et modèle analytique

4.3.1 Le PFT-index

L'index PFT (pour $Publisher - Follower - Term$ index) est un *mapping* (correspondance) qui a pour clé un compte $n \in N$, et pour valeur la *posting list* correspondante $Postings_{PFT}(n)$, *c.à.d.*, l'ensemble des *followers* ainsi que leurs filtres : $Postings_{PFT}(n) = \{(n_1, \{t^1_{n_1}, t^2_{n_1}, \ldots\}), (n_2, \{t^1_{n_2}, t^2_{n_2}, \ldots\}), \ldots\}$, avec $n_i \in \Gamma^-(n)$ et $t^j_{n_i} \in label(n_i, n)$. Le PFT-index correspond à une factorisation d'abord sur chaque *publisher*, puis pour chaque *publisher* une seconde factorisation sur les identifiants de *followers*.

Exemple

La Figure 4.1 illustre le PFT-index correspondant à l'exemple présenté en section 3.3.1.

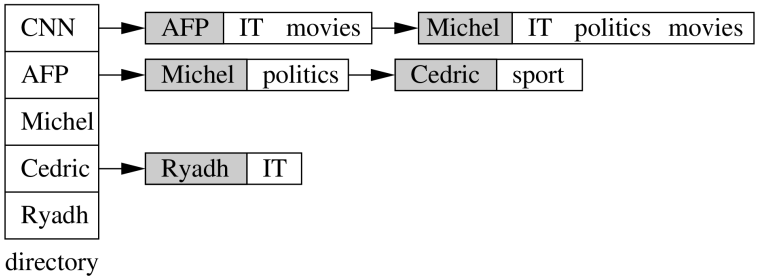

Figure 4.1 – Le PFT-index

Occupation mémoire du PFT-index

Soient φ le nombre moyen de *followers* pour un utilisateur du système et τ le nombre moyen de termes d'un filtre pour une paire (publisher, follower). L'index consiste en un *key directory* dont les *posting lists* contiennent les identifiants des *followers* ainsi que les termes. L'occupation mémoire pour le PFT-index pour un système représenté par un graphe LSG avec $|N|$ utilisateurs est :

$$\Delta^{PFT}_{memory}(LSG) = size(directory) + total(f\_id) + total(terms)$$

La taille du *directory* est le nombre de *publishers* qu'on suppose égal à N (chaque compte a au moins un *follower*). Le nombre d'identifiants de

followers total(f_id) présents dans la structure est $N \times \varphi$, et le nombre total de termes $total(terms)$ indexés est $N \times \varphi \times \tau$. On déduit donc :

$$\Delta^{PFT}_{memory}(LSG) = N \times \theta_{dir} + (N \times \varphi) \times \theta_{list} + (N \times \varphi \times \tau) \times \theta_{entry}$$

Temps de recherche (*matching*) du PFT-index

En supposant un *tweet* p de taille $|p|$ publié par l'utilisateur n. Le processus de notification accède à la *posting list* $Postings_{PFT}(n)$ et pour chaque *follower* n_i vérifie s'il existe un terme t_j dans *tweet* p tel que $label(n_i)$ contient t_j. Le temps moyen de *matching* est donc :

$$\Delta^{PFT}_{time}(LSG, p) = \varphi \times (|p| \times \tau)$$

Temps d'insertion/suppression du PFT-index

Pour insérer un nouveau filtre nous devons parcourir la *posting list* $Postings_{PFT}(n)$ jusqu'à trouver l'identifiant du *follower* à ajouter ou supprimer. Si le *follower* n'existe pas dans $Postings_{PFT}(n)$, une nouvelle entrée est alors ajoutée. La suppression nécessite un parcours additionnel de la liste de termes. Potentiellement, ceci peut conduire à la suppression de l'entrée du *follower*. Les coûts sont dont respectivement :

$$\Delta^{PFT}_{insert}(LSG) = \varphi/2$$

$$\Delta^{PFT}_{delete}(LSG) = \varphi/2 + \tau/2$$

4.3.2 Le PTF-index

Dans le PTF-index, (pour $Publisher - Term - Follower$ index), la clé est cette fois-ci un compte $n \in N$ et la valeur est la *posting list* correspondante $Postings_{PTF}(n)$. La *posting list* est factorisée sur les termes, chaque terme t est donc associé à une liste des *followers* de n qui ont choisi t comme filtre aux *tweets* de n. Ce qui nous donne $Postings_{PTF}(n) = \{(t_1, \{n^1_{t_1}, n^2_{t_1}, \ldots\}), (t_2, \{n^1_{t_2}, n^2_{t_2}, \ldots\}), \ldots\}$, avec $n^j_{t_i} \in \Gamma^-(n)$ et $t_i \in label(n_i, n)$.

Exemple

La Figure 4.2 illustre le PTF-index correspondant à l'exemple présenté en section 3.3.1.

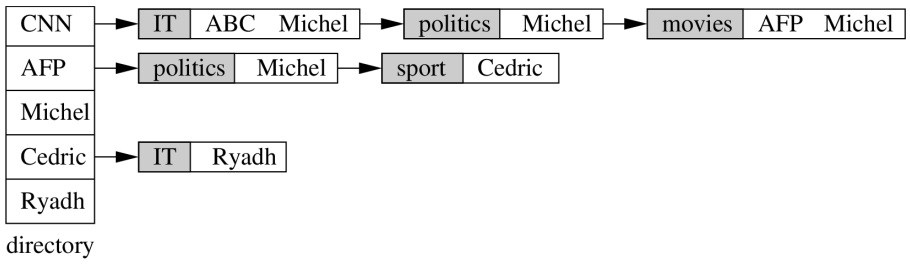

FIGURE 4.2 – Le PTF-index

Occupation mémoire du PTF-index

L'index est composé de N *posting lists*, chaque entrée doit stocker $\varphi \times \tau$ filtres associés à un *publisher*, comme dans le PFT-index, avec les termes pour critère de factorisation. Si l'on suppose que tout les *followers* d'un *publisher* posent des filtres distincts, la taille de $Postings_{PTF}(n)$ est $\varphi \times \tau$. Cependant nous observons que les *followers* expriment des intérêts similaires dans leur comportement en suivant un *publisher* particulier. Par conséquent le nombre de filtres distincts sur un *publisher* doit être inférieur à cette valeur.

Nous supposons comme dans plusieurs applications basées sur les mots-clés que le nombre total de termes dans une *posting list* suit une *loi de Heaps* [Baeza-Yates and Ribeiro-Neto, 1999, Manning et al., 2008], *c.à.d.* $|Postings_{PTF}(n)| = k \times T^{\beta}$, où k et β sont des constantes et T est le nombre total de termes dans le *posting*. Les coefficients de Heaps k et β dépendent fortement des caractéristiques du corpus de texte analysé et leurs valeurs pour ce qui concerne le micro-blogging doit être déterminées dans un travail futur. A noter que β est entre 0 et 1 (généralement entre $[0.4, 0.6]$). Ce qui implique que plus le nombre de *followers* augmente, meilleure est la factorisation. Ce comportement est attendu dans notre système où un grand nombre d'utilisateurs partagent le même filtre. Étant donné que le nombre de termes dans $Postings_{PTF}(n)$ est $N \times (\varphi \times \tau)$ et le nombre d'entrées indexées est toujours $N \times \varphi \times \tau$, on déduit que :

$$\Delta^{PTF}_{memory}(LSG) = N \times \theta_{dir} + (N \times k \times (\varphi \times \tau)^{\beta}) \times \theta_{list} + (N \times \varphi \times \tau) \times \theta_{entry}$$

Temps de matching du PTF-index

Supposons un nouveau *tweet* p publié par l'utilisateur n. On accède d'abord à la *posting list* du *publisher* n $Postings_{PTF}(n)$. puis pour chaque entrée $(t_i, \{n^1_{t_i}, \ldots, n^k_{t_i})$ on vérifie si p contient t_i. A chaque fois que cela arrive, on notifie chaque $n^j_{t_i}$ depuis l'entrée t_i. Le temps de matching moyen est donc donné par :

$$\Delta^{PTF}_{time}(LSG, p) = |p| \times k \times (\varphi \times \tau)^{\beta}$$

Temps d'insertion/suppression du PTF-index

Pour insérer un nouveau filtre t, nous parcourons la *posting list* $Postings_{PTF}(n)$ jusqu'à trouver l'entrée qui correspond à t. Ajouter un nouveau filtre consiste à insérer l'identifiant du *follower* qui exprime ce filtre t dans la liste adéquate. Si le terme n'existe pas dans $Postings_{PTF}(n)$, une nouvelle entrée pour ce terme est ajoutée. La suppression nécessite un parcours additionnel de la liste de *followers*, en moyenne $(\varphi \times \tau)/|Postings_{PTF}(n)|$, et peut conduire potentiellement à la suppression de l'entrée du terme. Les coûts sont donc respectivement :

$$\Delta^{PTF}_{insert}(LSG) = k \times (\varphi \times \tau)^{\beta}/2$$

$$\Delta^{PTF}_{delete}(LSG) = k \times (\varphi \times \tau)^{\beta}/2 + (\varphi \times \tau)/(k \times (\varphi \times \tau)^{\beta})$$

4.3.3 Le TPF-index

Dans le TPF-index, (pour $Term-Publisher-Follower$ index), la clé est un terme apparaissant dans un filtre. Donc le *directory* contient tout le vocabulaire de filtres $\mathcal{V}_F$. La *posting list* $Postings_{TPF}(t)$ associée à un terme t est l'ensemble des *publishers* avec leurs *followers* qui filtrent le *publisher* sur le terme t : $Postings_{TPF}(t) = \{(n_1, \{n^1_{n_1}, n^2_{n_1}, \ldots\}), (n_2, \{n^1_{n_2}, n^2_{n_2}, \ldots\}), \ldots\}$, avec $n^j_i \in \Gamma^-(n_i)$ et $t \in label(n_i, n^j_i)$. TPF-index correspond à une factorisation d'abord sur le terme de $\mathcal{V}_F$, puis pour chaque terme une seconde factorisation sur l'identifiant du *publisher*.

Exemple

La figure 4.3 illustre le TPF-index correspondant à l'exemple présenté en section 3.3.1.

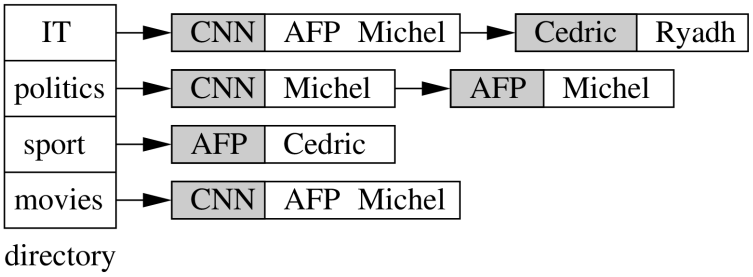

Figure 4.3 – Le TPF-index

Occupation mémoire du TPF-index

Le TPF-index consiste en $|\mathcal{V}_F|$ *posting lists*, une pour chaque terme du vocabulaire des filtres. Sachant que nous avons N comptes qui suivent en moyenne φ autres comptes avec une taille moyenne de filtre de τ, le nombre total de termes utilisés pour le filtrage est $N \times \varphi \times \tau$. Nous supposons toujours que la taille du vocabulaire suit une loi de *Heaps* donc le nombre de termes distinct est $|\mathcal{V}_F| = k \times (N \times \varphi \times \tau)^{\beta}$.

Nous supposons aussi que la distribution des termes dans l'ensemble de filtres suit une *loi de Zipf* [Baeza-Yates and Ribeiro-Neto, 1999, Manning et al., 2008], et que le nombre de *publishers* dont les *tweets* sont filtrés pour un terme donné est proportionnel à la fréquence du terme. Par conséquent, le nombre de *publishers* associés à un terme t_i dont le rang de fréquence est r_i est γ/r_i, où γ est une constante.

Ici aussi $N \times \varphi \times \tau$ entrées sont stockées, ce qui correspond aux identifiants des *followers*. Donc l'occupation mémoire pour le TPF-index est :

$$\Delta^{TPF}_{memory}(LSG) = |\mathcal{V}_F| \times \theta_{dir} + (\sum_{i=1}^{|\mathcal{V}_F|} \gamma/i) \times \theta_{list} + (|N| \times \varphi \times \tau) \times \theta_{entry}$$

Sachant que $|\mathcal{V}_F| \gg 1$, nous pouvons approximer $\sum_{i=1}^{|\mathcal{V}_F|} \gamma/i$ avec $\gamma \times ln(|\mathcal{V}_F|)$. on obtient donc :

$$\Delta^{TPF}_{memory}(LSG) = (k \times (N \times \varphi \times \tau)^{\beta}) \times \theta_{dir}$$
$$+(\gamma \times ln(k \times (N \times \varphi \times \tau)^{\beta})) \times \theta_{list} + (N \times \varphi \times \tau) \times \theta_{entry}$$

Temps de matching du TPF-index

Supposons un nouveau tweet p publié par l'utilisateur n. Pour chaque terme $t \in p$ on accède à la posting list de t $Postings_{TPF}(t)$. Puis pour chaque entrée $(n_i, \{n^1_{n_i}, n^2_{n_i}, \ldots\})$ on vérifie si n_i est le *publisher* de p, *c.à.d.*, si $n_i = n$. Si une entrée existe, on notifie alors $n^j_{n_i}$ et on arrête le parcours pour ce terme. Supposant la distribution Zipf des termes, la taille moyenne d'une posting list est de $ln(|\mathcal{V}_F|)/2$, et on parcourt en moyenne la moitié pour trouver le publisher. Le temps moyen de matching est donc :

$$\Delta^{TPF}_{time}(LSG, p) = |p| \times (\gamma \times ln(k \times (N \times \varphi \times \tau)^{\beta})/2)/2$$

Temps d'insertion/suppression du TPF-index

Pour insérer un nouveau filtre (n, n', t) nous devons parcourir la posting list $Postings_{TPF}(t)$ dont la taille est en moyenne $ln(|\mathcal{V}_F|)/2$ jusqu'à trouver l'entrée qui correspond à n. Ajouter un nouveau filtre consiste à insérer l'identifiant du *follower* qui pose le filtre t dans la liste qui correspond. Si le terme n'existe pas dans $Postings_{TPF}(t)$, une nouvelle entrée est rajoutée. La suppression nécessite un parcours additionnel de la liste des *followers*, en moyenne $(N \times \varphi \times \tau)/|\mathcal{V}_F|$ ce qui peut conduire potentiellement à la suppression d'une entrée d'un *publisher*. Par conséquent, les coûts sont de respectivement :

$$\Delta^{TPF}_{insert}(LSG) = (\gamma \times ln(k \times (N \times \varphi \times \tau)^{\beta})/2)/2$$

$$\Delta^{TPF}_{delete}(LSG) = (\gamma \times ln(k \times (N \times \varphi \times \tau)^{\beta})/2)/2 + (N \times \varphi \times \tau)/|\mathcal{V}_F|)$$

4.4 Expérimentations

Dans cette section, nous présentons les expériences que nous avons conçues et conduites afin d'analyser (*i*) comment se comportent nos structures sur des jeux de données réels collectés sur *Twitter* et (*ii*) l'impact des différents paramètres des systèmes de micro-blogging. Ces expériences qui valident le modèle analytique présenté plus tôt ont été conduites sur un processeur Intel Core i5 avec une fréquence de 3.60 GHz et 16 GB de RAM. Les structures ont été implantés en JAVA.

4.4.1 Espace mémoire

Les différents critères de factorisation font que nos structures ont des besoins mémoire différents. Le tableau 4.2 compare l'occupation mémoire des trois structures sur notre jeu de données pour le filtrage. *TPF*-index apparait comme étant la structure la moins coûteuse en espace mémoire. Plusieurs filtres sont partagés par un grand nombre d'utilisateurs ce qui nous a permis d'obtenir une meilleure factorisation, sur les termes d'abord. Mais aussi, on observe que beaucoup de *followers* filtrent un publisher sur le même terme. Par conséquent, pour un terme dans le *TPF*-index, on observe aussi une importante factorisation sur l'identifiant du *publisher*, en particulier pour les comptes à forte popularité (avec un grand nombre de followers). De manière opposée, le *PFT*-index ne bénéficie pas d'une bonne factorisation vu que tous les *publishers* ont une entrée dans le *directory* et pour chaque entrée on a une liste de termes pour chacun de ses *followers*.

Structure	Taille d'index (MB)
PFT	9021
PTF	3777
TPF	1869

Table 4.2 – Tailles des index

Pour mesurer l'impact des différents paramètres et ainsi valider notre modèle analytique, nous avons généré des jeux de données synthétiques avec des nombres constants de filtres (τ) pour chaque arc de graphe. Les résultats sont reportés sur la figure 4.4.
L'occupation mémoire croit linéairement avec τ pour les index PFT et PTF. En effet, faire croitre τ n'impacte pas la taille du *directory* qui dépend seulement du nombre de *publishers*, *c.à.d.*, N. En outre pour le PFT-index, le nombre d'éléments dans la *posting list* reste constant et égal au nombre de *followers*. Par contre, le nombre d'entrées suit linéairement τ. Pour le PTF-index le nombre d'éléments dépend du nombre de termes distincts utilisés comme filtres pour un publisher. En comparant avec le PFT-index on observe la même pente. Ceci révèle que τ a un faible impact sur le nombre d'éléments pour une *posting list* dans le PTF-index.

La loi de *Heaps* proposée dans notre modèle explique ce résultat, car elle suppose que le sous-vocabulaire des termes de filtres pour un *publisher* donné

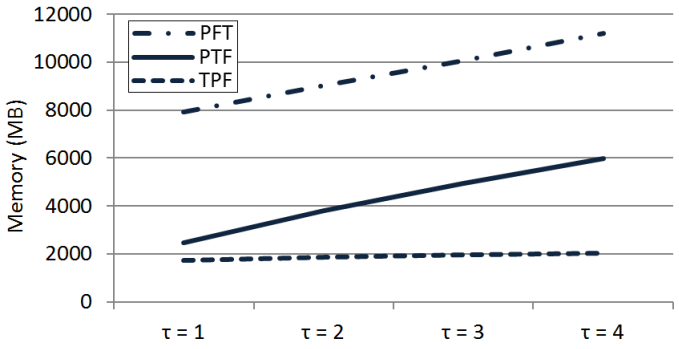

Figure 4.4 – Espace mémoire / τ

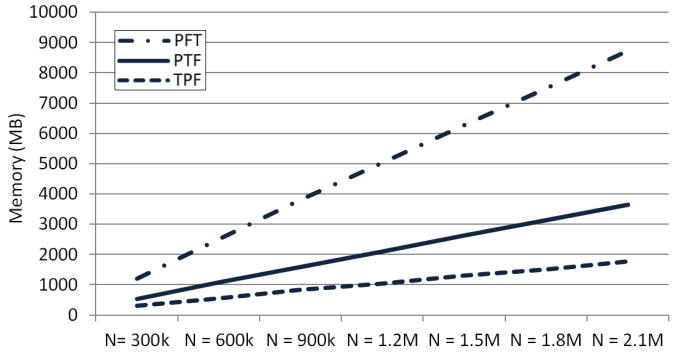

Figure 4.5 – Espace mémoire / N

augmente légèrement. Ainsi pour le PTF-index, comme pour le PFT-index, la croissance τ n'impacte pas la structure mais seulement le nombre d'entrées. Finalement, nous notons l'impact faible de τ sur le TPF-index. La raison est (i) la taille du directory reste constante et égale à $\mathcal{V}_F$, (ii) sachant que les filtres sont générés sur les tweets d'un publisher, un nouveau terme a une grande probabilité d'être déjà présent pour le même publisher dans la structure, donc le nombre d'éléments des listes demeure bas et augmente lentement.

La figure 4.5 illustre l'impact de N sur les différentes structures. Toutes les structures affichent une croissance linéaire. Pour le PFT et le PTF index, ajouter un nouvel utilisateur conduit à ajouter une nouvelle entrée de

directory, une nouvelle posting list et de nouvelles entrées dans la posting list pour ses followers avec leurs filtres. Cependant, la factorisation sur les identifiants de termes dans une posting list est plus efficace que celle sur les identifiants de followers ce qui explique la meilleure pente pour le PTF. En ce qui concerne le TPF, après une courte étape d'initialisation qui correspond à la création des différentes entrées du directory et des différents éléments des posting lists, l'augmentation de N implique seulement de rajouter de nouvelles entrées de liste. Ceci explique la croissance linéaire avec une faible pente. Les deux figures valident le modèle analytique présenté précédemment.

4.4.2 Temps d'indexation

Dans le tableau 4.3, nous comparons le temps nécessaire à la construction des différentes structures en mémoire centrale pour les différents jeux de données. Uniform(1), (2) and (3) correspondent aux scénarios où chaque arc publisher-follower est associé à respectivement 1, 2 ou 3 termes de filtres. Comme prévu, plus la taille de l'index est petite, moins de temps est dépensé en sa construction. En effet pour le TPF-index nous avons de plus petites posting lists, ce qui implique que nous avons besoin de moins de temps pour trouver l'élément de la liste où le nouveau filtre doit être inséré. Par contre, dans le PFT-index, un coûteux parcours de larges posting lists est nécessaire. Nous notons aussi que le temps de construction n'est pas proportionnel à τ. Il s'avère que les éléments de posting lists sont crées rapidement et leur nombre évolue lentement. Après cette étape, augmenter le nombre de termes de filtre pour un follower correspond essentiellement à ajouter de nouvelles entrées dans des éléments existants ce qui se fait en temps relativement constant et qui explique le comportement linéaire.

Structure	réaliste	uniform(1)	uniform(2)	uniform(3)
PFT	753	736	741	1081
PTF	512	357	444	558
TPF	231	198	236	289

Table 4.3 – Temps d'indexation (en s)

4.4.3 Temps de recherche (matching)

Pour l'évaluation des temps de matching nous procédons comme suit : Tout d'abord, le tweet est décomposé en un ensemble de termes, puis nous utilisons l'index pour déterminer pour chaque terme du tweet l'ensemble des followers à notifier. A noter que, dû au fait que nous travaillons dans une logique disjonctive, quand le premier matching positif se manifeste nous pouvons directement notifier l'utilisateur correspondant. Le tableau 4.4 illustre les temps moyens de matching pour un flux de 100,000 tweets sur les différents index.

On observe que le TPF-index affiche de mauvaises performances de matching : pour notre jeu de données réaliste, avec une valeur de temps de matching de 2.5 ms par tweet, il peut gérer moins de 400 tweets par seconde donc insuffisant pour passer à l'échelle. Pour chaque terme du tweet nous récupérons une large posting list avec potentiellement autant d'éléments que de publishers N existants.

Par contre, le PTF-index récupère rapidement les followers à notifier : il peut traiter un tweet toutes les 15 μs, il est donc capable de gérer des pics à plus de 66.000 tweets par seconde. Dans ce cas de figure, nous accédons directement à la posting list correspondante au publisher. Puis, nous parcourons tous ses éléments ce qui correspond à tous les followers du compte et on vérifie pour chacun d'entre eux le matching avec les termes du tweet. A noter que le nombre des termes d'un filtre est petit (entre 1 et 3) pour un follower, et que nous faisons le test pour tous les termes du tweet en un seul parcours. Évidemment, ces résultats sont des valeurs moyennes et le temps de matching est plus élevé pour les publishers à forte popularité et plus rapide avec ceux qui ont très peu de followers.

Structure	PFT	PTF	TPF
Temps de matching (μs)	808	15	2564

Table 4.4 – Temps de matching

La figure 4.6 illustre l'impact de τ sur le temps de matching. Comme dans le tableau 4.4, le PTF-index surpasse les deux autres propositions de 2 ordres de grandeur. Nous notons que le temps de matching pour le

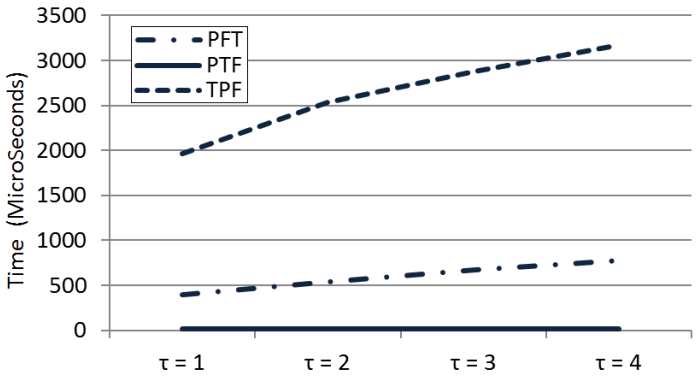

Figure 4.6 – Temps de matching / τ

PFT-index croit linéairement avec τ pendant que le TPF-index suit une croissance sous linéaire. Pour le PFT-index, le matching implique un accès direct à la posting list d'un publisher puis un parcours de tous ses éléments en vérifiant si les entrées associés correspondent aux termes du tweet. Augmenter la valeur de τ ne change ni le nombre d'entrées du directory, ni le nombre d'éléments des listes. Donc seulement à la dernière étape, la comparaison avec les termes nécessite un peu plus de temps. Le nombre d'entrées est proportionnel à τ ce qui explique la croissance linéaire.

En ce qui concerne le TPF-index, nous observons le comportement correspondant à la loi de Zipf dans la distribution des fréquences de termes, donc en augmentant τ nous rajoutons généralement des entrées dans la posting list des termes les plus fréquents. Une conséquence est que plus on a de filtres indexés, plus grande est la probabilité de rajouter une entrée dans une posting list existante. Dès lors que les tailles des posting lists ont une croissance sous-linéaire et que nous parcourons autant de listes que le nombre de termes dans le tweet, cela justifie que le temps de matching croit de manière sous linéaire par rapport à τ.

Nous illustrons aussi dans la Figure 4.7 l'évolution des temps de matching par rapport à N. Nous notons que à la fois le PFT et le PTF-index ont un temps de matching constant. En effet, augmenter N n'impacte que la taille du directory en y ajoutant de nouvelles entrées, mais les posting lists gardent un nombre constant d'éléments et pour chaque élément un nombre constant

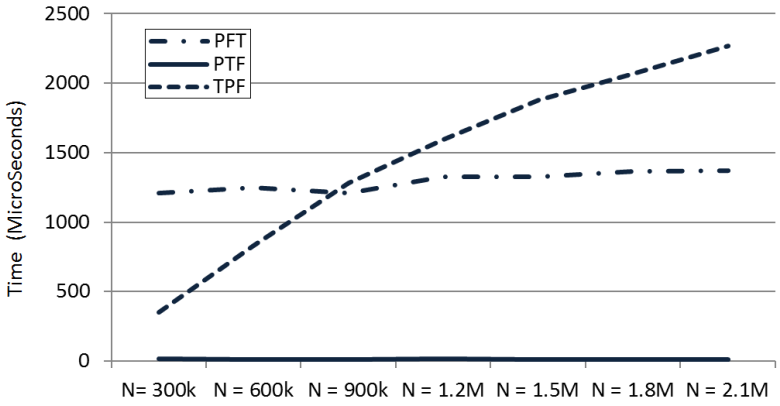

Figure 4.7 – Temps de matching / N

d'entrées. Comme pour un nouveau tweet nous parcourons seulement une seule posting list qui correspond au publisher, le temps de matching est constant avec N. Le TPF-index montre de meilleurs résultats que le PFT-index pour des valeurs de N inférieures à 900.000. Le temps de matching avec le TPF-index croit de manière sous-linéaire par rapport à N pour les mêmes raisons que pour τ. Ces résultats confirment les équations présentées dans notre modèle analytique.

4.4.4 Efficacité du filtrage

Afin d'évaluer l'impact du filtrage sur le nombre de tweets remis, nous avons décidé de comparer le nombre de tweets remis par le serveur de micro-blogging avec et sans la mise en place du filtrage. Nous avons testé le système de filtrage sur différentes variantes du jeu de données : Un jeu dit "réaliste" avec un nombre filtres par arêtes variant entre 1 et 3 et des jeux uniformes avec un nombre fixe de filtres sur chaque arête.

Comme attendu le nombre de tweets remis chute de manière importante. Nous mesurons un gain de presque 98% sur nos jeux de données (voir tableau 4.5). Notre logique disjonctive pour les filtres justifie que plus on a de termes dans les filtres, plus on doit délivrer de *tweets*. Cependant, nous n'observons pas de croissance linéaire dans le nombre de tweets à remettre. En effet, il existe une grande probabilité pour un tweet de correspondre à plusieurs termes d'un follower quand ce follower a des filtres de grande taille.

Filtrage	Réaliste	uniform(1)	uniform(2)	uniform(3)	uniform(4)
Activé	211,651	161,196	212,364	244,578	267,164
Désactivé	11,035,437				

Table 4.5 – Nombre de *tweets* remis

4.5 L'évolution du graphe social

Comme rapporté par [Kwak et al., 2011, Kivran-Swaine et al., 2011], les utilisateurs des plateformes de micro-blogging ont un comportement de suivi dynamique ce qui implique que le graphe social sous-jacent dans les plateforme comme *Twitter* est en constante évolution. Les relations de suivi se font et se défont au gré de l'intérêt des utilisateurs, intérêt qui peut aussi être temporaire (ex. les comptes créés pour des événements ponctuels). Dans le but de gérer au mieux les notifications de *tweets*, nous proposons une structure hybride qui tire parti des structures PFT et PTF et qui est présentée en section 4.6.

Afin d'illustrer comment nos structures réagissent face à la dynamicité du graphe, nous présentons les temps d'insertion/suppression moyens dans le tableau 4.6.

Structure	Temps d'insertion	Temps de suppression
PFT-index	2274	566
PTF-index	3247	1731
TPF-index	3023	1057
Hybrid-index (seuil = 400)	2498	818

Table 4.6 – Temps moyens d'insertion/suppression (en Nano-Secondes)

L'index de type TPF-index ne reflétant pas la structure du graphe social, une recherche chronophage des listes doit être effectuée pour les termes populaires dans le cas d'une insertion. Dans le cas d'une suppression, le même problème se pose pour les comptes populaires suivis sur plusieurs *topics*.

Aussi, en concordance avec notre étude analytique, le PFT-index apparait comme étant la structure la plus dynamique et requiert 30% de temps en moins pour l'insertion et plus de 67% de temps en moins pour la suppression par rapport au PTF-index. Ceci s'explique par le fait que la factorisation exploitée par le PFT (d'abord sur le *publisher* puis sur les *followers*) correspond à une factorisation sur les arêtes du graphe social. Dès lors, l'ajout ou la suppression d'une arête peut se faire efficacement. La factorisation par *publisher* puis par terme dans le PTF-index ne respecte pas le stockage naturel du graphe sous forme de liste d'adjacence et conduit donc à des temps de mise à jour plus importants.

4.6 Une structure hybride

Comme la plupart des plateformes sociales, les systèmes de microblogging sont caractérisés par une forte hétérogénéité. La figure 4.8 ainsi que la figure 4.9 illustrent cette hétérogénéité sur notre jeu de données. Nous pouvons distinguer cinq classes d'utilisateurs en se basant sur leur nombre de *followers*. Nous observons sur la figure 4.8 que plus de la moitié des utilisateurs sont suivis par moins de 10 comptes. En parallèle, nous retrouvons sur notre système une minorité de comptes très populaires qui sont suivis par un grand nombre d'utilisateurs (0,78% des utilisateurs sont suivis par plus de 1000 comptes).

En s'intéressant aux fréquences de publication, nous pouvons observer en figure 4.9 que les classes d'utilisateurs avec le plus grand nombre de *followers* sont aussi les classes avec le plus de publications. Ce phénomène est souvent rapporté dans les réseaux sociaux [Kwak et al., 2010, Ahn et al., 2007].

Se basant sur cet aspect, nous avons exploré la piste d'une structure hybride qui combine les index du type PFT et PTF. Cette nouvelle structure va stocker les comptes qui ont un nombre de *followers* inférieur à un certain seuil de manière similaire au PFT-index et les comptes ayant un grand nombre de *followers* (supérieur au seuil défini) en utilisant une factorisation PTF. La motivation principale pour cette approche vient de l'observation que la structure PTF gère mieux les comptes populaires que le PFT-index (voir section 4.4.3).

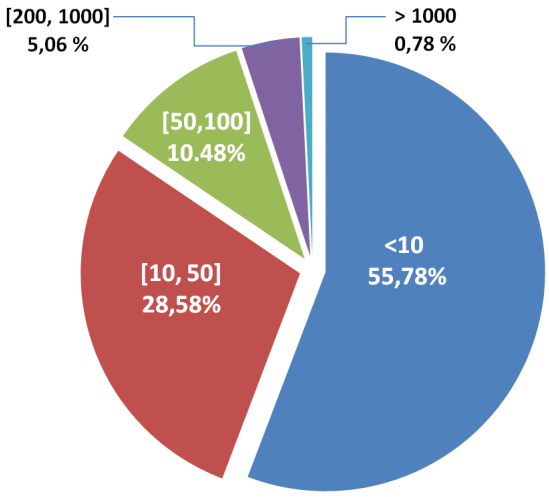

Figure 4.8 – Utilisateurs suivant leur nombre de *followers*

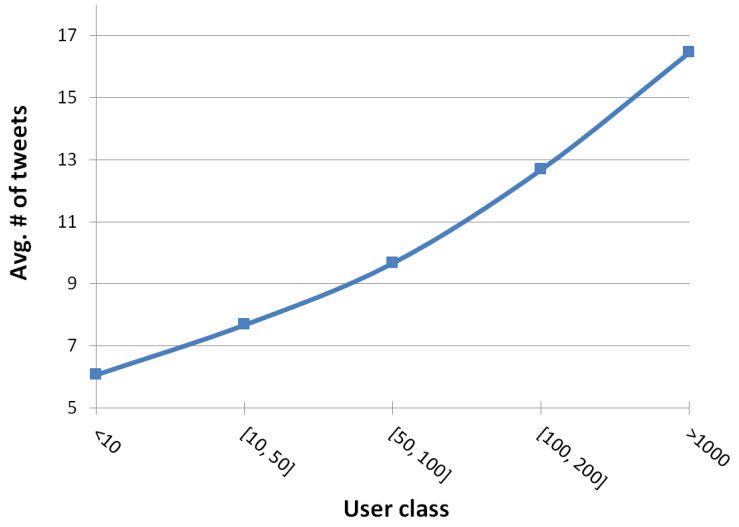

Figure 4.9 – Utilisateurs suivant leur nombre moyens de *tweets*

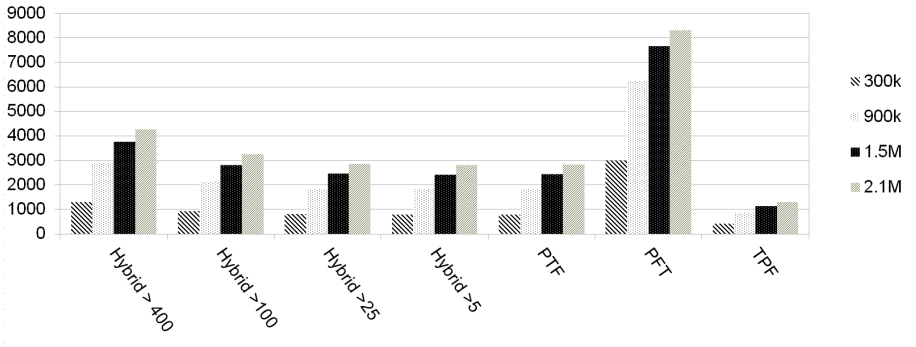

Figure 4.10 – Espace mémoire occupé par rapport à N et au seuil choisi

4.6.1 Occupation mémoire de la structure hybride

Le paramètre clé dans le contexte hybride est la valeur choisie pour le seuil. Le seuil représente le nombre minimal de *followers* qu'un compte doit avoir pour être considéré comme compte populaire. Nous appellerons ce seuil dans ce qui suit le *seuil de popularité*. En fonction de cette valeur, nous décidons de stocker un compte sous forme PFT-index (non-populaire) ou bien sous forme de PTF-index (populaire). Les résultats concernant l'évolution de l'espace mémoire occupé par la structure hybride en fonction du seuil de popularité sont illustrés en figure 4.10.

Nous observons que la gestion des grand comptes (en nombre de *followers*) sous format PTF-index a un impact direct sur la taille occupée par les structures d'indexation en mémoire centrale. Plus la valeur de seuil de popularité choisie est petite, plus la taille de la structure d'index est petite. En conséquence, nous observons que la taille mémoire de l'index hybride est bornée par les tailles des structures PFT et PTF. Par exemple, pour un seuil de popularité de 5, la taille mémoire occupée par la structure hybride est similaire a celle occupée par l'index PTF. En contraste, un seuil de popularité de de 400 induit une croissance de la taille mémoire de 30% par rapport au PTF-index.

4.6.2 Temps de matching pour la structure hybride

Nous reportons les temps de matching obtenus pour la structure hybride en figure 4.11. Nous observons que le temps de matching demeure constant

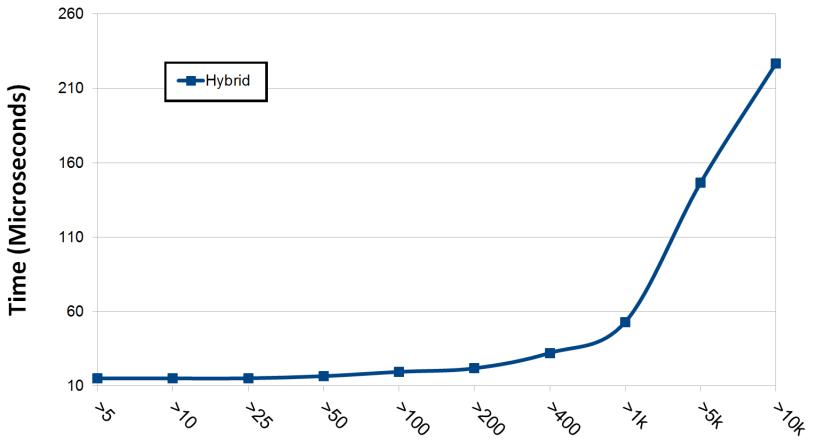

Figure 4.11 – Temps de matching pour la structure hybride par rapport au seuil popularité

pour les valeurs de seuils de popularité entre situées 5 et 25. Avec ce paramètre de partitionnement de l'espace des utilisateurs, la structure hybride a un temps de matching moyen de 15 μs, donc similaire à la valeur observée pour le PTF-index (voir section 4.4.3). Le tableau 4.7 affiche le nombre d'utilisateurs (sur les 2,17 millions d'utilisateurs dans notre graphe social) indexés sous forme de PTF et ce pour différentes valeurs de seuil de popularité. Nous observons qu'à partir d'un seuil de popularité de 25, ce qui mène à considérer près de 25% des utilisateurs en tant que comptes populaires, le temps de matching croit rapidement. Cependant, nous observons qu'avec un seuil de popularité de 400 seulement 2,5% sont considérés populaires et indexés en tant que PTF mais le gain en temps de matching par rapport à une structure de PFT est de l'ordre de 25,2 ($32\mu s$ contre $808\mu s$). Ces gains observés confirment notre intuition qui consiste à distinguer les (rares) comptes populaires pour l'indexation.

Par rapport à la gestion de la dynamicité du graphe, il apparait que la structure hybride gère les mises à jour plus efficacement que l'index PTF et presque aussi vite que le PFT (Tableau 4.6). Une insertion est réalisée en 2498 ns et une suppression en 818 ns (en moyenne, avec un seuil de popularité de 400) par rapport à 2274 ns et 566 ns pour le PFT-index. Cela nous pousse à conclure donc que l'utilisation d'une factorisation spécifique pour les grands comptes (en terme de *followers*) permet d'accroitre l'efficacité

Seuil de popularité	Nombre d'utilisateurs
5	1,349,490
10	959,976
25	547,001
50	337,393
100	199,033
200	209,844
400	54,024
1,000	16,961
5,000	2,118
10,000	896

Table 4.7 – Nombre d'utilisateurs pour différentes valeurs de seuil de popularité

générale du système de filtrage.
La structure hybride réalise un compromis intéressant entre le PTF-index et le PFT-index. Par exemple avec la structure hybride et un seuil de popularité de 400, l'index occupe 30% d'espace en plus qu'un PTF-index mais réalise un gain de 50% par rapport à un PFT-index. En parallèle, le temps de matching moyen obtenu est le double de celui obtenu par le PTF mais demeure 25 fois plus petit que le temps nécessaire pour le PFT-index. Aussi, la structure hybride affiche de meilleures performances que le PFT-index en ce qui concerne la gestion de la dynamicité du graphe avec un gain de 23% pour le temps nécessaire à l'insertion et 53% pour la suppression.

Le tableau 4.8 offre un récapitulatif des points forts et points faibles des différentes structures.

	PFT-index	PTF-index	TPF-index	Hybride
Occupation mémoire	- -	+	+ +	+
Temps de recherche	+	+ +	- -	+ +
Temps m.à.j	+ +	- -	-	+

Table 4.8 – Efficacité des structures d'indexation

4.7 Conclusion

Dans ce chapitre, nous avons comparé différentes structures de listes inverses qui permettent d'indexer des filtres dans le but d'améliorer l'expérience utilisateur tout en réduisant le nombre de messages transmis par le systèmes de micro-blogging. Nous avons proposé un modèle analytique de coût en temps et en mémoire pour ces structures. Ce modèle a été validé expérimentalement sur un un jeu de données conséquent. La structure PTF-index est apparue comme le candidat qui a atteint la meilleure performance sur les critères de passage à l'échelle. Malgré une occupation mémoire deux fois plus importante que pour le TPF-index, cette structure surpasse les autres propositions de plusieurs ordres de grandeur en ce qui concerne le temps de matching.

Nous avons aussi démontré que le fait d'exploiter l'hétérogénéité présente sur les plateformes sociales comme reporté par [Kwak et al., 2010] (ex. 5% des comptes avec plus de 100.000 *followers*) permet d'améliorer la performance globale du système de filtrage. Notre proposition d'un filtrage hybride fournit un compromis intéressant tout en améliorant la capacité a suivre l'évolution et la dynamicité du graphe social.

Comme pistes d'améliorations possibles, nous envisageons de considérer Le clustering de filtres. Cette approche permettrait de regrouper les différents filtres à l'intérieur d'une même *posting list* afin d'atteindre de meilleures performances. Gérer les cas de conjonction et de négation pour les filtres est aussi un autre défi. Nous envisageons aussi l'exploitation d'ontologies pour aboutir un filtrage plus "intelligent" (gérer les relations de synonymie et les inclusions).

Finalement, étant donné les filtres qu'un utilisateur peut poser sur le système, nous sommes capables de dresser un profil précis de ses intérêts. Le chapitre suivant décrit une approche de recommandation basée sur les données implicites au graphe social étiqueté *LSG*.

Chapitre 5

Recommandation de comptes

5.1 Introduction

Après avoir introduit un mécanisme de filtrage efficace sur les graphes sociaux de type micro-blogging. Nous avons choisi d'aborder le problème de la découverte d'information pertinente à travers la recommandation de comptes à suivre. En effet, même si le filtrage permet le nettoyage du contenu reçu par les utilisateurs des plateformes sociales, il offre cependant une vue réduite et étroite de ce qui se passe plus globalement sur le réseau. La recommandation de comptes aux utilisateurs permettra de palier ce phénomène.

Dans ce chapitre, nous présentons d'abord notre motivation et nos objectifs. Nous décrivons ensuite le score de recommandation proposé qui exploite efficacement toutes les données intrinsèques au graphe social étiqueté que nous nommons *LSG*. Après avoir défini notre score ainsi que ses composantes, nous décrivons une approche qui permet l'approximation efficace de nos scores sur le graphe social en exploitant une approche par *landmark*. Nous présentons ensuite les différentes expériences conduites afin de valider la qualité mais aussi l'efficacité de notre approche de recommandation de comptes. Enfin, une conclusion résume les contributions de notre approche ainsi que les pistes d'amélioration envisagées.

5.2 Motivation

Toujours dans l'optique d'améliorer l'expérience des utilisateurs, nous proposons une stratégie afin de générer des recommandations personnalisées de comptes à suivre sur une plateforme de type micro-blogging. Le score qui est au cœur de notre stratégie de recommandation prend en compte trois aspects déduits directement de la structure de *LSG* : la proximité topologique, la sémantique qui existe sur les liens entre les nœuds du graphe ainsi que l'autorité des comptes recommandés sur des *topics* précis.

Comme décrit dans les chapitres 1 et 2, La taille du graphe social sous-jacent dans les réseau sociaux soulève une question de faisabilité quant au déploiement des stratégies de recommandations car ces stratégies impliquent la plupart du temps des explorations locales du graphe social. Afin d'accélérer le processus de recommandation, nous proposons une approximation efficace des scores basée sur une approche par *landmarks* c.à.d la sélection d'un ensemble de nœuds dans le graphe social, appelées "*landmarks*", qui joueront le rôle de "stations locales" (*hubs*) et qui emmagasineront un ensemble de données sur leur voisinage étendu. Cet ensemble de *landmarks* est déterminé en appliquant différentes stratégies de sélection que nous décrivons en section 5.5.

A noter que nous illustrons notre stratégie de recommandation dans un contexte micro-blogging avec des expérimentations sur des données issues de *Twitter*. Cependant, notre modèle de recommandation est plus général et peut être aisément adapté à n'importe quelle plateforme sociale où les utilisateurs publient du contenu textuel et reçoivent les publications des comptes qu'il suivent sur le réseau.

5.3 Recommandation

Dans cette section, nous définissons les différentes composantes du score de recommandation proposé. Notre système de recommandation exploite le modèle de graphe *LSG* présenté et décrit en chapitre 3. Le tableau 5.1 regroupe les différentes notations utilisées dans les définitions de ce chapitre.

Afin d'illustrer le fonctionnement de notre système de recommandation, nous illustrons en Figure 5.1 un exemple de *LSG* avec le contenu des publications pour les comptes B et C.

N, E	resp. l'ensemble de nœuds et d'arêtes
Γ^u, $\Gamma^u[t]$	*followers* de u (resp. total des followers sur un *topic* t)
$\mathcal{T}$	le vocabulaire de *topics*
$\mathcal{P}$	l'ensemble de toutes les publications
$label_n$, $label_e$	la fonction d'étiquetage pour les nœuds du graphe, resp. pour les arêtes
$katz_\beta(u, v)$	Le score de *Katz* d'un nœud v pour un nœud u avec un facteur de décroissance β
$\sigma(u, v, t)$	le score de recommandation d'un nœud v pour un nœud u sur un *topic* t
$\omega_p(t)$	le score d'un chemin p dans le graphe pour un *topic* t
$\overline{\omega}_p(t)$	le score d'un chemin p
$auth(u, t)$	le score d'autorité pour un nœud u sur un *topic* t
$\varepsilon_e(t)$	la pertinence d'une arête e pour un *topic* t
λ, $\mathcal{L}$	un *landmark*, resp. l'ensemble des *landmarks*
$P_{u,v}$	l'ensemble des chemins existants entre u et v
$P_{u,\lambda,v}$	l'ensemble des chemins existants entre u and v passant par le nœud λ
$\Upsilon_k(\lambda)$	le *k-voisinage* de λ
$R_{u,v}$	le vecteur de recommandation de v pour u pour tous les *topics*

Table 5.1 – Notations

Notre objectif est de recommander à un utilisateur u du système des comptes pertinents à suivre pour un *topic* t. Divers scores et stratégies de recommandation ont étés proposés dans le contexte des réseaux sociaux (voir Section 2.4.5) pour recommander des comptes, des publications ou bien des *topics*. Ces propositions considèrent le plus souvent soit la topologie du réseau social sous-jacent soit le contenu des publications des utilisateurs. Notre proposition, RecLand, se base sur les hypothèses suivantes :

i) *La proximité topologique* est une donnée cruciale pour le calcul de la recommandation, *c.à.d* un compte u fait confiance à ses voisins directs dans le graphe, aux voisins de ses voisins, etc. Cette confiance s'amoindrit avec l'accroissement de la distance [Liben-Nowell and Kleinberg, 2003].

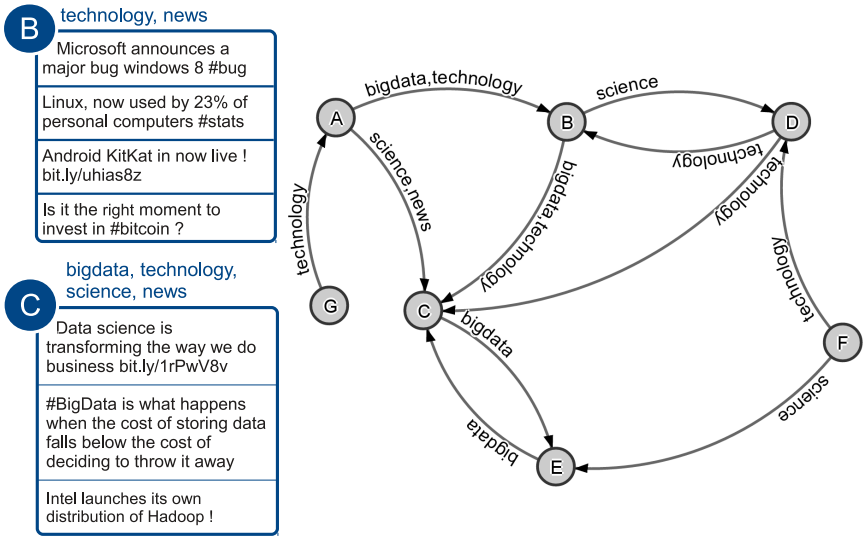

Figure 5.1 – Exemple de *LSG* dans un contexte de recommandation

ii) *Le nombre de chemins* entre u et v est lui aussi important : un compte v a de grandes chances d'être considéré comme important par un compte u s'il existe un grand nombre d'autres nœuds liés à u qui recommandent à leur tour v.

iii) *La pertinence par rapport à un topic* sur les chemins existants entre u et v doit être aussi considérée dans le calcul des recommandations.

Il a été démontré que le score de *Katz* [Katz, 1953a] (défini dans l'état de l'art, section 2.4.5) surpasse d'autres mesures topologiques sur les réseaux issus du Web dans un contexte de prédiction de liens [Liben-Nowell and Kleinberg, 2003]. Ce score considère le nombre de chemins existants entre deux nœuds ainsi que leur longueurs respectives. Nous proposons dans notre système une extension de ce score qui intègre l'aspect sémantique c.à.d la pertinence des chemins entre un nœud et sa recommandation par rapport à un *topic* donné.

5.3.1 Le score de recommandation

Pour rappel, la mesure de *Katz* entre un nœud u et un nœud v s'exprime de la façon suivante :

$$katz_\beta(u,v) = \sum_{l=1}^{\infty} \beta^l \times |P_{u,v}^{\langle l \rangle}| = \sum_{p \in P_{u,v}} \beta^{|p|}$$

où $P_{u,v}$ est l'ensemble de tous les chemins existants entre u et v, $P_{u,v}^{\langle l \rangle} \subseteq P_{u,v}$ l'ensemble de tous les chemins de longueur égale à l existants entre u et v et $\beta \in [0,1]$ le facteur de décroissance qui permet d'accorder plus d'importance aux chemins courts. Une faible valeur de β implique des prédictions similaires aux mesures basées sur le nombre de voisins communs car les long chemins n'auront alors qu'une faible contribution au score total. En se basant sur ce score topologique, nous proposons notre score de recommandation.

Définition 1 (Score de recommandation) *Le score de recommandation $\sigma(u,v,t)$ d'un utilisateur v pour un utilisateur u sur un topic t et les chemins $p = u \rightsquigarrow v$ est défini comme suit :*

$$\sigma(u,v,t) = \sum_{p \in P_{u,v}} \overline{\omega}_p(t) = \sum_{p \in P_{u,v}} \beta^{|p|} \omega_p(t) \tag{5.1}$$

où le score du chemin $\overline{\omega}_p(t)$ est composé du score de Katz $\beta^{|p|}$, et de $\omega_p(t)$: le score sémantique du chemin entre u et v de longueur $|p|$ pour un topic t.

Le score sémantique du chemin $\omega_p(t)$ prend en considération la pertinence des nœuds (autorité) ainsi que celle des arêtes (sur les *topics*) sur ce chemin pour chaque *topic*. Nous explicitons ces concepts dans ce qui suit.

L'autorité d'un nœud

Nous définissions une fonction d'autorité $auth(u,t)$ pour chaque nœud u du graphe social sur un *topic* t. Cette fonction dépend du nombre d'utilisateurs qui suivent u sur le *topic* t. Le score d'autorité est composé de deux parties :

(i) un *score d'autorité locale* qui assigne une valeur plus élevée aux utilisateurs u qui sont "spécialisés" sur un *topic* t par rapport aux utilisateurs qui publient sur une vaste variété de *topics*.

(ii) une *popularité globale* qui permet d'attribuer un meilleur score aux utilisateurs les plus suivis sur t et ce dans la globalité du graphe social.

La combinaison d'une composante locale et d'une composante globale est une approche classique traditionnellement adoptée pour l'évaluation de l'importance d'un document (ex. le score de TF/IDF) mais aussi l'importance d'un nœud en terme d'autorité dans un graphe de pages Web [Jeh and Widom, 2003] ou plus récemment dans le monde du micro-blogging [Gupta et al., 2013]. Nous définissons formellement le score $auth(u,t)$ de la manière suivante :

Définition 2 (Le score d'autorité d'un nœud) *Le score d'autorité d'un nœud $auth(u,t)$ s'exprime comme suit :*

$$auth(u,t) = \underbrace{\frac{|\Gamma^u[t]|}{|\Gamma^u|}}_{local} \times \underbrace{\frac{\log(1+\Gamma^u[t])}{\log(1+max_u(\Gamma^u[t]))}}_{global}$$

où $|\Gamma^u[t]|$ est le nombre de followers de u sur le topic t, et $|\Gamma^u|$ est le nombre total de followers de u.

La composante d'autorité locale est égale à 1 lorsque u est suivi exclusivement sur le *topic* t tandis que l'autorité globale peut être égale à 1 lorsque u est le compte le plus suivi sur le *topic* t dans tout le réseau. Si aucun utilisateur ne suit u sur t les deux composantes sont égales à zéro. Nous avons décidé d'introduire une fonction logarithmique afin de réduire l'écart entre les comptes populaires avec un nombre important de *followers* et les comptes qui ont très peu de *followers*.

A noter que, mis à part pour $max_u(\Gamma^u[t])$, toutes les composantes du score peuvent êtres calculées "localement", en exploitant seulement les informations du nœud en question dans le *LSG*. Le calcul en temps réel des recommandations implique l'estimation de la valeur de $max_u(\Gamma^u[t])$ ce qui peut introduire un sur-coût car cette estimation peut entrainer la consultation de tout le graphe social. Cependant, cette valeur peut être stockée et recalculée périodiquement. En effet, malgré la dynamicité du

graphe social (qui touche particulièrement les comptes populaires), l'application de la fonction logarithmique limite l'impact de telles variations $(\log(max_u(\Gamma^u[t])) \sim \log(max_u(\Gamma^u[t]) + \Delta)$. Par conséquent, nous pouvons évaluer la fonction d'autorité pendant l'exécution.

Le fait de combiner une composante locale à une composante globale dans le calcul de l'autorité d'un utilisateur permet d'assigner une valeur élevée de score d'autorité sur un *topic* t à un compte qui a un grand nombre de *followers* principalement sur ce *topic* t. Cela permet aussi d'assigner la même importance à un compte généraliste et très suivi et à un compte très spécialisé mais avec un petit nombre de *followers*.

Exemple 1 (Autorité globale et locale) *Dans le graphe exemple illustré en figure 5.1 et qui contient un aperçu des tweets publiés par les utilisateurs B et C ainsi que les topics sur lesquels ils publient. L'utilisateur B est plus pertinent sur le topic* `technology` *que l'utilisateur C. En effet, B et C ont la même autorité globale avec deux followers sur ce topic pour les deux comptes. Cependant, le score d'autorité locale de B sur* `technology` *est plus élevé que celui de C du fait que sur les 3 arcs entrant de B, 2 sont étiqueté avec* `technology` *tandis que pour C ce ratio n'est que de 2 sur 6.*
Pour le topic `bigdata`*, l'autorité locale de B et de C est similaire (1 sur 3 pour B et 2 sur 6 pour C) mais il apparait que C est plus suivi sur* `bigdata` *(avec 2 comptes contre 1 pour B). Dès lors, le score d'autorité totale de C sur le topic* `bigdata` *est plus important.*

La pertinence par arêtes

Un chemin $p = u \rightsquigarrow v$ dans le graphe est considéré pertinent par rapport à un *topic* t lorsque les *topics* présents sur les arêtes constituant p sont sémantiquement proches de t. Cela peut être vu comme une mesure de la "conductivité" du chemin p sur le topic t. Aussi, basé sur notre hypothèse de proximité, nous considérons qu'une arête distante sur le chemin contribue moins au score de recommandation qu'un arête proche de u. Cet aspect est contrôlé par un facteur de décroissance $\alpha \in [0, 1]$ qui réduit l'influence d'une arête e plus la distance i à partir de u augmente. Plus précisément, nous définissons la pertinence d'une arête e à la distance i de u sur un chemin p comme suit :

$$\varepsilon_e(t) = \alpha^i \times max_{t' \in topics(e)}(sim(t', t)) \tag{5.2}$$

Où $topics(e)$ retourne les *topics* qui étiquettent une arête e et $sim : F^2 \to \mathbb{R}$ évalue la similarité sémantique entre deux *topics* t et t'. Dans notre implantation nous utilisons la mesure de *Wu and Palmer* [Wu and Palmer, 1994] sur l'ontologie WORDNET [1] mais d'autres mesures peuvent tout aussi bien être employées (ex. Resnik, Disco [2] , etc.). Le choix de la meilleure mesure de similarité dépasse le cadre de notre approche de recommandation. Lorsqu'une arête e est étiquetée avec plusieurs *topics*, nous ne gardons que le *topics* avec le plus grand score de similarité sémantique afin d'éviter d'accroitre artificiellement les scores des arêtes étiquetées par un grand nombre de *topics*.

Le score d'un chemin pour un *topic*

Finalement, nous considérons que l'importance d'un chemin p est élevée lorsque la pertinence des nœuds ainsi que celle des arêtes sur ce chemin sont élevés :

$$\omega_p(t) = \sum_{e \in p} \varepsilon_e(t) \times auth(end(e), t) \tag{5.3}$$

où $end(e)$ retourne le nœud destination d'une arête dirigée e.

Le score de recommandation de v pour l'utilisateur u sur un *topic* t est alors obtenu en substituant $\omega_p(t)$ dans l'équation (5.1) par sa formule donnée par l'équation (5.3).

Exemple 2 (La pertinence d'un chemin) *Dans l'exemple illustré en Figure 5.1, nous voulons recommander à l'utilisateur A des comptes à suivre sur le topic* `technology` *en effectuant une exploration (pour cet example nous supposons une exploration dans un périmètre $k = 2$ sauts). Les utilisateurs D et E peuvent êtres atteints respectivement par les chemins $p_1 = A \to B \to D$ et $p_2 = A \to C \to E$, chacun de longueur 2.*
La pertinence de l'arête $A \xrightarrow{bigdata,technology} B$ est plus importante que celle de $C \xrightarrow{technology} E$ du fait que la première est à une distance 1 de A tandis

1. http://wordnet.princeton.edu/
2. http://www.linguatools.de/disco/disco_en.html

que la deuxième est à distance 2*. De plus, l'autorité sur le topic* `technology` *du nœud B calculé comme* $(local) \times (global) = \frac{2}{3} \times \frac{log(1+2)}{log(1+2)}$ *est plus importante que l'autorité de C sur ce même topic qui est égale à* $\frac{2}{6} \times \frac{log(1+2)}{log(1+2)}$*. Plus globalement, la pertinence sémantique des arêtes sur le chemin* p_1 *par rapport à* `technology` *est plus élevée que celle des arêtes sur* p_2 *ce qui mène à ce que que le nœud D obtienne un meilleur score de recommandation que le nœud E.*

Il est important de souligner que la définition du score de recommandation implique aussi que nous pouvons déduire le score d'un chemin $\overline{\omega}_p(t)$ pour un chemin $p = u \rightsquigarrow v$ de longueur $|p|$ à partir du score $\overline{\omega}'_p(t)$ de son préfixe p' de longueur $|p-1|$ et du score de pertinence de l'arête finale $\varepsilon_e(t)$:

$$\overline{\omega}_p(t) = \beta.\overline{\omega}_{p'}(t) + \beta^n.\alpha^n.\varepsilon_e(t).auth(end(e), t)$$

Par induction, nous déduisons la propriété de composition suivante que nous allons exploiter dans l'estimation des scores de recommandation (voir section 5.4.3).

Proposition 1 (Composition de scores) *En supposant un chemin* $p = p_1.p_2$*, avec* $\overline{\omega}_{p_1}(t)$ *et* $\overline{\omega}_{p_2}(t)$ *les scores de chemins de* p_1 *et* p_2 *pour un topic* t*. Le score du chemin* p *est :*

$$\overline{\omega}_p(t) = \beta^{|p_2|}.\overline{\omega}_{p_1} + \beta^{|p_1|}.\alpha^{|p_1|}.\overline{\omega}_{p_2}$$

Notre score de recommandation peut être considéré comme une extension du score de *Katz* afin de considérer les intérêts des *followers* (exprimés sous forme d'arêtes étiquetés dans le *LSG*) et le score d'autorité par rapport au *topic* d'intérêt pour chacun des utilisateurs présents sur le chemin.

5.3.2 La convergence du calcul des scores

La présence de cycles dans le graphe peut nous mener à considérer un nombre infini de chemins. En conséquence, les approches basées sur le score de *Katz* supposent soit la convergence des scores de chemin, soit l'existence d'un seuil pour les scores des chemin en dessous duquel les chemins sont

écartés du calcul (*pruning*). En utilisant une représentation matricielle, les scores de *Katz* entre toutes les paires de nœuds du graphe peuvent s'exprimer comme suit :

$$K_\beta = \sum_{i=1}^{\infty} \beta^i A^i = (I - \beta A)^{-1} - I$$

Où A représente la matrice d'adjacence pour le graphe social dirigé et $A^k[i,j]$ le nombre de chemins de longueur k entre i et j. Dans ce qui suit, nous faisons référence à K_β pour la matrice de *Katz* avec le facteur de décroissance β.

De cette notation matricielle, le vecteur des scores de *Katz* $K_\beta(n)$ pour un nœud n peut être calculé grâce à la formule récursive suivante :

$$K_\beta^{(k+1)}(n) = A.K_\beta^{(k)}(n) + B$$

où B est un vecteur pour qui $B[n] = 1$ et $\{B[x] = 0, \forall x \neq n\}$ (voir [Bonchi et al., 2012]).

En se basant sur la Proposition 1, notre calcul de score pour un chemin $p.e$ où e est une arête et p un chemin de longueur $|p|$ s'exprime alors de la façon suivante :

$$\overline{\omega}_{p.e}(t) = \beta.\overline{\omega}_p + \beta^{|p|}.\alpha^{|p|}.\overline{\omega}_e = \beta.\overline{\omega}_p + \beta^{|p|}.\alpha^{|p|}.\varepsilon_e(t).auth(end(e), t)$$

Dès lors, notre score de recommandation d'un nœud n vers tout autre nœud du graphe social pour un *topic* t peut s'exprimer de manière récursive comme suit :

$$R^{(k+1)}(t) = (\beta.A).R^{(k)}(t) + (\alpha\beta.A).S(t).K_{\alpha\beta}^{(k)}(n)$$

où S est une matrice contenant le produit des scores de similarité et d'autorité, définie par $S[i,j] = sim(max_{t' \in topics(i,j)}(t', t)) \times auth(j, t)$.

Il a été démontré que le calcul du score de *Katz* K_β converge lorsque $\beta < 1/\sigma_{max}(A)$ où $\sigma_{max}(A)$ est la valeur propre la plus élevée dans A (voir [Bonchi et al., 2012]). Se basant sur ce résultat, nous déduisons la condition de convergence suivante :

Proposition 2 (Convergence du calcul de score) *Si $\beta < 1/\sigma_{max}(A)$, alors le calcul itératif de notre score de recommandation converge.*

Preuve 1 *$K_{\alpha\beta}^{(k)}$ converge si $\alpha.\beta < 1/\sigma_{max}(A)$ (un calcul de score de Katz avec un facteur de décroissance $\alpha.\beta$). En considérant l'étape k' comme l'étape à laquelle la convergence est atteinte. Alors, pour chaque $k > k'$, nous avons le calcul récursif $R^{(k+1)}(t) = (\beta.A).R^{(k)}(t)+C$ avec $C = (\alpha\beta.A).S(t).K_{\alpha\beta}^{(\infty)}(n)$ constant. La convergence pour $R^{(k+1)}(t)$ est atteinte quand $R^{(\infty)}(t) = (\beta.A).R^{(\infty)}(t)+C$ dès lors quand $R^{(\infty)}(t) = (I - \beta.A)^{-1} \times C$. Ceci peut être atteint si $\beta < 1/\sigma_{max}(A)$. Sachant que $\beta > \alpha\beta$, cette condition est alors suffisante pour assurer la convergence.*

Le score de recommandation basé sur *Katz* présenté suppose l'exploration de *tous* les chemins à partir du nœud pour lequel les recommandations sont générées. Afin de générer des recommandations à la volée pendant l'exécution, nous proposons un algorithme basé sur une approche par *landmarks* en deux étapes :

— (*i*) une étape de pré-traitement qui calcule les recommandations pour un ensemble de nœuds, pour chaque *topic* du vocabulaire des *topics*.

— (*ii*) une approximation en temps-réel, au moment de la requête, du calcul des recommandations pour un nœud sur un *topic* donné en exploitant les données pré-calculées.

5.4 Une estimation efficace des recommandations

5.4.1 Vue d'ensemble de l'approche basée sur les *landmarks*

Dans notre contexte, la taille du graphe social rend difficile le calcul efficace de recommandations basées sur des propriétés topologiques. En supposant comme candidats à la recommandations tous les nœuds atteignables en moins de K sauts à partir du compte à recommander. La taille de l'ensemble des candidats est (dans le pire de cas) égale à N^K (pour un graphe complet) et out_{avg}^K dans le cas moyen où out_{avg} représente le nombre moyen de nœuds suivi par un compte sur le réseau. Afin de contourner cette limitation, notre idée est d'adopter une approche par *landmarks* afin d'évaluer efficacement les recommandations.

Nous pré-calculons donc, pour un échantillon de nœuds choisis dans le graphe (*landmarks*), un *top-k* de recommandations (k étant un paramètre

pré-établi par le système) sur chacun des *topics* $t \in \mathcal{T}$. Puis, pendant l'exécution, afin de déterminer les recommandations pour un compte n, nous explorons le graphe autour de n jusqu'à une profondeur $k < K$, où K est la profondeur d'exploration pour un calcul à la convergence. Après avoir récolté toutes les recommandations des différents *landmarks* rencontrés pendant l'exploration, celles-ci sont pondérées en se basant sur notre score de recommandation entre n et les *landmarks*. Une fusion des différents résultats obtenus permet d'obtenir les recommandations finales pour n.

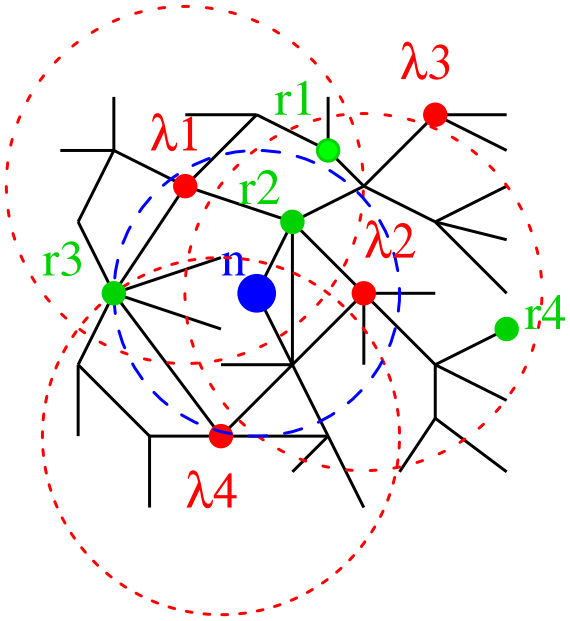

Figure 5.2 – Exemple de recommandation par *landmarks*

La Figure 5.2 illustre notre approche, n est le nœud pour lequel nous calculons les recommandations, λ_1, λ_2, λ_3 et λ_4 sont des *landmarks*. En explorant le graphe en effectuant un *BFS* (*Breadth-First-Search*[3], représenté par la ligne blue en pointillés) à partir de n, seulement λ_1, λ_2 et λ_4 ont été rencontrés. Si l'on observe les recommandation générées, nous remarquons que r_1 n'as pas été rencontré durant l'exploration. Son score est le résultat de la combinaison des scores obtenus à travers les chemins $n - \lambda_1 - r_1$ et $n - \lambda_2 - r_1$. Le score de r_2 est quant à lui le résultat de la combinaison des scores $n - \lambda_1 - r_1$ et $n - \lambda_2 - r_1$ avec le score du chemin à distance

3. http://fr.wikipedia.org/wiki/Algorithme_de_parcours_en_largeur

2 pour $n - \lambda_2$ car il a été croisé durant le *BFS*. On remarque aussi que la recommandation r_4 a été générée uniquement grâce au *landmark* λ_2.

A noter que cette approche estime une borne inférieure des différents scores de recommandation proposés. Généralement, les approches basées sur les *landmarks* dans un contexte de calcul de plus court chemin (voir section 2.5) sont utilisées pour l'estimation d'une borne supérieure en exploitant l'inégalité triangulaire. La raison est la suivante : lorsqu'un *landmark* λ recommande un compte n_2 à compte n_1, le calcul considère l'ensemble P_1 de tous les chemins entre n_1 et λ de longueur inférieure à k_1 ainsi que l'ensemble P_2 de tous les chemins existants entre λ et n_2 de longueur inférieure à k_2. Cependant l'ensemble $P_{1,2} = \{p_i.p_j | p_i \in P_1 \wedge p_j \in P_2\}$ est inclus dans l'ensemble de tous les chemins existants entre n_1 et n_2 de longueur inférieure à $k_1 + k_2$.

5.4.2 Pré-traitement

Dans l'étape de pré-traitement, nous supposons un sous-ensemble $\mathcal{L} \subset N$ de nœuds, appelés *landmarks* tel que $|\mathcal{L}| \ll |N|$. En plus d'un échantillonnage aléatoire, différentes stratégies de sélection peuvent êtres établies afin de déterminer l'ensemble $\mathcal{L}$. Par exemple, les stratégies basées sur les *landmarks* dans un contexte de calcul de plus court chemin se basent essentiellement sur des propriétés de "Centralité" des nœuds (Centralité d'intermédiarité ou de proximité [4]) afin de choisir les *landmarks*. Les caractéristiques de notre graphe social dirigé ainsi que le double rôle des utilisateurs (*publishers, followers*) nous permettent d'envisager d'autres stratégies de sélection basées sur la topologie. Parmi ces stratégies on retrouve le choix des nœuds ayant le plus fort degré entrant (comptes les plus populaires sur le réseau) ou bien les nœuds qui suivent le plus grand nombre de comptes (les lecteurs les plus actifs). On peut aussi mettre en place des contraintes pour maximiser la couverture comme par exemple obliger les *landmarks* choisis à respecter une distance minimale entre eux.

Le choix de la stratégie de sélection de *landmarks* impacte la qualité ainsi que les performances globales du système de recommandation. Nous avons testé et comparé un ensemble de stratégies expérimentalement en Section 5.5.

4. http://fr.wikipedia.org/wiki/Analyse_des_réseaux_sociaux

Pour chaque *landmark* $\lambda \in \mathcal{L}$, nous effectuons une exploration du type *BFS* sans éviter les cycles. Nous appelons l'ensemble des nœuds atteints à la profondeur k à partir de λ le *k-voisinage* de λ noté $\Upsilon_k(\lambda)$. $\Upsilon_\infty(\lambda)$ désigne l'ensemble de tous les nœuds atteignables à partir de λ. Nous calculons pour chaque nœud $n \in \Upsilon_\infty(\lambda)$ un vecteur de recommandation $R_{\lambda,n}$ tel que $R_{\lambda,n}[t] = \sigma(\lambda, n, t)$ ainsi qu'un score de connectivité $katz_\beta(\lambda, n)$. Le score $\sigma(\lambda, n, t)$ est le score de du *landmark* λ pour le compte n sur le topic t, et $katz_\beta(\lambda, n)$ représente le score de *Katz* présenté dans en Section 5.3.1 avec un facteur de décroissance β, utilisé par la suite afin d'estimer le score de recommandation final.

Notre algorithme effectue une "fusion" des différents $R_{\lambda,n}$ pour tout $n \in \Upsilon_\infty(\lambda)$ afin d'obtenir une *posting list* R_λ pour laquelle chaque entrée $R_\lambda[t]$ qui correspond à un terme t est associé une liste de nœuds recommandés, classés par pertinence en fonction de leur score de recommandation $\sigma(\lambda, n, t)$. Nous gardons dans ces listes uniquement un top-k des recommandations calculées.

L'algorithme 1 effectue le calcul des scores de recommandation (nous avons omis les différentes initialisation à 0 par soucis de simplification). Les paramètres utilisés sont le facteur de décroissance choisi pour la fonction de *Katz* ainsi que la profondeur maximale pour notre exploration *BFS*. Cette valeur est utilisée par l'étape de *BFS* pour le nœud qui pose la requête de recommandation (voir Algorithme 2). Pour les *landmarks*, étant donné le fait que la calcul se fait jusqu'à la convergence, cette valeur n'est pas utile dans ce contexte et peut être initialisée à une valeur importante (∞).

Pour un *landmark* donné, cet algorithme est appelé avec en paramètre l'ensemble de *topics* $\mathcal{T}$. Les itérations en ligne 2 permettent d'explorer le k-voisinage de λ. A chaque itération, nous ajoutons au k-voisinage les nœuds atteignables avec un saut additionnel (l. 3-4). Pour tous les nœuds atteints à cette étape (l. 5), nous calculons le score de recommandation pour chacun des termes du vocabulaire des *topics* (si ce nœud à déjà été rencontré, nous mettons à jour cette valeur) (l. 6-7) ainsi que le score de *Katz* pour m (l. 9). Par la suite, nous mettons à jour le score de recommandation pour n (l. 11) et son score de *Katz* (l. 12). Un test de convergence est effectué en lignes 14-20 (la différence entre deux étapes doit varier de moins d'une certaine valeur ν). Lorsque tous les vecteurs finissent par converger, l'algorithme retourne les listes de recommandations pour chaque *topic* ainsi que le score de *Katz* en ligne (l. 23).

Algorithm 1: Landmark_Recomm(k, β)

Require: une longueur de chemin maximale k, un facteur de décroissance β.
Ensure: un ensemble de listes de recommandations R, un vecteur de connectivité $katz_\beta$

$\Upsilon_0 := \lambda, \quad i := 0, converged := false$
while $i < k$ **and** $converged = false$ **do**
  for all $n \in \Upsilon_i$ **do**
    $\Upsilon_{i+1} := \Upsilon_{i+1} \cup \Gamma^+(n)$
    for all $m \in \Gamma^+(n)$ **do**
      for all $t \in \mathcal{T}$ **do**
        $\sigma^{(i+1)}(\lambda, m, t) = \sigma^{(i+1)}(\lambda, n, t) + \beta \times \sigma^{(i)}(\lambda, n, t) + \beta^{i+1}.\alpha^{i+1}.\varepsilon_{n,m}(t))$
      end for
      $katz_\beta^{(i+1)}(\lambda, m) = katz_\beta^{(i+1)}(\lambda, m) + \beta \times katz_\beta^{(i)}(\lambda, n)$
    end for
    $R_{\lambda,n}^{(i+1)}[t] = R_{\lambda,n}[t] + \sigma^{(i)}(\lambda, n, t)$
    $katz_\beta(\lambda, n)^{(i+1)} = katz_\beta(\lambda, n) + katz_\beta^{(i)}(\lambda, n)$
  end for
  if $|katz_\beta(\lambda, n)^{(i+1)} - katz_\beta(\lambda, n)^{(i)}|/katz_\beta(\lambda, n)^{(i+1)} < \nu$ **then**
    $converged := true$
  end if
  for all $t \in \mathcal{T}$ **do**
    if $|R_{\lambda,n}^{(i+1)}[t] - R_{\lambda,n}^{(i)}[t]|/R_{\lambda,n}^{(i)}[t] < \nu$ **then**
      $converged := true$
    end if
  end for
  $i := i + 1$
end while
return for all $t \in \mathcal{T}$ top_recommendations$(R_{\lambda,n}[t]), katz_\beta(\lambda, n))$

5.4.3 Approximation rapide des recommandations

Nous présentons dans cette section l'algorithme réalisant l'estimation des score de recommandation à l'arrivée d'une requête, pendant l'exécution, en se basant sur les calculs qui ont été effectués pour chaque *landmark* à l'étape de pré-traitement. Nous supposons que nous devons générer les recommandations de comptes à suivre pour un compte n sur un *topic* t.

Le traitement commence par une exploration *BFS* à partir du nœud n

similaire à celle effectuée dans l'étape de pré-traitement (voir Section 5.4.2) pour une profondeur d'exploration maximale k. L'objectif de ce BFS est triple : (i) l'exploration du voisinage de n et la découverte des *landmarks* les plus proches, (ii) le calcul des scores de chemins pour le *topic* t pour les chemins existants entre n et chaque *landmark* découvert et (iii) la mise à jour des scores de recommandation stockées par les *landmarks* en tenant compte des chemins entre n et les *landmarks*.

Afin de définir notre score, nous supposons $\Lambda \subseteq \mathcal{L}$ l'ensemble des *landmarks* trouvés par le BFS et R_λ la *posting list* contenant les recommandation pour chaque $\lambda \in \Lambda$. Chaque R_λ contient pour chaque *topic* t une liste des comptes recommandés N_{recomm} associés à leur score de recommandation $R_{\lambda,n}[t] = \sigma(\lambda, n, t)$ et leur score de Katz $katz_\beta(\lambda, n)$ pour chaque $n \in N_{recomm}$. Nous supposons aussi $P_{a,b}$ comme étant l'ensemble des chemins à partir d'un nœud a vers un nœud b et $P_{a,b,c}$ l'ensemble des chemins partants d'un nœud a vers un nœud c qui passent à travers le nœud b. Se basant sur les *landmarks*, nous définissons le score de recommandation comme suit :

Définition 3 (L'approximation du score de recommandation) *Le score de recommandation approché d'un nœud $n \in N$ pour un nœud m sur un topic t via un landmark λ, noté $\widetilde{\sigma}(m, \lambda, n, t)$, correspond au score cumulé de tous les chemins de $P_{m,\lambda,n}$.*

La valeur de ce score de recommandation approché est donné par la proposition suivante :

Proposition 3 (Calcul approché du score de recommandation)

$$\widetilde{\sigma}(m, \lambda, n, t) = \sigma(m, \lambda, t).katz_\beta(\lambda, n) + katz_{\beta.\alpha}(m, \lambda).\sigma(\lambda, n, t)$$

Preuve 2 *Chaque chemin p de $P_{m,\lambda,n}$ peut être décomposé en deux sous-chemins p_1 et p_2 tel que $p_1 \in P_{m,\lambda}$ et $p_2 \in P_{\lambda,n}$. Aussi, il est évident que chaque chemin $p = p_1.p_2$ tel que $p_1 \in P_{m,\lambda}$ et $p_2 \in P_{\lambda,n}$ est un chemin de $P_{m,\lambda,n}$. Par conséquent, en se basant sur la proposition 1, nous avons :*

$$
\begin{aligned}
\widetilde{\sigma}(m,\lambda,n,t) &= \sum_{p\in P_{m,\lambda,n}} \omega_p(t) \\
&= \sum_{p_1\in P_{m,\lambda}} \sum_{p_2\in P_{\lambda,n}} \beta^{|p_2|}.\omega_{p_1}(t) + \beta^{|p_1|}.\alpha^{|p_1|}.\omega_{p_2}(t) \\
&= \sum_{p_1\in P_{m,\lambda}} \omega_{p_1}(t).\sum_{p_2\in P_{\lambda,n}} \beta^{|p_2|} + \sum_{p_1\in P_{m,\lambda}} (\beta.\alpha)^{|p_1|}.\sum_{p_2\in P_{\lambda,n}} \omega_{p_2}(t) \\
&= \sigma(m,\lambda,t).katz_{\beta}(\lambda,n) + katz_{\beta.\alpha}(m,\lambda).\sigma(\lambda,n,t)
\end{aligned}
$$

Nous effectuons le calcul de notre score de recommandation approché pour un nœud n et un *topic* t en appliquant l'algorithme 2. Nous utilisons l'algorithme Landmark_Recomm afin de calculer les score de recommandation et les scores de *Katz* pour n vers tous les nœuds atteignables à distance k (l. 1). En pratique, les recommandations sont calculées pour un seul *topic*. Le facteur de décroissance utilisé dans ce cas est $\beta.\alpha$. Pour tous les *landmarks* rencontrés durant l'exploration, nous combinons leurs scores de recommandation pour t avec le score calculé à partir de n vers les *landmarks* comme décrit dans la Proposition 3 (l. 5).

Algorithm 2: Approx_Recomm(n, k, t, β, α)

Require: un nœud n, une profondeur max. de BFS k, un *topic* t, un facteur de décroissance pour les chemins β et pour les arêtes α.
Ensure: une liste triée de recommandations $\widetilde{R_n}$ pour n

1: $(R_n, katz_{\beta.\alpha}(n)) \leftarrow$Landmark_Recomm$(G, n, k, t, \beta.\alpha)$
2: **for all** $m \in R_n$ **do**
3: **if** $m \in \mathcal{L}$ **then**
4: **for all** $o \in R_m[t]$ **do**
5: $\widetilde{R_n} := Merge(\widetilde{R_n}, (o, \sigma(n,m,t).katz_{\beta}(m,o) + katz_{\beta.\alpha}(n,m).\sigma(m,o,t)))$
6: **end for**
7: **end if**
8: **end for**
9: **return** $\widetilde{R_n}$

Après avoir défini les différentes composantes du score de recommandation proposé, une étape de validation expérimentale est nécessaire afin de tester nos hypothèses.

5.5 Expérimentations

Dans cette section, nous décrivons les expérimentations que nous avons conduites afin de valider notre approche de recommandation sur le jeu de données "Twitter 400k" issu du réseau *Twitter*. Ce jeu de données est décrit de manière détaillée en Section 3.6.

5.5.1 Implantation

Notre système de recommandation a été implanté en JAVA (JVM 1.7). Les différentes expériences ont étés réalisées sur un serveur 64-bit avec un processeur de 10 cœurs du type Intel Xeon à 2.20GHz et 128 Go de mémoire centrale. Le serveur utilise un système d'exploitation Linux avec un Kernel version 3.11.10.

Nous avons implanté nos index en exploitant une structure de listes inverses issue du travail réalisé sur le filtrage (voir Chapitre 4). Étant donné la surcharge induite par la JVM (*Java Virtual Machine*), nous avons fait le choix de nous baser sur des structures de données basiques (Entiers sur 32 bits, tableaux) afin d'améliorer l'aptitude au passage à l'échelle de notre système en terme d'espace occupé et de temps d'exécution. Nous avons appliqué les mêmes choix de conception pour l'implantation des stratégies concurrentes.

Pour plus d'efficacité, nous avons pré-calculé et stocké les scores de similarité sémantique (basés sur la mesure de *Wu and Palmer* calculée sur WORDNET 3.0) dans une matrice de similarité maintenue en mémoire centrale. Cette matrice triangulaire a été stockée sous la forme d'un tableau avec une fonction d'accès spécifique afin d'optimiser le stockage et de permettre la gestion éventuelle de vocabulaires de *topics* volumineux. Même si, dans le jeu de données utilisé, les 18 *topics* standards résultent en une matrice de similarité de 2.5 Ko, un vocabulaire de 10.000 *topics* nécessiterait moins de 750Mo et pourrait donc tenir aisément en mémoire centrale.

Les recommandations calculées par les *landmarks* sont elles aussi indexées sous forme de listes inverses : pour chaque *landmark* est associé l'ensemble de nœuds recommandés ainsi que leur score de recommandation pour chaque *topic* de $\mathcal{T}$. Les *landmarks* sont choisis en appliquant les stratégies de sélection listées dans le tableau 5.2.

5.5.2 Qualité des recommandations

Afin d'estimer la qualité des recommandations générées par notre système, nous adoptons la méthodologie décrite par [Cremonesi et al., 2010] :

i) Nous construisons un ensemble candidat de T comptes en retirant des arêtes du graphe d'après une stratégie de sélection d'arêtes ;
ii) Pour chaque compte i de l'ensemble candidat sont choisis aléatoirement 1000 comptes du graphe social ;
iii) Le système calcule les recommandations des 1001 comptes (1000 comptes choisis + compte de l'ensemble candidat) et génère une liste triée L ;
iv) Lorsque le compte de l'ensemble candidat appartient au top-N de L nous comptabilisons un succès (*hit*), sinon un échec.
v) L'itération sur l'ensemble-candidat se poursuit.

La mesure de précision (resp. rappel) que nous utilisons est décrite par [Cremonesi et al., 2010] par la formule $\#hits/N.T$ (resp. $\#hits/T$). Pour nos expérimentations, la taille de l'ensemble candidat a été initialisée à $T = 100$ et les valeurs considérées sont les moyennes calculées sur 10 essais successifs. Nous comparons la qualité des recommandations générées avec les résultats produits par deux autres approches : la mesure standard de *Katz* [Liben-Nowell and Kleinberg, 2003] qui est strictement topologique, et TwitterRank [Weng et al., 2010] qui considère, en plus de la topologie, une similarité entre les comptes sur des *topics* donnés.

La valeur choisie pour le facteur de décroissance β pour *Katz* et notre proposition RecLand est de $0,0005$. Il a été reporté par [Liben-Nowell and Kleinberg, 2003] que cette valeur permet d'obtenir de bons résultats sur les graphes issus du Web. La valeur des coefficients α et β pour TwitterRank à été initialisée à 0.85 comme indiqué par [Weng et al., 2010]. Les résultats présentés ont été obtenus à la convergence.

La Figure 5.3 illustre la performance des trois systèmes de recommandation pour des arêtes choisies aléatoirement lors de la construction des ensembles-candidats. Nous pouvons observer que TwitterRank est distancé par les deux autres algorithmes. En effet, seulement 4% des recommandations générées avec TwitterRank apparaissent dans le top-1. Ce taux est de 13% pour *Katz* et 30% pour RecLand. Il apparait donc que notre approche apporte un gain de respectivement 7.5 et 2.3 par rapport à *Katz* et Twitter-

Rank sur le top-1. Sur un top-10, ce gain demeure significatif avec des gains respectifs de $4,1$ et de $1,2$ de RecLand par rapport à respectivement *Katz* et TwitterRank.

La Figure 5.4 confirme le fait que RecLand fournit de meilleures recommandations. En effet, pour une même valeur de rappel, nous observons que la précision de notre approche est au minimum le double de celle obtenue par *Katz* et de plus d'un ordre de grandeur plus élevée que celle obtenue par TwitterRank. Par exemple, pour une valeur de rappel de $0,3$, RecLand offre une précision de $0,3$ lorsque *Katz* est à $0,09$ et TwitterRank à $0,01$.

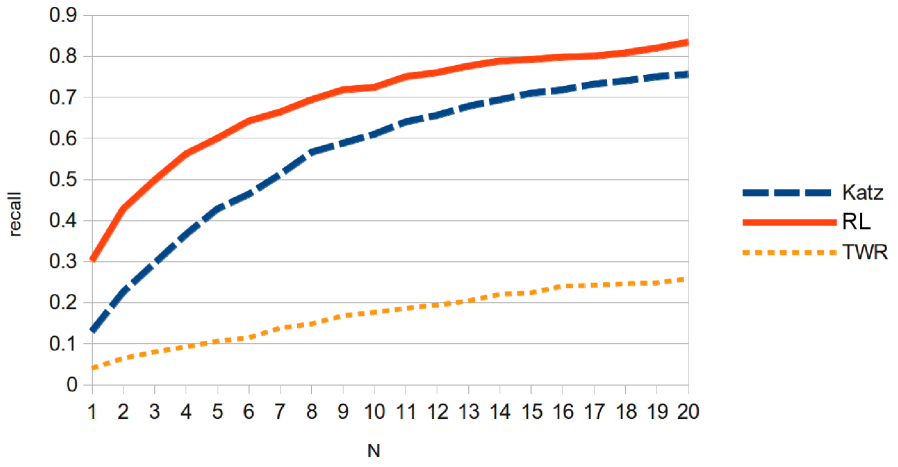

Figure 5.3 – Rappel à N

Cependant, on observe la présence d'une grande disparité lorsque que l'on considère les résultats sur les deux dimensions d'analyse qui sont : la stratégie de sélection des arcs à retirer et la popularité du *topic* sur lequel générer les recommandations.
Sur un top-10, la Figure 5.5 montre une faible valeur de rappel pour notre stratégie lorsqu'on génère des recommandations pour un compte qui a moins de 10 followers. En parallèle, les comptes populaires (ayant une valeur de degré entrant supérieure à 100) sont la plupart du temps présents dans le top-10 des recommandations générées avec une valeur de rappel oscillant entre $0,8$ et $0,9$. Ce phénomène s'explique par l'approche topologique adop-

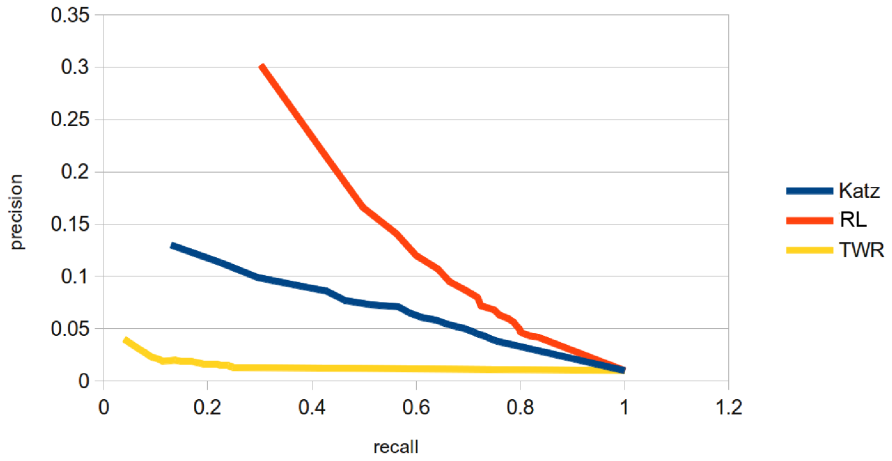

Figure 5.4 – Précision VS rappel

tée qui est basée sur l'existence de chemins, les comptes ayant un grand nombre de chemins entrants voient ainsi leur score augmenter.

Il apparait aussi que TwitterRank obtient le meilleur résultat pour les comptes très populaires. En effet, la plupart des comptes populaires sont étiquetés par plusieurs *topics*. Tandis que TwitterRank se base sur la présence ou l'absence d'un *topic* sans prendre en compte le nombre d'étiquettes sur les arcs entrants, notre score s'appuie sur le nombre d'étiquettes afin de déterminer l'autorité d'un compte. Dans ce cas, la présence de plusieurs arcs entrants avec des *topic* différents entraine une diminution du score d'autorité du compte pour un *topic* donné. Inversement, un compte avec moins de 10 *followers* a rarement des *topics* différents sur ses arcs entrants. Notre approche qui consiste à considérer les *topics* sur les chemins entre les comptes à recommander et leurs recommandations se montre alors particulièrement efficace.

En ce qui concerne le degré sortant des comptes pour lesquels les arcs sont retirés, on observe que TwitterRank obtient un meilleur résultat avec les comptes ayant un faible degré sortant (suivant peu de comptes). La raison est que le fait de retirer un arc sur un compte déjà peu connecté au réseau peut entrainer la déconnexion (complète ou partielle) d'une partie du graphe. Comme Katz et RecLand se basent sur le nombre de chemins

communs, cela peut avoir un impact important sur les scores finaux. De manière opposée, les comptes suivant plus 1000 autres comptes offrent un grand nombre de chemins alternatifs pour atteindre les comptes recommandés.

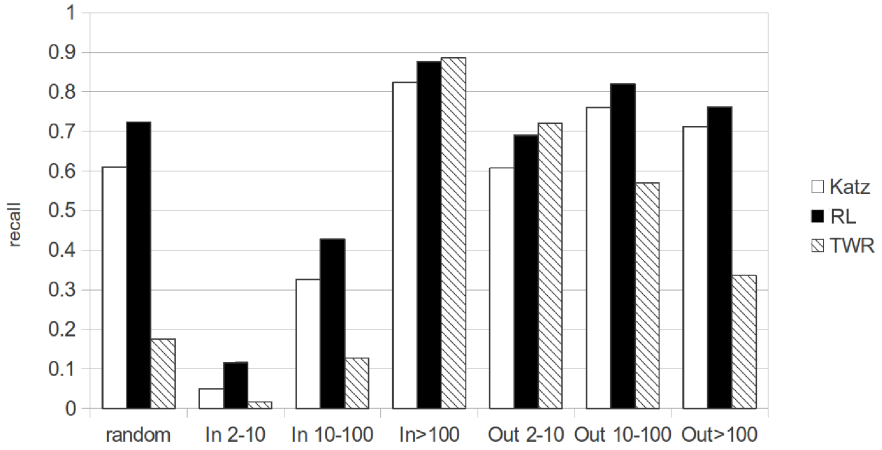

FIGURE 5.5 – Rappel suivant la stratégie de suppression d'arêtes

Étant donné la distribution biaisée des *topics* dans le graphe social (voir Figure 3.5 en Section 3.6), nous avons décidé d'étudier l'impact de la popularité d'un *topic* sur les recommandations générées. Les résultats sont illustrés en Figure 5.6 pour les *topics* `social`, `leisure` et `technology`.

Deux observations apparaissent à la lecture des résultats de cette expérience. Premièrement, les meilleurs valeurs de rappel sont obtenues pour les *topics* les moins populaires. Par exemple, pour un *topic* peu fréquent comme `social`, nous obtenons une valeur de rappel pour les 10 premières recommandations (top-10) de $0,959$ pour RECLAND, $0,751$ pour Katz et $0,253$ pour TwitterRank. Inversement, pour un *topic* populaire tel que `technology`, nous obtenons respectivement $0,462$, $0,424$ and $0,09$. En effet, pour un *topic* populaire, plusieurs comptes sont découverts dans un voisinage proche du compte pour lequel les recommandations sont générées avec potentiellement de meilleurs scores que les comptes des arcs retirés, ce qui explique le fait que la valeur absolue du rappel baisse. En second lieu, nous observons que notre approche, RECLAND, surpasse les approches concurrentes dans les trois cas de figures (*topic* populaire, à popularité moyenne, et peu populaire).

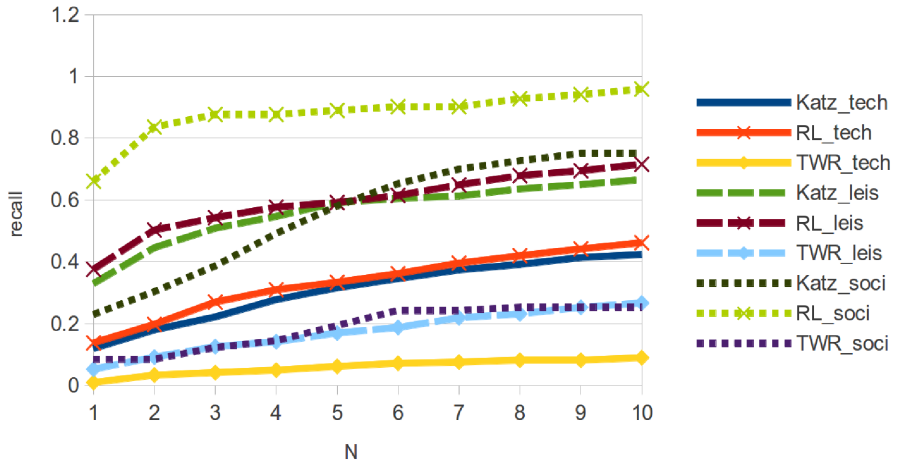

Figure 5.6 – Rappel suivant la popularité des topics

L'ensemble des expériences présentées ci-dessus permet de tester la capacité des algorithmes à prédire les nouveaux liens crées sur le réseau. Afin de valider l'aspect qualitatif des recommandations générées nous avons soumis les 3 approches concurrentes à une étape de validation par les utilisateurs.

5.5.3 Validation par les utilisateurs

Afin d'évaluer la pertinence des recommandations générées, nous avons conduit une tâche de validation par 54 utilisateurs (étudiants, doctorants et membres du laboratoire) parmi lesquels 46% se définissent comme utilisateurs de plateformes sociales tel que *Twitter*. Nous avons mis en place un sondage en ligne où les utilisateurs doivent évaluer la pertinence d'un ensemble de recommandations pour un *topic* donné sur une échelle de 1 (faible pertinence) à 5 (forte pertinence).

Un ensemble de recommandations à évaluer est constitué par le top-3 des recommandations générées respectivement par *Katz*, TwitterRank ainsi que RecLand. Nous avons donc 9 comptes recommandés pour les *topic* `Technology`, `Social` et `Leisure`. Sur l'interface du sondage en ligne, la liste des recommandations est triée aléatoirement, et pour chaque recommanda-

tion est affichée une liste de 5 publications (*tweets*), choisies aléatoirement parmi les publications du compte recommandé. La Figure 5.7 illustre un aperçu de l'interface de notre système d'évaluation.

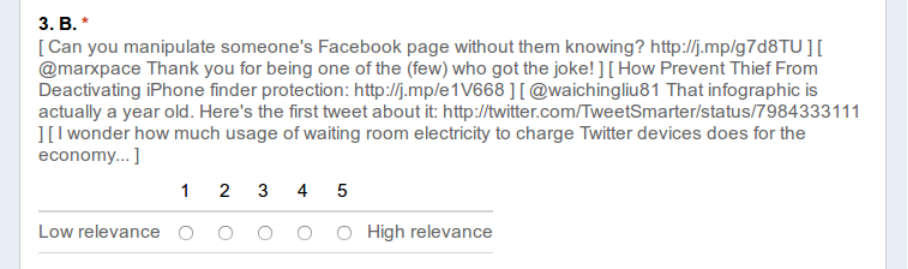

Figure 5.7 – Extrait de l'interface du système d'évaluation pour le *topic* `Technology`

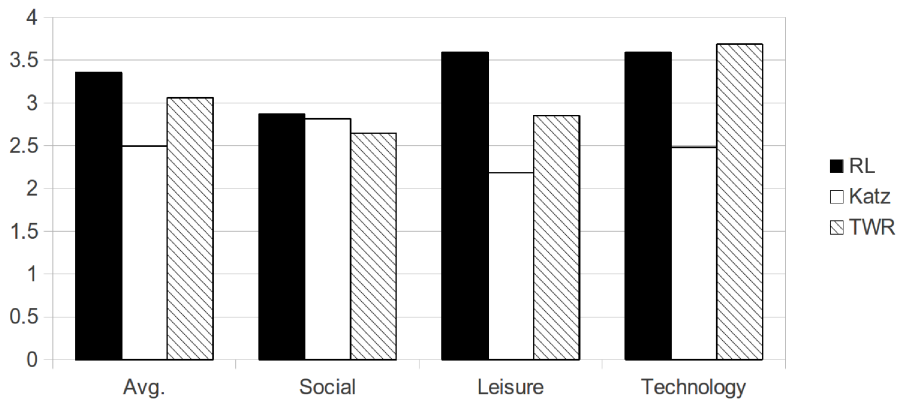

Figure 5.8 – Scores de pertinence obtenus par la validation par les utilisateurs

La Figure 5.8 présente les résultats de notre sondage. Ces résultats montrent que RecLand et TwitterRank fournissent les recommandations les plus pertinentes. Cependant, selon la popularité du *topic* pour lequel les recommandations sont générées, des différences apparaissent. Par exemple, on observe que le *topic* `social` offre des résultat homogènes avec des scores oscillant entre $2,7$ pour TwitterRank, $2,8$ pour Katz et $2,9$ pour RecLand. La raison est que la définition de ce topic est vaste. En effet les comptes

catégorisés `social` sont souvent des comptes généralistes qui publient sur plusieurs sujets tandis que les *topics* `leisure` or `technology` sont moins ambigus.

Pour ces *topics*, on observe que RecLand et TwitterRank surpassent *Katz*, ce qui s'explique par l'avantage que ces approches ont en considérant le contenu publié en plus de la topologie seule. Ainsi RecLand obtient un meilleur score de pertinence sur les *topic* moyennement populaires comme `leisure` grâce à sa capacité à recommander des comptes peu suivis, mais spécialisés sur un sujet. par contre, sur un sujet très populaire comme `technology`, TwitterRank obtient un score légèrement meilleur.

Après avoir testé la pertinence ainsi que la qualité des recommandations générées, nous avons mis en place des expérimentations qui permettent d'évaluer le gain apporté par notre approche par *landmarks*.

5.5.4 Approximation du calcul de recommandations

Nous avons conduit une série d'expériences afin d'illustrer les bénéfices de notre approche basée sur les *landmarks* pour le calcul approché des scores de recommandation. Étant donné le fait que les résultats dépendent grandement du choix des *landmarks*, comme le montre [Potamias et al., 2009], nous avons décidé d'implanter et de comparer les recommandations en se basant sur 11 stratégies de sélection de *landmarks* différentes (présentées dans le Tableau 5.2).

Construction des index de *landmarks*

Cette première expérience met en avant l'écart important qui peut exister dans la phase de sélection des *landmarks*. Le Tableau 5.3), illustre les temps moyen de sélection et de calcul pour un *landmark* pour chacune des stratégies de sélection.

Il apparait évident que les stratégies aléatoires telles que Random ou Btw-Pub sont les plus rapides ($\approx 1ms$ par *landmark*) tandis que les stratégies basées sur la mesure de centralité sont 4 ordres de grandeurs plus lentes avec des temps moyens variant entre 30 et $60s$ (en raison de la complexité de l'ordre de $O(N^2.\log N + NE)$ de la mesure de centralité en appliquant l'algorithme de [Johnson, 1977]). Le tableau 5.3 illustre aussi les temps né-

Algorithme	Description
Random	Choix des *landmarks* avec une distribution uniforme
Follow	Sélection des *landmarks* grâce à une probabilité dépendant de leur # de followers
Publish	Sélection des *landmarks* grâce à une probabilité dépendant du # de comptes qu'ils suivent
In-Deg	*Landmarks* choisis parmi les nœuds ayant le plus grand degré entrant
Btw-Fol	Sélection des *landmarks* avec # de *followers* entre [min_follow, max_follow]
Out-Deg	*Landmarks* choisis parmi les nœuds ayant le plus grand degré sortant
Btw-Pub	Sélection des *landmarks* avec # de comptes suivis entre [min_publish, max_publish]
Central	Les *landmarks* sont les nœuds atteignables à une certaine distance à partir d'un ensemble de nœuds "*seeds*"
Out-Cen	Les *landmarks* sont les nœuds qui atteignent le plus de nœuds d'un ensemble de "*seeds*"
Combine	Combinaison pondérée de Central et Out-Cen
Combine2	Combinaison pondérée de Btw-Fol et Btw-Pub

Table 5.2 – Algorithmes de sélection de *Landmarks*

cessaires au calcul des tables de recommandation pour chaque *landmark*. Nous pouvons observer que ce calcul est indépendant de la stratégie de sélection ce qui signifie que la convergence est atteinte en un nombre similaire d'itérations et en explorant un nombre similaire de chemins.

Comparaison des stratégies de sélection de *landmarks*

Dans cette expérience, nous évaluons la stratégie de sélection de *landmarks* présenté en Section 5.4.3. Nous effectuons une exploration *BFS* à profondeur 2 depuis un nœud donné puis nous combinons les scores obtenus avec les scores des *landmarks* découverts en appliquant l'algorithme 2. Nous comparons ensuite les recommandations obtenues par le calcul approché avec celles calculées à la convergence. Les résultats moyens pour 100 *landmarks* sont reportés sur le tableau 5.4.

D'abord, nous observons que le nombre de *landmarks* rencontrés pendant l'exploration *BFS* à distance 2 varie d'une stratégie à l'autre allant de $7,3$

Stratégie	*Landmarks*	
	sélection. (ms)	calcul. (s)
Random	0.6	27.0
Follow	179.0	30.1
Publish	172.4	29.9
In-Deg	62.9	31.3
Btw-Fol	7.5	31.8
Out-Deg	60.4	30.4
Btw-Pub	1.7	29.7
Central	30,603.0	29.1
Out-Cen	33,193.1	27.8
Combine	65,223.1	27.3
Combine2	67,382.0	28.3

Table 5.3 – Temps moyen de sélection et de calcul pour un *landmark* par stratégie de sélection

en moyenne pour les stratégies de type Combine à $25,4$ pour la stratégie Out-Deg. Les mesures basées sur la centralité affichent le moins de *landmarks* trouvés. Ceci est du au fait que ces mesures choisissent les *landmarks* parmi les nœuds qui forment des "ponts" entre des composantes connexes du graphe et donc plus difficilement atteignables par une exploration à profondeur 2.

Il apparait aussi que le temps de traitement est inversement proportionnel au nombre de *landmarks* trouvés, ce qui peut sembler contre-intuitif étant donné que plus de calculs (combinaisons de scores) sont exécutés s'il y a un nombre important de *landmarks*. L'explication réside dans le fait qu'un "*pruning*" est effectué à la rencontre d'un *landmark* pendant le *BFS* c.à.d que les chemins qui passent par un *landmark* ne sont pas considérés pour l'exploration. Le temps de calcul étant largement dominé par l'étape d'exploration, ce choix de *pruning* réduit considérablement le temps global de traitement. Nous observons que le calcul approché permet un gain allant de 2 à 3 ordres de grandeurs en comparaison avec l'approche exacte.

Les 3 dernières colonnes du Tableau 5.4 affichent les valeurs de distance de *Kendall Tau* entre les résultats obtenus par la méthode approchée et la méthode exacte (à la convergence) lorsque les *landmarks* stockent respectivement les top-10, top-100 et top-1000 recommandations. La distance de

Stratégie	#lnd	Temps en ms (gain)		L10	L100	L1000
Random	11.0	24	(225)	0.130	0.124	0.125
Follow	17.8	9	(599)	0.377	0.140	0.096
Publish	14.9	12	(449)	0.349	0.136	0.100
In-Deg	25.0	6	(899)	0.523	0.149	0.066
Btw-Fol	20.8	25	(216)	0.061	0.059	0.058
Out-Deg	25.4	7	(770)	0.518	0.147	0.064
Btw-Pub	12.7	24	(225)	0.129	0.127	0.123
Central	13.6	24	(225)	0.134	0.123	0.125
Out-Cen	8.8	19	(284)	0.172	0.131	0.121
Combine	7.3	18	(300)	0.180	0.125	0.118
Combine2	10.5	22	(245)	0.129	0.126	0.124

Table 5.4 – Comparaison des stratégies de sélection de *landmarks*

Kendall Tau[5] est une mesure utilisée afin de comparer deux listes classées. Plus cette valeur est importante, plus les classements comparés sont dissimilaires. Nous observons par exemple que les stratégies Out-Deg et In-Deg présentent des résultats similaires par rapport à la méthode exacte. En effet, il apparait que les top-100 comptes recommandés par les deux stratégies sont très similaires.

Le fait de garder le top-1000 recommandations permet d'atteindre des valeurs de distance de *Kendall Tau* oscillant entre $0,06$ et $0,13$. En effet, un classement de 1000 compte ne va pas être très impacté par une exploration à la convergence. Par contre, on observe qu'un top-10 va quant à lui impliquer des valeurs de *Kendall Tau* plus importantes car un classement sur 10 comptes seulement va être fortement impacté par une exploration plus approfondie (à la convergence).

Aussi, on remarque que pour les *landmarks* ayant un fort degré sortant (la distribution de degrés en Figure 3.3 révèle que ce sont aussi ces mêmes comptes qui ont un fort degré entrant) les listes à top-10 varient grandement de l'approche exacte. Ceci est du simplement au fait que ces comptes sont capables d'atteindre très vite (à saut 1 ou 2) un très grand nombre de nœuds .

Enfin, il important de noter que, même en gardant les top-1000 recommandations pour chaque *topic*, l'espace occupé par l'index des recommandations d'un *landmark* n'est que de 1,4 Mo. Nous pouvons donc aisément

5. http://en.wikipedia.org/wiki/Kendall_tau_distance

gérer les tables de *landmarks* en mémoire centrale.

5.6 Conclusion

Nous avons présenté dans ce chapitre RecLand, notre score de recommandation qui intègre la topologie du graphe et les informations sémantiques concernant les intérêts des utilisateurs disponibles sur le *LSG* afin de recommander des comptes à suivre. Afin de faire face aux éventuels calculs prohibitifs inhérents aux larges graphes, nous avons proposé une approche basée sur les *landmarks* qui nécessite une étape de pré-traitement pour un ensemble réduit et identifié de nœuds du graphe. Cette approche permet d'atteindre un gain de 2 à 3 ordres de grandeur en comparaison avec l'approche exacte. Les expérimentations conduites ainsi que la validation par les utilisateurs ont permis de démontrer que RecLand surpasse les approches concurrentes.

Comme pistes d'évolutions, nous envisageons d'abord l'étude de stratégies de mises à jour. En effet, comme reporté dans le Chapitre 4, les liens dans les réseaux sociaux peuvent avoir un cycle de vie très éphémère. La dynamicité du graphe impactant directement nos scores, nous avons donc besoin de mettre en place des stratégies de rafraichissement adaptées.

Enfin, nous avons fait le choix de gérer la recommandation de manière centralisée. Cette décision a été motivée par le fait que pour la manipulation de larges graphes, le coût de la distribution des structure peut très vite outrepasser le gain induit par l'approche de calcul distribué. Par exemple, le système de recommandation mis en production pour gérer l'ensemble de la tâche de recommandation de comptes à suivre chez *Twitter* est hébergé sur un seul serveur physique [Gupta et al., 2013]. Cependant, avec la croissance continue des tailles de graphe, des stratégies intelligentes de distribution doivent êtres élaborées. Dans notre approche, nous pouvons envisager la distribution du graphe en prenant en compte la connectivité des différentes composantes, mais aussi en combinant la distribution avec la sélection de *landmarks* ce qui permettrait d'effectuer les calculs de pré-traitement localement, minimisant ainsi les transferts sur le réseau.

Chapitre 6

Conclusion

Le travail présenté dans cette thèse a pour but d'améliorer l'expérience des utilisateurs des plateformes sociales via l'introduction de deux mécanismes complémentaires qui sont le **filtrage** de flux sociaux ainsi que la **recommandation** de comptes à suivre sur le réseau.
Ce chapitre résume les différentes contributions de cette thèse et se referme en présentant les perspectives de recherches envisagées pour ce travail.

6.1 Contributions

Après avoir mené une étude de l'existant qui a permis la caractérisation des données sociales. Nous avons proposé le modèle de *LSG* (*Labelled Social Graph*) qui permet de capturer les intérêts des utilisateurs sous forme de graphe étiqueté. En se basant sur ce modèle, nous avons ensuite généré divers jeux de données en exploitant des données réelles provenant du réseau social *Twitter*.

La quantité de données à laquelle sont exposés les utilisateurs des plateformes sociales ne cesse de croitre. Dans ce contexte, la première contribution de ce travail de thèse est la mise en place d'un système de filtrage personnalisé pour les systèmes du type micro-blogging nommé MicroFilter [Dahimene et al., 2012, Dahimene and du Mouza, 2013a, Dahimene and du Mouza, 2013b, Dahimene et al.b].
Nous avons mis en place diverses stratégies d'indexation basées sur des structures de listes inverses. Chacune des structures proposées exploite une fac-

torisation spécifique aux données issues des plateformes de micro-blogging. Après une estimation analytique des différentes propositions, nous avons testé expérimentalement ces stratégies de filtrage afin d'analyser leur comportement en situation de passage à l'échelle. La validation expérimentale nous a ensuite orienté sur la piste d'une approche hybride, exploitant ainsi l'hétérogénéité des données sociales afin d'offrir un compromis intéressant en terme d'occupation mémoire et de temps de recherche (*matching*).

Basé sur le même modèle de graphe social étiqueté, la seconde contribution de notre travail consiste en la mise en place d'un système de recommandation de comptes à suivre sur une plateforme de type micro-blogging. Nous avons appelé ce système RecLand [Dahimene et al., 2014, Dahimene et al.a].
L'idée principale de cette stratégie de recommandation consiste en l'exploitation de l'ensemble des données implicites au modèle de *LSG*. Chacune des arêtes étiquetées représentant l'intérêt d'un utilisateur sur les publications d'un compte donné, nous avons proposé un score de recommandation de comptes à suivre axé sur trois dimensions :

(*i*) *Une dimension topologique*. En se basant sur l'idée que les mesures topologiques sur les graphes sociaux sont un bon indicateur de similarité [Liben-Nowell and Kleinberg, 2003], nous avons exploité cet aspect pour une partie de notre score en proposant une extension du score de *Katz*.

(*ii*) *Une dimension sémantique*. Chaque arc contenant une information sur l'intérêt des utilisateurs, nous exploitons cette donnée en mesurant une distance de similarité sémantique.

(*iii*) *Une dimension d'autorité* qui permet de mesurer l'influence d'un compte sur un sujet (*topic*) donné en analysant la topologie des arêtes du graphe autour du compte. Nous avons décomposé cette composante du score en une mesure d'autorité locale et une mesure d'autorité globale.

L'avantage principal de cette approche est qu'elle permet l'exploitation d'informations sémantiques sans avoir à analyser le contenu des publications. Ces informations sont puisées directement dans la structure du graphe *LSG*. Afin de pouvoir répondre à la contrainte de passage à l'échelle, nous avons introduit un algorithme qui permet d'estimer une valeur approchée de notre

score de recommandation. Cette approche se base sur un ensemble de nœuds spéciaux appelés "*landmarks*" qui sont responsable de stocker un ensemble d'informations sur leur voisinage proche. La recommandation pour un compte donné s'effectue par la suite en explorant le graphe social jusqu'à une profondeur fixe, et en combinant les scores de recommandation calculés avec les scores stockés par les différents *landamrks* rencontrés.

Nous avons ensuite validé expérimentalement notre proposition en comparant les résultats obtenus par notre stratégie de recommandation avec les résultats de deux approches concurrentes. Cette validation a aussi été confirmée par un sondage effectué sur un ensemble d'utilisateurs. Enfin nous avons évalué expérimentalement l'impact ainsi que le gain introduit par l'approximation des scores de recommandation en utilisant des *landmarks*.

6.2 Perspectives

Suite à ce travail, nous envisageons les perspectives de recherche suivantes.

Un filtrage intelligent

Nous avons mis en place une structure qui permet de répondre efficacement aux requêtes continues de filtrage lors de l'arrivée d'une nouvelle publication. Les requêtes utilisées dans ce contexte sont un ensemble de mots-clés que nous traitons via une logique disjonctive. Une des perspectives envisagée est de pouvoir gérer le filtrage avec des logiques de requêtes plus complexes en introduisant la conjonction ou bien la négation. Aussi, nous pouvons aussi envisager l'exploitation d'une ontologie (telle que Wordnet) qui permettrait de faire des recherches approchées via une analyse sémantique. La réécriture de requêtes est une autre solution envisageable. Le défi consiste à introduire ces fonctionnalités tout en garantissant le passage à l'échelle.

La gestion des mises à jour

Les arêtes des graphes sociaux peuvent avoir un cycle de vie très éphémère. En effet, on observe une forte dynamicité dans les structures de graphe

où les utilisateurs s'abonnent et se désabonnent fréquemment.

Un des défi qui reste ouvert est la gestion efficace de ces mises à jour dans nos structures d'indexation. Nous avons analysé cet aspect dans un contexte de filtrage et abouti à une structure hybride. Il reste à étudier l'impact de telles modifications sur nos index de recommandation et à élaborer des stratégies de rafraichissement adaptées. Une approche souvent adoptée en production consiste à recalculer les mesures périodiquement. On pourrait envisager une approche plus évoluée en traitant les *landmarks* séparément et en mettant à jour uniquement les tables des *landmarks* concernées par d'éventuels changements de la topologie du graphe.

La distribution

Afin de pouvoir suivre l'évolution des tailles des graphes sociaux, nous devons envisager la distribution des données ainsi que des calculs sur des *clusters* de plusieurs machines.

Certains travaux existants s'intéressent au problème de la distribution de graphes. En effet, choisir une subdivision de graphe est un réel défi qui doit prendre en considération la connectivité du réseau mais aussi minimiser le coût des transferts sur le réseau. Dans cette optique, nous nous intéressons à l'élaboration de stratégies de distribution qui permettraient d'exploiter notre système de recommandation sur un ensemble de nœuds en *cluster*. Nous souhaitons aussi travailler sur une approche qui permettrait d'introduire du parallélisme dans le calcul des scores de recommandations. Ce parallélisme pourrait exister entre les nœuds des machines d'un même *cluster*, mais aussi sur une même machine en exploitant l'architecture multi-cœurs des processeurs modernes.

Une granularité plus fine

Actuellement, notre système de recommandation est capable de générer pour un utilisateur une liste triée par pertinence de comptes à suivre sur un sujet donné.

Une piste envisagée consisterait à fournir des recommandations au niveau du *tweet*. L'approche consisterait à proposer un ensemble de publications jugées pertinentes à un utilisateurs. Deux principaux challenges sont

introduits par cette approche : (*i*) La garantie de "*fraicheur*" des publications suggérées : en effet, l'aspect temps-réel est une composante principale des plateforme sociales, le système de devra être capable de fournir des recommandations "à la volée", (*ii*) Les taux de publication sur les plateformes sociales peuvent atteindre des pics prohibitifs (plus de 100.000 publications à la seconde), le système devra être capable de suivre le rythme tout en répondant efficacement aux requêtes de recommandation.

Annexe A

Captures d'écran de MicroFilter

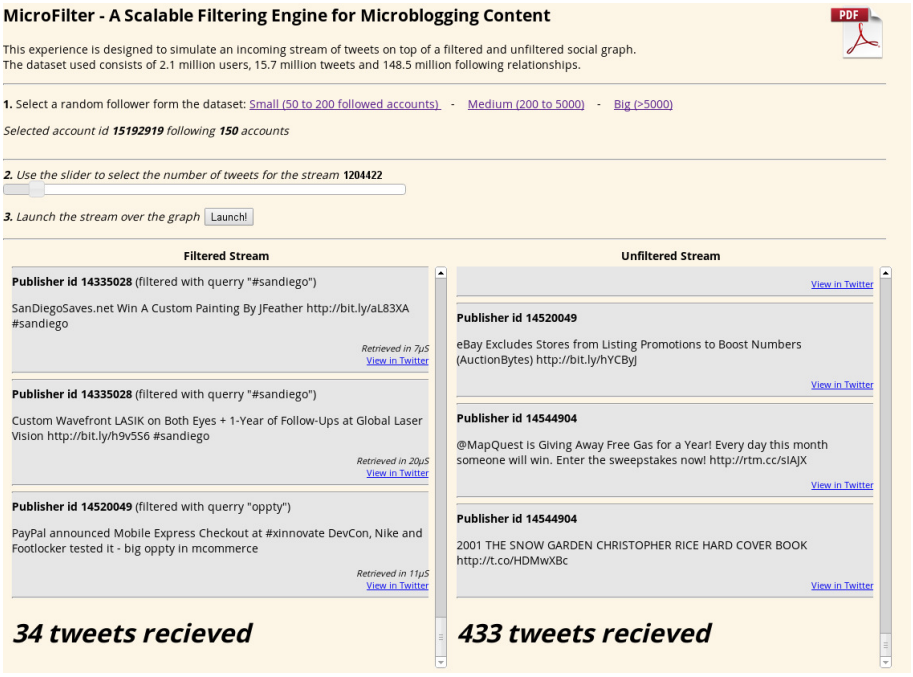

FIGURE A.1 – MICROFILTER : Vue d'ensemble

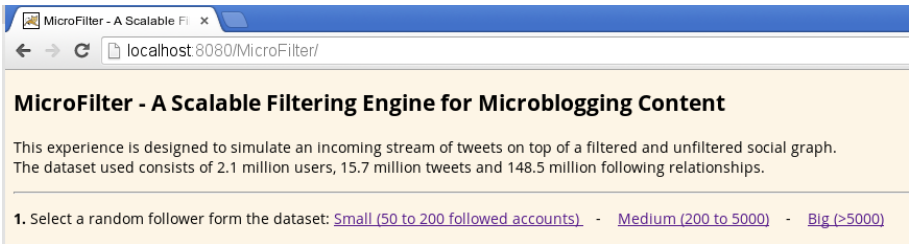

Figure A.2 – MicroFilter : Sélection d'un compte

Figure A.3 – MicroFilter : Paramètres

Figure A.4 – MicroFilter : Flux

Annexe B

Captures d'écran de RecLand

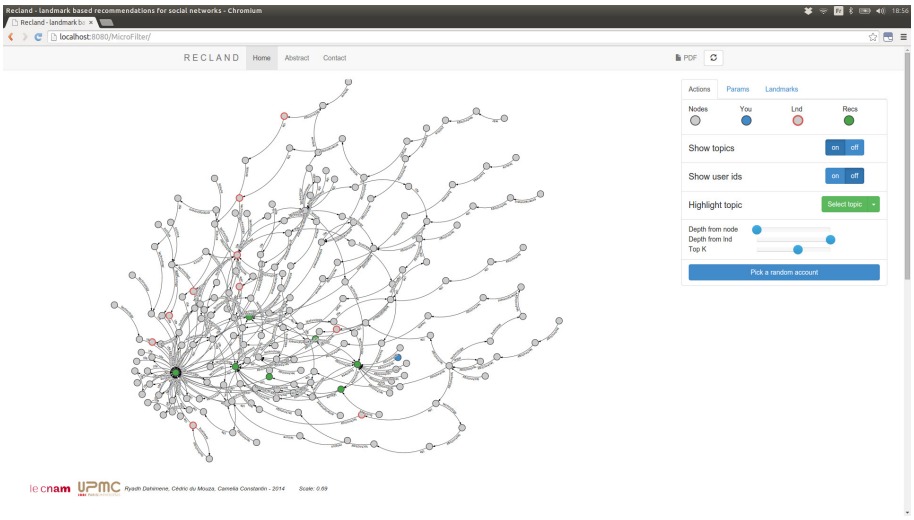

FIGURE B.1 – RECLAND : Vue d'ensemble

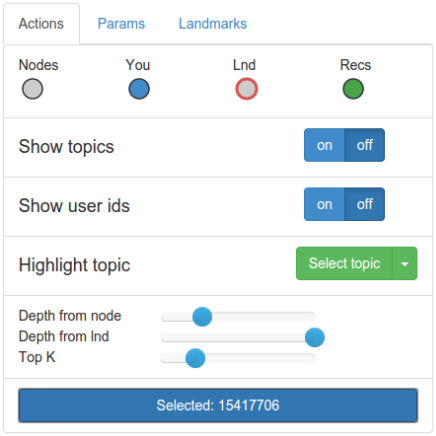

Figure B.2 – RecLand : Sélection d'un compte

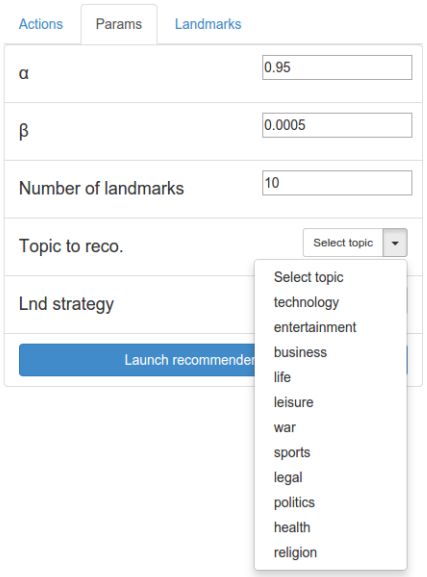

Figure B.3 – RecLand : Paramètres

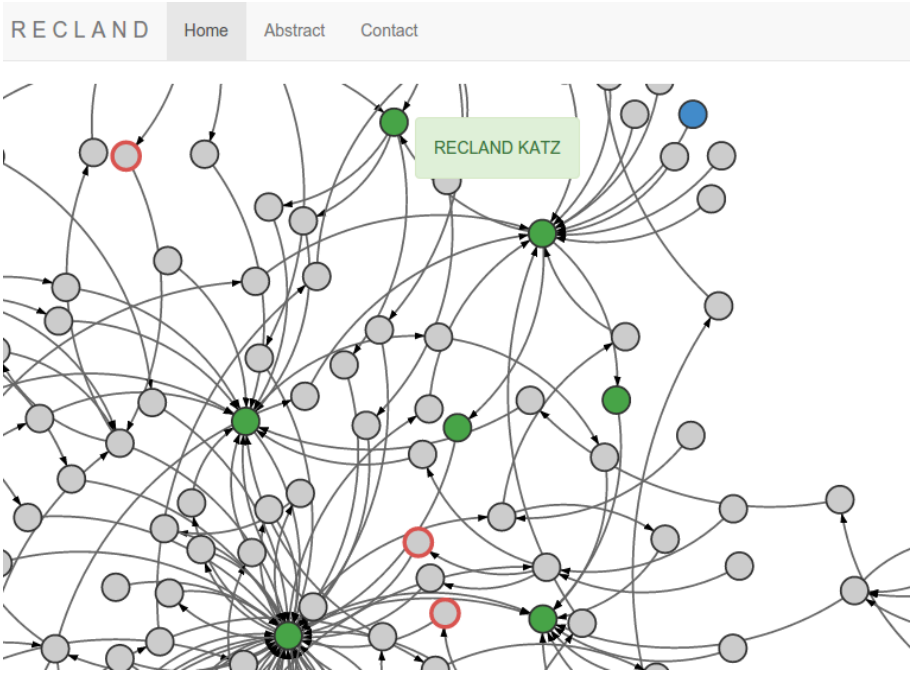

Figure B.4 – RecLand : Affichage des résultats

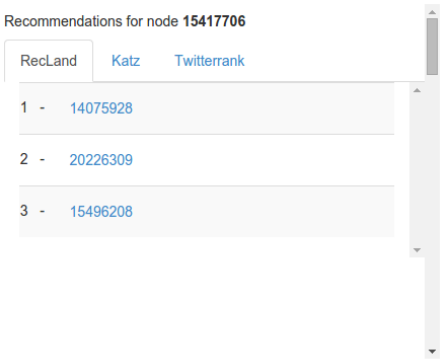

Figure B.5 – RecLand : Top-k générés

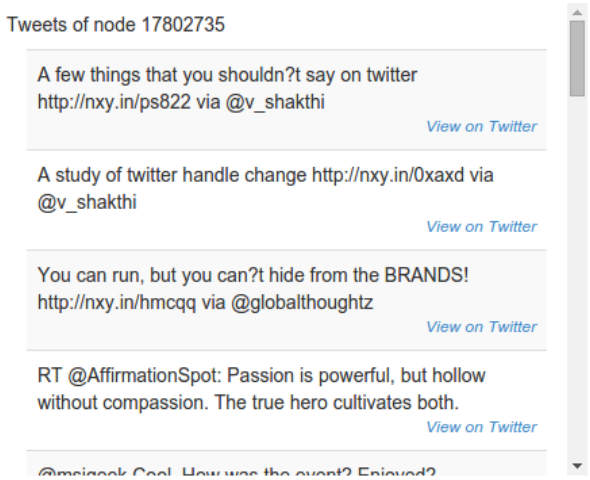

Figure B.6 – RecLand : Affichage des *Tweets*

Bibliographie

[Achananuparp et al., 2012] Achananuparp, P., Lim, E.-P., Jiang, J., and Hoang, T.-A. (2012). Who is Retweeting the Tweeters ? Modeling, Originating, and Promoting Behaviors in the Twitter Network. *Jour. on Transactions on Management Information Systems (TMIS)*, 3(3) :13.

[Adomavicius and Tuzhilin, 2005] Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems : A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6) :734–749.

[Ahn et al., 2007] Ahn, Y., Han, S., Kwak, H., Moon, S. B., and Jeong, H. (2007). Analysis of topological characteristics of huge online social networking services. In *Proc. Intl. World Wide Web Conference (WWW)*, pages 835–844.

[Albakour et al., 2013] Albakour, M., Macdonald, C., and Ounis, I. (2013). On sparsity and drift for effective real-time filtering in microblogs. In *Proc. Intl. Conf. on Information and Knowledge Management (CIKM)*, pages 419–428.

[Amatriain, 2013] Amatriain, X. (2013). Big & personal : data and models behind netflix recommendations. In *Proceedings of the 2nd International Workshop on Big Data, Streams and Heterogeneous Source Mining : Algorithms, Systems, Programming Models and Applications, BigMine 2013, Chicago, IL, USA, August 11, 2013*, pages 1–6.

[Anderson, 2006] Anderson, C. (2006). *The Long Tail : Why the Future of Business Is Selling Less of More*. Hyperion.

[Backstrom et al., 2012] Backstrom, L., Boldi, P., Rosa, M., Ugander, J., and Vigna, S. (2012). Four degrees of separation. In *Proc. Intl. Conf. on Web Science (WebSci)*, pages 33–42.

[Baeza-Yates and Ribeiro-Neto, 1999] Baeza-Yates, R. A. and Ribeiro-Neto, B. A. (1999). *Modern Information Retrieval*. ACM Press / Addison-Wesley.

[Bakshy et al., 2011] Bakshy, E., Hofman, J. M., Mason, W. A., and Watts, D. J. (2011). Everyone's an Influencer : Quantifying Influence on Twitter. In *Proc. Intl. Conf. on Web Search and Web Data Mining (WSDM)*, pages 65–74.

[Bergen] Bergen, J. Facebook stores 10,000x more photos than the Library of Congress.
http://www.geek.com/geek-cetera/facebook-stores-10000x-more-photos-than-the-library-of-congress-1422873/. Accessed : 12/08/2014.

[Boldi et al., 2011] Boldi, P., Rosa, M., and Vigna, S. (2011). Hyperanf : approximating the neighbourhood function of very large graphs on a budget. In *Proc. Intl. World Wide Web Conference (WWW)*, pages 625–634.

[Bonchi et al., 2012] Bonchi, F., Esfandiar, P., Gleich, D. F., Greif, C., and Lakshmanan, L. V. S. (2012). Fast Matrix Computations for Pairwise and Columnwise Commute Times and Katz Scores. *Internet Mathematics*, 8(1-2) :73–112.

[Bontcheva et al., 2013] Bontcheva, K., Gorrell, G., and Wessels, B. (2013). Social media and information overload : Survey results. *CoRR*, abs/1306.0813.

[Brin and Page, 1998] Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7) :107–117.

[Broder et al., 2011] Broder, A. Z., Das, S., Fontoura, M., Ghosh, B., Josifovski, V., Shanmugasundaram, J., and Vassilvitskii, S. (2011). Efficiently Evaluating Graph Constraints in Content-Based Publish/Subscribe. In *Proc. Intl. World Wide Web Conference (WWW)*, pages 497–506.

[Budalakoti and Bekkerman, 2012] Budalakoti, S. and Bekkerman, R. (2012). Bimodal Invitation-Navigation Fair Bets Model for Authority Identification in a Social Network. In *Proc. Intl. World Wide Web Conference (WWW)*, pages 709–718.

[Busch et al., 2012] Busch, M., Gade, K., Larson, B., Lok, P., Luckenbill, S., and Lin, J. (2012). Earlybird : Real-time search at twitter. In *Proc. Intl. Conf. on Data Engineering (ICDE)*, pages 1360–1369.

[Cha et al., 2010] Cha, M., Haddadi, H., Benevenuto, F., and Gummadi, P. K. (2010). Measuring User Influence in Twitter : The Million Follower Fallacy. In *Proc. Intl. Conf. on Weblogs and Social Media (ICWSM)*.

[Chaoji et al., 2012] Chaoji, V., Ranu, S., Rastogi, R., and Bhatt, R. (2012). Recommendations to Boost Content Spread in Social Networks. In *Proc. Intl. World Wide Web Conference (WWW)*, pages 529–538.

[Chen et al., 2011] Chen, C., Li, F., Ooi, B. C., and Wu, S. (2011). TI : an efficient indexing mechanism for real-time search on tweets. In *Proc. Intl. Conf. on Management of Data (SIGMOD)*, pages 649–660.

[Chen et al., 2012] Chen, K., Chen, T., Zheng, G., Jin, O., Yao, E., and Yu, Y. (2012). Collaborative personalized tweet recommendation. In *Proc. Intl. Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 661–670.

[Cheng and Evans] Cheng, A. and Evans, M. An In-Depth Look at the 5% of Most Active Users.
http://www.sysomos.com/insidetwitter/mostactiveusers/. Accessed : 12/08/2014.

[Cremonesi et al., 2010] Cremonesi, P., Koren, Y., and Turrin, R. (2010). Performance of Recommender Algorithms on Top-n Recommendation Tasks. In *Proc. Intl. Conf. on Recommender Systems (RECSYS)*, pages 39–46.

[Dahimene et al.a] Dahimene, R., Constantin, C., and du Mouza, C. Topic-sensitive user recommendation in micro-blogging systems. In *Under review.*

[Dahimene et al., 2014] Dahimene, R., Constantin, C., and du Mouza, C. (2014). Recland : A recommender system for social networks. In *Proc. Intl. Conf. on Information and Knowledge Management (CIKM)*, pages 1–3.

[Dahimene and du Mouza, 2013a] Dahimene, R. and du Mouza, C. (2013a). Microfilter : real time filtering of microblogging content. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013, Companion Volume*, pages 633–634.

[Dahimene and du Mouza, 2013b] Dahimene, R. and du Mouza, C. (2013b). Microfilter : Scalable real-time filtering of micro-blogging content. In *BDA'13, Base de Données Avancées, Octobre 2013, pp.1–5, Nantes.*

[Dahimene et al.b] Dahimene, R., du Mouza, C., and Scholl, M. Filtering structures for microblogging content. In *Under Review.*

[Dahimene et al., 2012] Dahimene, R., du Mouza, C., and Scholl, M. (2012). Efficient Filtering in Micro-blogging Systems : We Won't Get Flooded Again. In *Proc. Intl. Conf. on Scientific and Statistical Databases (SSDBM)*, pages 168–176.

[Das et al., 2007] Das, A. S., Datar, M., Garg, A., and Rajaram, S. (2007). Google news personalization : Scalable online collaborative filtering. In *Proc. Intl. World Wide Web Conference (WWW)*, pages 271–280.

[Diaz-Aviles et al., 2012] Diaz-Aviles, E., Drumond, L., Gantner, Z., Schmidt-Thieme, L., and Nejdl, W. (2012). What is happening right now ... that interests me ? : online topic discovery and recommendation in twitter. In *Proc. Intl. Conf. on Information and Knowledge Management (CIKM)*, pages 1592–1596.

[Diederich and Iofciu, 2006] Diederich, J. and Iofciu, T. (2006). Finding communities of practice from user profiles based on folksonomies. In *Proceedings of the EC-TEL06 Workshops, Crete, Greece , October 1-2, 2006*.

[DMRa] DMR. By the Numbers : 130 Amazing Facebook User and Demographic Statistics. http://expandedramblings.com/index.php/by-the-numbers-17-amazing-facebook-stats/. Accessed : 12/08/2014.

[DMRb] DMR. By the Numbers : 215 Amazing Twitter Statistics. http://expandedramblings.com/index.php/march-2013-by-the-numbers-a-few-amazing-twitter-stats/. Accessed : 12/08/2014.

[Dunbar, 1992] Dunbar, R. (1992). Neocortex size as a constraint on group size in primates. *Journal of Human Evolution*, 22(6) :469 – 493.

[Facebook] Facebook. Q2 2014 Earnings. http://files.shareholder.com/downloads/AMDA-NJ5DZ/3349478089x0x770377/abc6b6d4-df03-44e1-bb4d-7877f01c41e0/. Accessed : 12/08/2014.

[Foster et al., 2011] Foster, J., Özlem Çetinoğlu, Wagner, J., Roux, J. L., Hogan, S., Nivre, J., Hogan, D., and van Genabith, J. (2011). #hardtoparse : POS Tagging and Parsing the Twitterverse. In *Proc. Intl. Work. on Analyzing Microtext (AMW)*.

[Garcia-Silva et al., 2012] Garcia-Silva, A., Kang, J.-H., Lerman, K., and Corcho, O. (2012). Characterising emergent semantics in twitter lists. pages 530–544.

[Golder and Yardi, 2010] Golder, S. A. and Yardi, S. (2010). Structural Predictors of Tie Formation in Twitter : Transitivity and Mutuality. In *Proc. Intl. Conf. on Social Computing (SocialCom) / IEEE Intl. Conf. on Privacy, Security, Risk and Trust (PASSAT)*, pages 88–95.

[Gonzalez et al., 2011] Gonzalez, R., Rumín, R. C., Cuevas, Á., and Guerrero, C. (2011). Where are my followers ? understanding the locality effect in twitter. *CoRR*.

[Gubichev et al., 2010] Gubichev, A., Bedathur, S. J., Seufert, S., and Weikum, G. (2010). Fast and accurate estimation of shortest paths in large graphs. In *CIKM*, pages 499–508.

[Gupta et al., 2013] Gupta, P., Goel, A., Lin, J., Sharma, A., Wang, D., and Zadeh, R. (2013). WTF : the Who to Follow Service at Twitter. In *Proc. Intl. World Wide Web Conference (WWW)*, pages 505–514.

[Haghani et al., 2010] Haghani, P., Michel, S., and Aberer, K. (2010). The Gist of Everything New : Personalized Top-k Processing over Web 2.0 Streams. In *Proc. Intl. Conf. on Information and Knowledge Management (CIKM)*, pages 489–498.

[Han and Baldwin, 2011] Han, B. and Baldwin, T. (2011). Lexical Normalisation of Short Text Messages : Makn Sens a #twitter. In *Proc. Intl. Meeting of the Association for Computational Linguistics : Human Language Technologies (ACL)*, pages 368–378.

[Hannon et al., 2010] Hannon, J., Bennett, M., and Smyth, B. (2010). Recommending twitter users to follow using content and collaborative filtering approaches. In *Proc. Intl. Conf. on Recommender Systems (RECSYS)*, pages 199–206.

[Hmedeh et al., 2012] Hmedeh, Z., Kourdounakis, H., Christophides, V., du Mouza, C., Scholl, M., and Travers, N. (2012). Subscription Indexes for Web Syndication Systems. In *Proc. Intl. Conf. on Extending Database Technology (EDBT)*, pages 1–12.

[IBM] IBM. What's Big data ?
http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html. Accessed : 08/08/2014.

[Java et al., 2007] Java, A., Song, X., Finin, T., and Tseng, B. L. (2007). Why We Twitter : An Analysis of a Microblogging Community. In *Proc. Intl. Work. on Advances in Web Mining and Web Usage Analysis (SNA-KDD)*, pages 118–138.

[Jeh and Widom, 2003] Jeh, G. and Widom, J. (2003). Scaling Personalized Web Search. In *Proc. Intl. World Wide Web Conference (WWW)*, pages 271–279.

[Johnson, 1977] Johnson, D. B. (1977). Efficient algorithms for shortest paths in sparse networks. *J. ACM*, 24(1) :1–13.

[JORF] JORF. Journal Officiel de la République Française.
http://www.legifrance.gouv.fr/affichTexte.do?cidTexte=JORFTEXT000029388087. Accessed : 22/08/2014.

[Kaplan and Haenlein, 2010] Kaplan, A. M. and Haenlein, M. (2010). Users of the world, unite ! the challenges and opportunities of social media. *Business Horizons*, 53(1) :59 – 68.

[Katz, 1953a] Katz, L. (1953a). A New Status Index Derived from Sociometric Analysis. *Psychometrika*, 18(1) :39–43.

[Katz, 1953b] Katz, L. (1953b). A new status index derived from sociometric analysis. *Psychometrika*, 18(1) :39–43.

[Kivran-Swaine et al., 2011] Kivran-Swaine, F., Govindan, P., and Naaman, M. (2011). The Impact of Network Structure on Breaking Ties in Online Social Networks : Unfollowing on Twitter. In *Proc. Intl. Conf. on Human Factors in Computing Systems (CHI)*, pages 1101–1104.

[Kumar et al., 2003] Kumar, R., Novak, J., Raghavan, P., and Tomkins, A. (2003). On the bursty evolution of blogspace. In *Proc. Intl. World Wide Web Conference (WWW)*, pages 568–576.

[Kwak et al., 2011] Kwak, H., Chun, H., and Moon, S. B. (2011). Fragile Online Relationship : A First Look at Unfollow Dynamics in Twitter. In *Proc. Intl. Conf. on Human Factors in Computing Systems (CHI)*, pages 1091–1100.

[Kwak et al., 2010] Kwak, H., Lee, C., Park, H., and Moon, S. B. (2010). What Is Twitter, a Social Network or a News Media ? In *Proc. Intl. World Wide Web Conference (WWW)*, pages 591–600.

[Kywe et al., 2012] Kywe, S. M., Hoang, T.-A., Lim, E.-P., and Zhu, F. (2012). On Recommending Hashtags in Twitter Networks. In *Proc. Intl. Conf. on Social Informatics (SOCINFO)*, pages 337–350.

[Laboreiro et al., 2010] Laboreiro, G., Sarmento, L., Teixeira, J., and Oliveira, E. (2010). Tokenizing Micro-blogging Messages Using a Text Classification Approach. In *Proc. Intl. Work. on Analytics for Noisy Unstructured Text Data (AND)*, pages 81–88.

[Lempel and Moran, 2001] Lempel, R. and Moran, S. (2001). Salsa : The stochastic approach for link-structure analysis. *ACM Transactions on Information Systems*, 19(2) :131–160.

[Liang et al., 2012] Liang, H., Xu, Y., Tjondronegoro, D., and Christen, P. (2012). Time-aware topic recommendation based on micro-blogs. In *Proc. Intl. Conf. on Information and Knowledge Management (CIKM)*, pages 1657–1661.

[Liben-Nowell and Kleinberg, 2003] Liben-Nowell, D. and Kleinberg, J. (2003). The Link Prediction Problem for Social Networks. In *Proc. Intl. Conf. on Information and Knowledge Management (CIKM)*, pages 556–559.

[Linden et al., 2003] Linden, G., Smith, B., and York, J. (2003). Industry report : Amazon.com recommendations : Item-to-item collaborative filtering. *IEEE Distributed Systems Online*, 4(1).

[Liu et al., 2005] Liu, T.-Y., Yang, Y., Wan, H., Zeng, H.-J., Chen, Z., and Ma, W.-Y. (2005). Support Vector Machines Classification with a Very Large-Scale Taxonomy. *SIGKDD Explorations*, 7(1) :36–43.

[Manning et al., 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

[Milgram, 1967] Milgram, S. (1967). The small world problem. *Psychology Today*, 67(1) :61–67.

[Myers et al., 2014] Myers, S. A., Sharma, A., Gupta, P., and Lin, J. (2014). Information network or social network ? : The structure of the twitter follow graph. In *Intl. World Wide Web Conference (WWW) (Companion Volume)*, pages 493–498.

[Pal and Counts, 2011] Pal, A. and Counts, S. (2011). Identifying Topical Authorities in Microblogs. In *Proc. Intl. Conf. on Web Search and Web Data Mining (WSDM)*, pages 45–54.

[Pennacchiotti et al., 2012] Pennacchiotti, M., Silvestri, F., Vahabi, H., and Venturini, R. (2012). Making your interests follow you on twitter. In *Proc. Intl. Conf. on Information and Knowledge Management (CIKM)*, pages 165–174.

[Pereira et al., 2000] Pereira, J., Fabret, F., Llirbat, F., Preotiuc-Pietro, R., Ross, K. A., and Shasha, D. (2000). Publish/subscribe on the web at extreme speed. In *Proc. Intl. Conf. on Very Large Data Bases (VLDB)*, pages 627–630.

[Potamias et al., 2009] Potamias, M., Bonchi, F., Castillo, C., and Gionis, A. (2009). Fast shortest path distance estimation in large networks. In *Proc. Intl. Conf. on Information and Knowledge Management (CIKM)*, pages 867–876.

[Ricci et al., 2011] Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors (2011). *Recommender Systems Handbook*. Springer.

[Romero and Kleinberg, 2010] Romero, D. M. and Kleinberg, J. M. (2010). The Directed Closure Process in Hybrid Social-Information Networks, with an Analysis of Link Formation on Twitter. *CoRR*, abs/1003.2469.

[Sakaki et al., 2010] Sakaki, T., Okazaki, M., and Matsuo, Y. (2010). Earthquake Shakes Twitter Users : Real-time Event Detection by Social Sensors. In *Proc. Intl. World Wide Web Conference (WWW)*, pages 851–860.

[Sankaranarayanan et al., 2009] Sankaranarayanan, J., Samet, H., Teitler, B. E., Lieberman, M. D., and Sperling, J. (2009). TwitterStand : News in Tweets. In *Proc. Intl. Symp. on Geographic Information Systems (ACM-GIS)*, pages 42–51.

[Sarma et al., 2010] Sarma, A. D., Gollapudi, S., Najork, M., and Panigrahy, R. (2010). A sketch-based distance oracle for web-scale graphs. In *Proc. Intl. Conf. on Web Search and Web Data Mining (WSDM)*, pages 401–410.

[Sethu and Chu, 2012] Sethu, H. and Chu, X. (2012). A new algorithm for extracting a small representative subgraph from a very large graph. *CoRR*, abs/1207.4825.

[Silberstein et al., 2010] Silberstein, A., Terrace, J., Cooper, B. F., and Ramakrishnan, R. (2010). Feeding Frenzy : Selectively Materializing Users' Event Feeds. In *Proc. Intl. Conf. on Management of Data (SIGMOD)*, pages 831–842.

[Sriram et al., 2010] Sriram, B., Fuhry, D., Demir, E., Ferhatosmanoglu, H., and Demirbas, M. (2010). Short Text Classification in Twitter to Improve Information Filtering. In *Proc. Intl. Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 841–842.

[Teevan et al., 2011] Teevan, J., Ramage, D., and Morris, M. R. (2011). #TwitterSearch : a Comparison of Microblog Search and Web Search. In *Proc. Intl. Conf. on Web Search and Web Data Mining (WSDM)*, pages 35–44.

[Thorup and Zwick, 2001] Thorup, M. and Zwick, U. (2001). Approximate distance oracles. In *Proc. Annual ACM Symposium on Theory of Computing (STOC)*, pages 183–192.

[Tretyakov et al., 2011] Tretyakov, K., Armas-Cervantes, A., García-Bañuelos, L., Vilo, J., and Dumas, M. (2011). Fast fully dynamic landmark-based estimation of shortest path distances in very large graphs. In *Proc. Intl. Conf. on Information and Knowledge Management (CIKM)*, pages 1785–1794.

[Tumasjan et al., 2010] Tumasjan, A., Sprenger, T. O., Sandner, P. G., and Welpe, I. M. (2010). Predicting elections with twitter : What 140 characters reveal about political sentiment. In *Proc. Intl. Conf. on Weblogs and Social Media (ICWSM)*.

[Ugander et al., 2011] Ugander, J., Karrer, B., Backstrom, L., and Marlow, C. (2011). The anatomy of the facebook social graph. *CoRR*.

[Uysal and Croft, 2011] Uysal, I. and Croft, W. B. (2011). User Oriented Tweet Ranking : A Filtering Approach to Microblogs. In *Proc. Intl. Conf. on Information and Knowledge Management (CIKM)*, pages 2261–2264.

[Velichety and Ram, 2013] Velichety, S. and Ram, S. (2013). Examining lists on twitter to uncover relationships between following, membership and subscription. In *Proc. Intl. World Wide Web Conference (WWW)*, WWW '13 Companion, pages 673–676.

[Vosecky et al., 2012] Vosecky, J., Leung, K. W.-T., and Ng, W. (2012). Searching for Quality Microblog Posts : Filtering and Ranking Based on Content Analysis and Implicit Links. In *Proc. Intl. Conf. on Database Systems for Advanced Applications (DASFAA)*, pages 397–413.

[Watts and Strogatz, 1998] Watts, D. and Strogatz, S. (1998). Collective dynamics of 'small-world' networks. *Nature*, (393) :440–442.

[Weaver] Weaver, E. Evan weaver's blog (software engineer at twitter). http://blog.evanweaver.com. Accessed : 10/06/2012.

[Welch et al., 2011] Welch, M. J., Schonfeld, U., He, D., and Cho, J. (2011). Topical Semantics of Twitter Links. In *Proc. Intl. Conf. on Web Search and Web Data Mining (WSDM)*, pages 327–336.

[Weng et al., 2010] Weng, J., Lim, E.-P., Jiang, J., and He, Q. (2010). TwitterRank : Finding Topic-sensitive Influential Twitterers. In *Proc. Intl. Conf. on Web Search and Web Data Mining (WSDM)*, pages 261–270.

[Wu et al., 2013] Wu, L., Lin, W., Xiao, X., and Xu, Y. (2013). LSII : an indexing structure for exact real-time search on microblogs. In *Proc. Intl. Conf. on Data Engineering (ICDE)*, pages 482–493.

[Wu and Palmer, 1994] Wu, Z. and Palmer, M. (1994). Verbs Semantics and Lexical Selection. In *ACL*, pages 133–138.

[Yamaguchi et al., 2012] Yamaguchi, Y., Amagasa, T., and Kitagawa, H. (2012). Tagging users based on twitter lists. *Int. J. Web Eng. Technol.*, 7(3) :273–298.

[Yan and Garcia-Molina, 1994] Yan, T. W. and Garcia-Molina, H. (1994). Index Structures for Selective Dissemination of Information Under the Boolean Model. *ACM Transactions on Database Systems (TODS)*, 19(2) :332–364.

Mohamed Ryadh DAHIMENE

Filtrage et Recommandation sur les Réseaux Sociaux

Résumé

Ces dernières années, le contenu disponible sur le Web a augmenté de manière considérable dans ce qu'on appelle communément le Web social. Pour l'utilisateur moyen, il devient de plus en plus difficile de recevoir du contenu de qualité sans se voir rapidement submergé par le flot incessant de publications. Pour les fournisseurs de service, le passage à l'échelle reste problématique. L'objectif de cette thèse est d'aboutir à une meilleure expérience utilisateur à travers la mise en place de systèmes de filtrage et de recommandation. Le filtrage consiste à offrir la possibilité à un utilisateur de ne recevoir qu'un sous ensemble des publications des comptes auxquels il est abonné. Tandis que la recommandation permet la découverte d'information à travers la suggestion de comptes à suivre sur des sujets donnés. Nous avons élaboré MicroFilter un système de filtrage passant à l'échelle capable de gérer des flux issus du Web ainsi que RecLand, un système de recommandation qui tire parti de la topologie du réseau ainsi que du contenu afin de générer des recommandations pertinentes.

Mots-clés : Réseaux Sociaux, Filtrage, Recommandation, Indexation, Micro-blogging

Résumé en anglais

In the last years, the amount of available data on the social Web has exploded. For the average user, it became hard to find quality content without being overwhelmed with publications. For service providers, the scalability of such services became a challenging task. The aim of this thesis is to achieve a better user experience by offering the filtering and recommendation features. Filtering consists to provide for a given user, the ability of receiving only a subset of the publications from the direct network. Where recommendation allows content discovery by suggesting relevant content producers on given topics. We developed MicroFilter, a scalable filtering system able to handle Web-like data flows and RecLand, a recommender system that takes advantage of the network topology as well as the content in order to provide relevant recommendations.

Keywords : Social Networks, Filtering, Recommendation, Indexing, Micro-Blogging